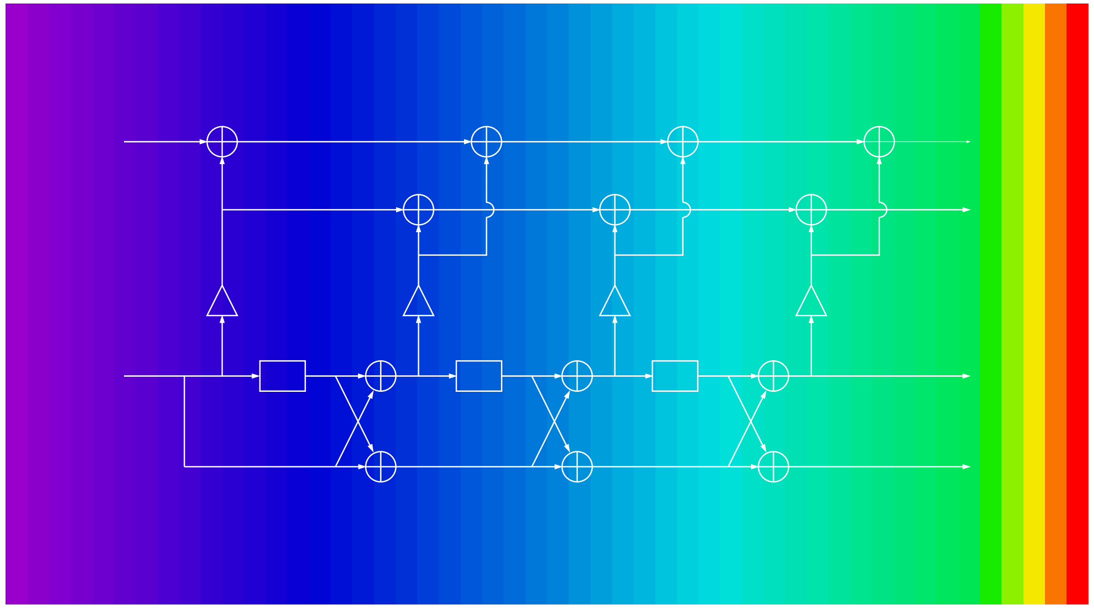


Applied Optimum Signal Processing

Sophocles J. Orfanidis

http://www.ece.rutgers.edu/~orfanidi/aosp

2018

Applied Optimum Signal Processing

Applied Optimum Signal Processing

A MATLAB-based Introduction

Sophocles J. Orfanidis

Rutgers University

2018

`http://www.ece.rutgers.edu/~orfanidi/aosp`

To my parents
John and Clio Orfanidis

And To
Monica, John, Anna, and Owen

This book is an updated and enlarged 2018 edition of *Optimum Signal Processing*, which was published in 2007 as a republication of the second edition published by McGraw-Hill Publishing Company, New York, NY, in 1988 (ISBN 0-07-047794-9), and also published earlier by Macmillan, Inc., New York, NY, 1988 (ISBN 0-02-389380-X). All copyrights to this work reverted to Sophocles J. Orfanidis in 1996.

Software tools:

MATLAB® is a registered trademark of The MathWorks, Inc.

OSP Toolbox – © S. J. Orfanidis 2018

CVX software by Michael Grant and Stephen Boyd, *CVX: Matlab software for disciplined convex programming*, version 2.0 beta, September 2013, `http://cvxr.com/cvx`.

Contents

1 Review of Random Signals 1

2 Signal Extraction Basics 104

3 Local Polynomial Filters 118

4 Minimum Roughness Filters 164

5 Local Polynomial Modeling 197

6 Exponential Smoothing 221

14 Spectrum Estimation and Array Processing 678

15 SVD and Signal Processing 765

16 Adaptive Filters 850

17 Appendices 923

References 930

Index 985

1

Review of Random Signals

1.1 Probability Density, Mean, Variance

In this section, we present a short review of probability concepts. It is assumed that the reader has some familiarity with the subject on the level of Papoulis' book [1].

Let x be a *random variable* having probability density $p(x)$. Its *mean*, *variance*, and *second moment* are defined by the expectation values

$$m = E[x] = \int_{-\infty}^{\infty} xp(x)\,dx = \text{mean}$$

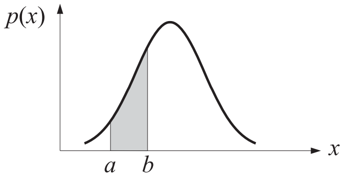
$$\sigma^2 = \text{Var}(x) = E\big[(x-m)^2\big] = \int_{-\infty}^{\infty}(x-m)^2 p(x)\,dx = \text{variance}$$

$$E[x^2] = \int_{-\infty}^{\infty} x^2 p(x)\,dx = \text{second moment}$$

These quantities are known as *second-order statistics* of the random variable x. Their importance is linked with the fact that most optimal filter design criteria require knowledge only of the second-order statistics and do not require more detailed knowledge, such as probability densities. It is necessary, then, to be able to extract such quantities from the actual measured data.

The probability that the random variable x will assume a value within an interval of values $[a,b]$ is given by

$$\text{Prob}[a \le x \le b] = \int_a^b p(x)\,dx = \text{shaded area}$$



The probability density is always normalized to unity by

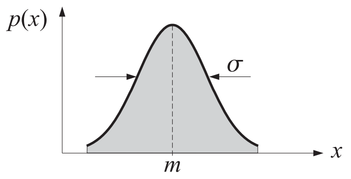
$$\int_{-\infty}^{\infty} p(x)\,dx = 1$$

which states that the probability of x taking a value somewhere within its range of variation is unity, that is, certainty. This property also implies

$$\sigma^2 = E\big[(x-m)^2\big] = E[x^2]-m^2$$

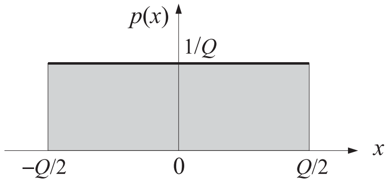
Example 1.1.1: Gaussian, or normal, distribution

$$p(x)=\frac{1}{\sqrt{2\pi}\sigma}\exp\big[-(x-m)^2/2\sigma^2\big]$$



Example 1.1.2: Uniform distribution

$$p(x)=\begin{cases}1/Q\,, & \text{for } -Q/2\le x\le Q/2\\ 0, & \text{otherwise}\end{cases}$$



Its variance is $\sigma^2 = Q^2/12$. □

Both the gaussian and the uniform distributions will prove to be important examples. In typical signal processing problems of designing filters to remove or separate noise from signal, it is often assumed that the noise interference is gaussian. This assumption is justified on the grounds of the central limit theorem, provided that the noise arises from many different noise sources acting independently of each other.

The uniform distribution is also important. In digital signal processing applications. the quantization error arising from the signal quantization in the A/D converters, or the roundoff error arising from the finite accuracy of the internal arithmetic operations in digital filters, can often be assumed to be uniformly distributed.

Every computer provides system routines for the generation of random numbers. For example, the routines RANDU and GAUSS of the IBM Scientific Subroutine Package generate uniformly distributed random numbers over the interval $[0,1]$, and gaussian-distributed numbers, respectively. GAUSS calls RANDU twelve times, thus generating twelve independent uniformly distributed random numbers $x_1, x_2, \dots, x_{12}$. Then, their sum $x = x_1+x_2+\cdots+x_{12}$, will be approximately gaussian, as guaranteed by the central limit theorem. It is interesting to note that the variance of x is unity, as it follows from the fact that the variance of each x_i, is $1/12$:

$$\sigma_x^2 = \sigma_{x_1}^2+\sigma_{x_2}^2+\cdots+\sigma_{x_{12}}^2 = \frac{1}{12}+\frac{1}{12}+\cdots+\frac{1}{12}=1$$

The mean of x is $12/2 = 6$. By shifting and scaling x, one can obtain a gaussian-distributed random number of any desired mean and variance.

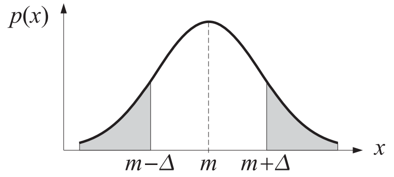
1.2 Chebyshev's Inequality

The variance σ^2 of a random variable x is a measure of the spread of the x-values about their mean. This intuitive interpretation of the variance is a direct consequence of Chebyshev's inequality, which states that the x-values tend to cluster about their mean in the sense that the probability of a value not occurring in the near vicinity of the mean is small; and it is smaller the smaller the variance.

More precisely, for any probability density $p(x)$ and any $\Delta > 0$, the probability that x will fall outside the interval of values $[m-\Delta, m+\Delta]$ is bounded by σ^2/Δ^2. Thus, for fixed Δ, as the variance σ^2 becomes smaller, the x-values tend to cluster more narrowly about the mean. In the extreme limiting case of a deterministic variable $x = m$, the density becomes infinitely narrow, $p(x)=\delta(x-m)$, and has zero variance.

$$\text{Prob}\big[|x-m|\ge\Delta\big]\le\frac{\sigma^2}{\Delta^2}$$

(Chebyshev's Inequality)



Chebyshev's inequality is especially important in proving asymptotic convergence results for sample estimates of parameters. For example, consider N independent samples $\{x_1, x_2, \dots, x_N\}$ drawn from a gaussian probability distribution of mean m and variance σ^2. The sample estimate of the mean is

$$\hat{m} = \frac{1}{N}(x_1+x_2+\cdots+x_N) \tag{1.2.1}$$

Being a sum of N gaussian random variables, $\hat{m}$ will itself be a gaussian random variable. Its probability density is completely determined by the corresponding mean and variance. These are found as follows.

$$E[\hat{m}]=\frac{1}{N}\big(E[x_1]+E[x_2]+\cdots+E[x_N]\big) = \frac{1}{N}(m+m+\cdots+m)=m$$

Therefore, $\hat{m}$ is an *unbiased estimator* of m. However, the goodness of $\hat{m}$ as an estimator must be judged by how small its variance is—the smaller the better, by Chebyshev's inequality. By the assumption of independence, we have

$$\text{var}(\hat{m})=E\big[(\hat{m}-m)^2\big] = \frac{1}{N^2}(\sigma_{x_1}^2+\sigma_{x_2}^2+\cdots+\sigma_{x_N}^2) = \frac{1}{N^2}(N\sigma^2) = \frac{\sigma^2}{N} \tag{1.2.2}$$

Thus, $\hat{m}$ is also a *consistent estimator* of m in the sense that its variance tends to zero as the number of samples N increases. The values of $\hat{m}$ will tend to cluster more and more closely about the true value of m as N becomes larger. Chebyshev's inequality implies that the probability of $\hat{m}$ falling outside any fixed neighborhood of m will tend to zero for large N. Equivalently, $\hat{m}$ will converge to m with probability one. This can also be seen from the probability density of $\hat{m}$, which is the gaussian

$$p(\hat{m})=\frac{N^{1/2}}{(2\pi)^{1/2}\sigma}\exp\Big[-\frac{N}{2\sigma^2}(\hat{m}-m)^2\Big]$$

In the limit of large N, this density tends to the infinitely narrow delta function density $p(\hat{m})= \delta(\hat{m}-m)$. In addition to the sample mean, we may also compute sample estimates of the variance σ^2 by

$$\hat{\sigma}^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i-\hat{m})^2 \tag{1.2.3}$$

It is easily shown [2,3] that this estimator is slightly biased. But for large N, it is *asymptotically unbiased* and *consistent* as can be seen from its mean and variance:

$$E[\hat{\sigma}^2]= \frac{N-1}{N}\sigma^2\,, \quad \mathrm{var}(\hat{\sigma}^2)= \frac{N-1}{N^2}\,2\sigma^4 \tag{1.2.4}$$

An unbiased and consistent estimator of σ^2 is the *standard deviation* defined by

$$s^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_i-\hat{m})^2 \tag{1.2.5}$$

It has $E[s^2]= \sigma^2$ and $\mathrm{var}(s^2)= 2\sigma^4/(N-1)$. In addition to the requirements of asymptotic unbiasedness and consistency, a good estimator of a parameter must also be judged in terms of its *efficiency* [2,3], which determines how closely the estimator meets its Cramér-Rao bound. This is discussed in Sec. 1.18. We will see there that the estimators (1.2.1) and (1.2.3)—being maximum likelihood estimators—are asymptotically efficient.

1.3 Joint and Conditional Densities, and Bayes' Rule

Next, we discuss random vectors. A pair of two different random variables $\mathbf{x} = (x_1, x_2)$ may be thought of as a vector-valued random variable. Its statistical description is more complicated than that of a single variable and requires knowledge of the joint probability density $p(x_1, x_2)$. The two random variables may or may not have any dependence on each other. It is possible, for example, that if x_2 assumes a particular value, then this fact may influence, or restrict, the possible values that x_1 can then assume.

A quantity that provides a measure for the degree of dependence of the two variables on each other is the conditional density $p(x_1|x_2)$ of x_1 given x_2; and $p(x_2|x_1)$ of x_2 given x_1. These are related by Bayes' rule

$$p(x_1,x_2)= p(x_1|x_2)p(x_2)= p(x_2|x_1)p(x_1)$$

More generally, Bayes' rule for two events A and B is

$$p(A,B)= p(A|B)p(B)= p(B|A)p(A)$$

The two random variables x_1 and x_2 are *independent* of each other if they do not condition each other in any way, that is, if

$$p(x_1|x_2)= p(x_1) \quad \text{or} \quad p(x_2|x_1)= p(x_2)$$

In other words, the occurrence of x_2 does not in any way influence the variable x_1. When two random variables are independent, their joint density factors into the product of single (marginal) densities:

$$p(x_1,x_2)= p(x_1)p(x_2)$$

The converse is also true. The *correlation* between x_1 and x_2 is defined by the expectation value

$$E[x_1x_2]= \iint x_1x_2p(x_1,x_2)\,dx_1dx_2$$

When x_1 and x_2 are independent, the correlation also factors as $E[x_1x_2]= E[x_1]E[x_2]$.

Example 1.3.1: Suppose x_1 is related to x_2 by

$$x_1 = 5x_2 + v$$

where v is a zero-mean, unit-variance, gaussian random variable assumed to be independent of x_2. Determine the conditional density and conditional mean of x_1 given x_2.

Solution: The randomness of x_1 arises both from the randomness of x_2 and the randomness of v. But if x_2 takes on a particular value, then the randomness of x_1 will arise only from v. Identifying elemental probabilities we have

$$p(x_1|x_2)dx_1 = p(v)dv = (2\pi)^{-1/2}\exp(-\frac{1}{2}v^2)dv$$

But, $dx_1 = dv$ and $v = x_1 - 5x_2$. Therefore,

$$p(x_1|x_2)= (2\pi)^{-1/2}\exp[-\frac{1}{2}(x_1-5x_2)^2]$$

The conditional mean is the mean of x_1 with respect to the density $p(x_1|x_2)$. It is evident from the above gaussian expression that the conditional mean is $E[x_1|x_2]= 5x_2$. This can also be found directly as follows.

$$E[x_1|x_2]= E\big[(5x_2+v)|x_2\big] = 5x_2 + E[v|x_2]= 5x_2$$

where we used the independence of v and x_2 to replace the conditional mean of v with its unconditional mean, which was given to be zero, that is, $E[v|x_2]= E[v]= 0$. □

The concept of a random vector generalizes to any dimension. A vector of N random variables

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}$$

requires knowledge of the joint density

$$p(\mathbf{x})= p(x_1,x_2,\dots,x_N) \tag{1.3.1}$$

for its complete statistical description. The second-order statistics of $\mathbf{x}$ are its mean, its *correlation matrix*, and its *covariance matrix*, defined by

$$\mathbf{m} = E[\mathbf{x}]\,,\quad R = E[\mathbf{x}\mathbf{x}^T]\,,\quad \Sigma = E\big[(\mathbf{x}-\mathbf{m})(\mathbf{x}-\mathbf{m})^T\big] \qquad (1.3.2)$$

where the superscript T denotes transposition, and the expectation operations are defined in terms of the joint density (1.3.1); for example,

$$E[\mathbf{x}] = \int \mathbf{x}\,p(\mathbf{x})\,d^N\mathbf{x}$$

where $d^N\mathbf{x} = dx_1 dx_2 \cdots dx_N$ denotes the corresponding N-dimensional volume element. The ijth matrix element of the correlation matrix R is the correlation between the ith random variable x_i with the jth random variable x_j, that is, $R_{ij} = E[x_i x_j]$. It is easily shown that the covariance and correlation matrices are related by

$$\Sigma = R - \mathbf{m}\mathbf{m}^T$$

When the mean is zero, R and Σ coincide. Both R and Σ are *symmetric positive semi-definite* matrices.

Example 1.3.2: The probability density of a gaussian random vector $\mathbf{x} = [x_1, x_2, \dots, x_N]^T$ is completely specified by its mean $\mathbf{m}$ and covariance matrix Σ, that is,

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{N/2}(\det\Sigma)^{1/2}}\,\exp\Big[-\frac{1}{2}(\mathbf{x}-\mathbf{m})^T\Sigma^{-1}(\mathbf{x}-\mathbf{m})\Big]$$

Example 1.3.3: Under a linear transformation, a gaussian random vector remains gaussian. Let $\mathbf{x}$ be a gaussian random vector of dimension N, mean $\mathbf{m}_x$, and covariance Σ_x. Show that the linearly transformed vector

$$\boldsymbol{\xi} = B\mathbf{x} \qquad \text{where } B \text{ is a nonsingular } N{\times}N \text{ matrix}$$

is gaussian-distributed with mean and covariance given by

$$\mathbf{m}_\xi = B\,\mathbf{m}_x\,,\quad \Sigma_\xi = B\Sigma_x B^T \qquad (1.3.3)$$

The relationships (1.3.3) are valid also for non-gaussian random vectors. They are easily derived as follows:

$$E[\boldsymbol{\xi}] = E[B\mathbf{x}] = BE[\mathbf{x}]\,,\quad E[\boldsymbol{\xi}\boldsymbol{\xi}^T] = E[B\mathbf{x}(B\mathbf{x})^T] = BE[\mathbf{x}\mathbf{x}^T]B^T$$

The probability density $p_\xi(\boldsymbol{\xi})$ is related to the density $p_x(\mathbf{x})$ by the requirement that, under the above change of variables, they both yield the same elemental probabilities:

$$p_\xi(\boldsymbol{\xi})\,d^N\boldsymbol{\xi} = p_x(\mathbf{x})\,d^N\mathbf{x} \qquad (1.3.4)$$

Since the Jacobian of the transformation from $\mathbf{x}$ to $\boldsymbol{\xi}$ is $d^N\boldsymbol{\xi} = |\det B| d^N\mathbf{x}$, we obtain $p_\xi(\boldsymbol{\xi}) = p_x(\mathbf{x})/|\det B|$. Noting the invariance of the quadratic form

$$\begin{aligned}(\boldsymbol{\xi}-\mathbf{m}_\xi)^T\Sigma_\xi^{-1}(\boldsymbol{\xi}-\mathbf{m}_\xi) &= (\mathbf{x}-\mathbf{m}_x)^T B^T (B\Sigma_x B^T)^{-1} B(\mathbf{x}-\mathbf{m}_x)\\ &= (\mathbf{x}-\mathbf{m}_x)^T\Sigma_x^{-1}(\mathbf{x}-\mathbf{m}_x)\end{aligned}$$

and that $\det\Sigma_\xi = \det(B\Sigma_x B^T) = (\det B)^2\det\Sigma_x$, we obtain

$$p_\xi(\boldsymbol{\xi}) = \frac{1}{(2\pi)^{N/2}(\det\Sigma_\xi)^{1/2}}\,\exp\Big[-\frac{1}{2}(\boldsymbol{\xi}-\mathbf{m}_\xi)^T\Sigma_\xi^{-1}(\boldsymbol{\xi}-\mathbf{m}_\xi)\Big]$$

Example 1.3.4: Consider two zero-mean random vectors $\mathbf{x}$ and $\mathbf{y}$ of dimensions N and M, respectively. Show that if they are *uncorrelated and jointly gaussian*, then they are also *independent* of each other. That $\mathbf{x}$ and $\mathbf{y}$ are jointly gaussian means that the $(N{+}M)$-dimensional joint vector $\mathbf{z} = \begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}$ is zero-mean and gaussian, that is,

$$p(\mathbf{z}) = \frac{1}{(2\pi)^{(N+M)/2}(\det R_{zz})^{1/2}}\,\exp\Big[-\frac{1}{2}\mathbf{z}^T R_{zz}^{-1}\mathbf{z}\Big]$$

where the correlation (covariance) matrix R_{zz} is

$$R_{zz} = E\left[\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}[\mathbf{x}^T, \mathbf{y}^T]\right] = \begin{bmatrix}E[\mathbf{x}\mathbf{x}^T] & E[\mathbf{x}\mathbf{y}^T]\\ E[\mathbf{y}\mathbf{x}^T] & E[\mathbf{y}\mathbf{y}^T]\end{bmatrix} = \begin{bmatrix}R_{xx} & R_{xy}\\ R_{yx} & R_{yy}\end{bmatrix}$$

If $\mathbf{x}$ and $\mathbf{y}$ are uncorrelated, that is, $R_{xy} = E[\mathbf{x}\mathbf{y}^T] = 0$, then the matrix R_{zz} becomes block diagonal and the quadratic form of the joint vector becomes the sum of the individual quadratic forms:

$$\mathbf{z}^T R_{zz}^{-1}\mathbf{z} = [\mathbf{x}^T, \mathbf{y}^T]\begin{bmatrix}R_{xx}^{-1} & 0\\ 0 & R_{yy}^{-1}\end{bmatrix}\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix} = \mathbf{x}^T R_{xx}^{-1}\mathbf{x} + \mathbf{y}^T R_{yy}^{-1}\mathbf{y}$$

Since $R_{xy} = 0$ also implies that $\det R_{zz} = (\det R_{xx})(\det R_{yy})$, it follows that the joint density $p(\mathbf{z}) = p(\mathbf{x}, \mathbf{y})$ factors into the marginal densities:

$$p(\mathbf{x}, \mathbf{y}) = p(\mathbf{x})p(\mathbf{y})$$

which shows the independence of $\mathbf{x}$ and $\mathbf{y}$.

Example 1.3.5: Given a random vector $\mathbf{x}$ with mean $\mathbf{m}$ and covariance Σ, show that the best choice of a *deterministic* vector $\hat{\mathbf{x}}$ which minimizes the quantity

$$R_{ee} = E[\mathbf{e}\mathbf{e}^T] = \text{minimum}\,,\quad \text{where } \mathbf{e} = \mathbf{x} - \hat{\mathbf{x}},$$

is the mean $\mathbf{m}$ itself, that is, $\hat{\mathbf{x}} = \mathbf{m}$. Also show that for this optimal choice of $\hat{\mathbf{x}}$, the actual minimum value of the quantity R_{ee} is the covariance Σ. This property is easily shown by working with the deviation of $\hat{\mathbf{x}}$ from the mean $\mathbf{m}$, that is, let

$$\hat{\mathbf{x}} = \mathbf{m} + \boldsymbol{\Delta}$$

Then, the quantity R_{ee} becomes

$$\begin{aligned}R_{ee} &= E[\mathbf{e}\mathbf{e}^T] = E\big[(\mathbf{x}-\mathbf{m}-\boldsymbol{\Delta})(\mathbf{x}-\mathbf{m}-\boldsymbol{\Delta})^T\big]\\ &= E\big[(\mathbf{x}-\mathbf{m})(\mathbf{x}-\mathbf{m})^T\big] - \boldsymbol{\Delta}E[\mathbf{x}^T - \mathbf{m}^T] - E[\mathbf{x}-\mathbf{m}]\boldsymbol{\Delta} + \boldsymbol{\Delta}\boldsymbol{\Delta}^T\\ &= \Sigma + \boldsymbol{\Delta}\boldsymbol{\Delta}^T\end{aligned}$$

where we used the fact that $E[\mathbf{x}-\mathbf{m}] = E[\mathbf{x}] - \mathbf{m} = 0$. Since the matrix $\boldsymbol{\Delta}\boldsymbol{\Delta}^T$ is nonnegative-definite, it follows that R_{ee}, will be minimized when $\boldsymbol{\Delta} = 0$, and in this case the minimum value will be $R_{ee}^{\min} = \Sigma$.

Since R_{ee} is a matrix, the sense in which it is minimized must be clarified. The statement that R_{ee} is greater than $R_{ee}^{\min}$ means that the difference $R_{ee} - R_{ee}^{\min}$ is a positive semi-definite (and symmetric) matrix, and therefore we have for the scalar quantities: $\mathbf{a}^T R_{ee}\mathbf{a} \geq \mathbf{a}^T R_{ee}^{\min}\mathbf{a}$ for any vector $\mathbf{a}$. □

1.4 Correlation Canceling and Optimum Estimation

The concept of correlation canceling plays a central role in the development of many optimum signal processing algorithms, because a correlation canceler is also the best linear processor for estimating one signal from another.

Consider two zero-mean random vectors $\mathbf{x}$ and $\mathbf{y}$ of dimensions N and M, respectively. If $\mathbf{x}$ and $\mathbf{y}$ are correlated with each other in the sense that $R_{xy} = E[\mathbf{x}\mathbf{y}^T] \neq 0$, then we may remove such correlations by means of a linear transformation of the form

$$\mathbf{e} = \mathbf{x} - H\mathbf{y} \tag{1.4.1}$$

where the $N\times M$ matrix H must be suitably chosen such that the new pair of vectors $\mathbf{e}, \mathbf{y}$ are no longer correlated with each other, that is, we require

$$R_{ey} = E[\mathbf{e}\mathbf{y}^T] = 0 \tag{1.4.2}$$

Using Eq. (1.4.1), we obtain

$$R_{ey} = E[\mathbf{e}\mathbf{y}^T] = E\big[(\mathbf{x} - H\mathbf{y})\mathbf{y}^T\big] = E[\mathbf{x}\mathbf{y}^T] - HE[\mathbf{y}\mathbf{y}^T] = R_{xy} - HR_{yy}$$

Then, the condition $R_{ey} = 0$ immediately implies that

$$\boxed{H = R_{xy}R_{yy}^{-1} = E[\mathbf{x}\mathbf{y}^T]E[\mathbf{y}\mathbf{y}^T]^{-1}} \tag{1.4.3}$$

Using $R_{ey} = 0$, the covariance matrix of the resulting vector $\mathbf{e}$ is easily found to be

$$R_{ee} = E[\mathbf{e}\mathbf{e}^T] = E\big[\mathbf{e}(\mathbf{x}^T - \mathbf{y}^T H)\big] = R_{ex} - R_{ey}H^T = R_{ex} = E\big[(\mathbf{x} - H\mathbf{y})\mathbf{x}^T\big]\,, \quad \text{or,}$$

$$R_{ee} = R_{xx} - HR_{yx} = R_{xx} - R_{xy}R_{yy}^{-1}R_{yx} \tag{1.4.4}$$

The vector

$$\hat{\mathbf{x}} = H\mathbf{y} = R_{xy}R_{yy}^{-1}\mathbf{y} = E[\mathbf{x}\mathbf{y}^T]E[\mathbf{y}\mathbf{y}^T]^{-1}\mathbf{y} \tag{1.4.5}$$

obtained by linearly processing the vector $\mathbf{y}$ by the matrix H is called the *linear regression*, or *orthogonal projection*, of $\mathbf{x}$ on the vector $\mathbf{y}$. In a sense to be made precise later, $\hat{\mathbf{x}}$ also represents the best "copy," or *estimate*, of $\mathbf{x}$ that can be made on the basis of the vector $\mathbf{y}$. Thus, the vector $\mathbf{e} = \mathbf{x} - H\mathbf{y} = \mathbf{x} - \hat{\mathbf{x}}$ may be thought of as the *estimation error*.

Actually, it is better to think of $\hat{\mathbf{x}} = H\mathbf{y}$ not as an estimate of $\mathbf{x}$ but rather as an estimate of *that part* of $\mathbf{x}$ which is correlated with $\mathbf{y}$. Indeed, suppose that $\mathbf{x}$ consists of two parts

$$\mathbf{x} = \mathbf{x}_1 + \mathbf{x}_2$$

such that $\mathbf{x}_1$ is correlated with $\mathbf{y}$, but $\mathbf{x}_2$ is not, that is, $R_{x_2y} = E[\mathbf{x}_2\mathbf{y}^T] = 0$. Then,

$$R_{xy} = E[\mathbf{x}\mathbf{y}^T] = E\big[(\mathbf{x}_1 + \mathbf{x}_2)\mathbf{y}^T\big] = R_{x_1y} + R_{x_2y} = R_{x_1y}$$

and therefore,

$$\hat{\mathbf{x}} = R_{xy}R_{yy}^{-1}\mathbf{y} = R_{x_1y}R_{yy}^{-1}\mathbf{y} = \hat{\mathbf{x}}_1$$

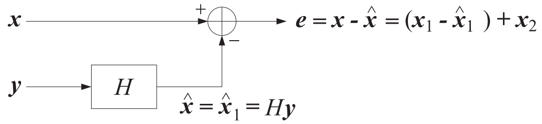


Fig. 1.4.1 Correlation canceler.

The vector $\mathbf{e} = \mathbf{x} - \hat{\mathbf{x}} = \mathbf{x}_1 + \mathbf{x}_2 - \hat{\mathbf{x}}_1 = (\mathbf{x}_1 - \hat{\mathbf{x}}_1) + \mathbf{x}_2$ consists of the estimation error $(\mathbf{x}_1 - \hat{\mathbf{x}}_1)$ of the $\mathbf{x}_1$-part plus the $\mathbf{x}_2$-part. Both of these terms are separately uncorrelated from $\mathbf{y}$. These operations are summarized in block diagram form in Fig. 1.4.1.

The most important feature of this arrangement is the *correlation cancellation* property which may be summarized as follows: If $\mathbf{x}$ has a part $\mathbf{x}_1$ which is correlated with $\mathbf{y}$, then this part will tend to be canceled as much as possible from the output $\mathbf{e}$. The linear processor H accomplishes this by converting $\mathbf{y}$ into the *best possible copy* $\hat{\mathbf{x}}_1$ of $\mathbf{x}_1$ and then proceeds to cancel it from the output. The output vector $\mathbf{e}$ is no longer correlated with $\mathbf{y}$. The part $\mathbf{x}_2$ of $\mathbf{x}$ which is uncorrelated with $\mathbf{y}$ remains entirely unaffected. It cannot be estimated in terms of $\mathbf{y}$.

The correlation canceler may also be thought of as an *optimal signal separator*. Indeed, the output of the processor H is essentially the $\mathbf{x}_1$ component of $\mathbf{x}$, whereas the output $\mathbf{e}$ is essentially the $\mathbf{x}_2$ component. The separation of $\mathbf{x}$ into $\mathbf{x}_1$ and $\mathbf{x}_2$ is optimal in the sense that the $\mathbf{x}_1$ component of $\mathbf{x}$ is removed as much as possible from $\mathbf{e}$.

Next, we discuss the *best linear estimator property* of the correlation canceler. The choice $H = R_{xy}R_{yy}^{-1}$, which guarantees correlation cancellation, is also the choice that gives the *best estimate* of $\mathbf{x}$ as a *linear* function of $\mathbf{y}$ in the form $\hat{\mathbf{x}} = H\mathbf{y}$. It is the best estimate in the sense that it produces the lowest *mean-square* estimation error. To see this, express the covariance matrix of the estimation error in terms of H, as follows:

$$R_{ee} = E[\mathbf{e}\mathbf{e}^T] = E\big[(\mathbf{x} - H\mathbf{y})(\mathbf{x}^T - \mathbf{y}^T H^T)\big] = R_{xx} - HR_{yx} - R_{xy}H^T + HR_{yy}H^T \tag{1.4.6}$$

Minimizing this expression with respect to H yields the optimum choice of H:

$$H_{\text{opt}} = R_{xy}R_{yy}^{-1}$$

with the minimum value for R_{ee} given by:

$$R_{ee}^{\text{min}} = R_{xx} - R_{xy}R_{yy}^{-1}R_{yx}$$

Any other value will result in a larger value for R_{ee}. An alternative way to see this is to consider a deviation ΔH of H from its optimal value, that is, in (1.4.5) replace H by

$$H = H_{\text{opt}} + \Delta H = R_{xy}R_{yy}^{-1} + \Delta H$$

Then Eq. (1.4.6) may be expressed in terms of ΔH as follows:

$$R_{ee} = R_{ee}^{\text{min}} + \Delta H\, R_{yy}\, \Delta H^T$$

Since R_{yy} is positive definite, the second term always represents a nonnegative contribution above the minimum value R_{ee}^{min}, so that $\big(R_{ee} - R_{ee}^{\text{min}}\big)$ is positive semi-definite. In summary, there are three useful ways to think of the correlation canceler:

1. Optimal estimator of $\mathbf{x}$ from $\mathbf{y}$.
2. Optimal canceler of that part of $\mathbf{x}$ which is correlated with $\mathbf{y}$.
3. Optimal signal separator

The point of view is determined by the application. The first view is typified by Kalman filtering, channel equalization, and linear prediction applications. The second view is taken in echo canceling, noise canceling, and sidelobe canceling applications. The third view is useful in the adaptive line enhancer, which is a method of adaptively separating a signal into its broadband and narrowband components. All of these applications are considered later on.

Example 1.4.1: If $\mathbf{x}$ and $\mathbf{y}$ are *jointly gaussian*, show that the linear estimate $\hat{\mathbf{x}} = H\mathbf{y}$ is also the *conditional mean* $E[\mathbf{x}|\mathbf{y}]$ of the vector $\mathbf{x}$ given the vector $\mathbf{y}$. The conditional mean is defined in terms of the conditional density $p(\mathbf{x}|\mathbf{y})$ of $\mathbf{x}$ given $\mathbf{y}$ as follows:

$$E[\mathbf{x}|\mathbf{y}] = \int \mathbf{x}\, p(\mathbf{x}|\mathbf{y})\, d^N\mathbf{x}$$

Instead of computing this integral, we will use the results of Examples 1.3.3 and 1.3.4. The transformation from the jointly gaussian pair $(\mathbf{x},\mathbf{y})$ to the uncorrelated pair $(\mathbf{e},\mathbf{y})$ is linear:

$$\begin{bmatrix}\mathbf{e}\\ \mathbf{y}\end{bmatrix} = \begin{bmatrix} I_N & -H \\ 0 & I_M\end{bmatrix}\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}$$

where I_N and I_M are the unit matrices of dimensions N and M, respectively. Therefore, Example 1.3.3 implies that the transformed pair $(\mathbf{e},\mathbf{y})$ is also jointly gaussian. Furthermore, since $\mathbf{e}$ and $\mathbf{y}$ are uncorrelated, it follows from Example 1.3.4 that they must be independent of each other. The conditional mean of $\mathbf{x}$ can be computed by writing

$$\mathbf{x} = \hat{\mathbf{x}} + \mathbf{e} = H\mathbf{y} + \mathbf{e}$$

and noting that if $\mathbf{y}$ is given, then $H\mathbf{y}$ is no longer random. Therefore,

$$E[\mathbf{x}|\mathbf{y}] = E\big[(H\mathbf{y}+\mathbf{e})|\mathbf{y}\big] = H\mathbf{y} + E[\mathbf{e}|\mathbf{y}]$$

Since $\mathbf{e}$ and $\mathbf{y}$ are independent, the conditional mean $E[\mathbf{e}|\mathbf{y}]$ is the same as the unconditional mean $E[\mathbf{e}]$, which is zero by the zero-mean assumption. Thus,

$$E[\mathbf{x}|\mathbf{y}] = H\mathbf{y} = R_{xy}R_{yy}^{-1}\mathbf{y} \qquad \text{(jointly gaussian } \mathbf{x} \text{ and } \mathbf{y}) \tag{1.4.7}$$

Example 1.4.2: Show that the conditional mean $E[\mathbf{x}|\mathbf{y}]$ is the best *unrestricted* (i.e., not necessarily linear) estimate of $\mathbf{x}$ in the *mean-square* sense. The best linear estimate was obtained by seeking the best linear function of $\mathbf{y}$ that minimized the error criterion (1.4.6), that is, we required a priori that the estimate was to be of the form $\hat{\mathbf{x}} = H\mathbf{y}$. Here, our task is more general: find the most general function of $\mathbf{y}$, $\hat{\mathbf{x}} = \hat{\mathbf{x}}(\mathbf{y})$, which gives the best estimate of $\mathbf{x}$, in the sense of producing the lowest mean-squared estimation error $\mathbf{e} = \mathbf{x} - \hat{\mathbf{x}}(\mathbf{y})$,

$$R_{ee} = E[\mathbf{e}\mathbf{e}^T] = E\big[(\mathbf{x}-\hat{\mathbf{x}}(\mathbf{y}))(\mathbf{x}^T - \hat{\mathbf{x}}(\mathbf{y})^T)\big] = \min$$

The functional dependence of $\hat{\mathbf{x}}(\mathbf{y})$ on $\mathbf{y}$ is not required to be linear a priori. Using $p(\mathbf{x},\mathbf{y}) = p(\mathbf{x}|\mathbf{y})p(\mathbf{y})$, the above expectation may be written as

$$\begin{aligned} R_{ee} &= \int (\mathbf{x}-\hat{\mathbf{x}}(\mathbf{y}))(\mathbf{x}^T - \hat{\mathbf{x}}(\mathbf{y})^T)\, p(\mathbf{x},\mathbf{y})\, d^N\mathbf{x}\, d^M\mathbf{y} \\ &= \int p(\mathbf{y})\, d^M\mathbf{y}\left[\int (\mathbf{x}-\hat{\mathbf{x}}(\mathbf{y}))(\mathbf{x}^T - \hat{\mathbf{x}}(\mathbf{y})^T)\, p(\mathbf{x}|\mathbf{y})\, d^N\mathbf{x}\right]\end{aligned}$$

Since $p(\mathbf{y})$ is nonnegative for all $\mathbf{y}$, it follows that R_{ee} will be minimized when the quantity

$$\int (\mathbf{x}-\hat{\mathbf{x}}(\mathbf{y}))(\mathbf{x}^T - \hat{\mathbf{x}}(\mathbf{y})^T)\, p(\mathbf{x}|\mathbf{y})\, d^N\mathbf{x}$$

is minimized with respect to $\hat{\mathbf{x}}$. But we know from Example 1.3.5 that this quantity is minimized when $\hat{\mathbf{x}}$ is chosen to be the corresponding mean; here, this is the mean with respect to the density $p(\mathbf{x}|\mathbf{y})$. Thus,

$$\hat{\mathbf{x}}(\mathbf{y}) = E[\mathbf{x}|\mathbf{y}] \tag{1.4.8}$$

To summarize, we have seen that

$$\hat{\mathbf{x}} = H\mathbf{y} = R_{xy}R_{yy}^{-1}\mathbf{y} = \text{best } \textit{linear} \text{ mean-square estimate of } \mathbf{x}$$

$$\hat{\mathbf{x}} = E[\mathbf{x}|\mathbf{y}] = \text{best } \textit{unrestricted} \text{ mean-square estimate of } \mathbf{x}$$

and Example 1.4.1 shows that the two are equal in the case of jointly gaussian vectors $\mathbf{x}$ and $\mathbf{y}$.

The concept of correlation canceling and its application to signal estimation problems will be discussed in more detail in Chap. 11. The adaptive implementation of the correlation canceler will be discussed in Chap. 16. In a typical signal processing application, the processor H would represent a *linear filtering operation* and the vectors $\mathbf{x}$ and $\mathbf{y}$ would be *blocks of signal samples*. The design of such processors requires knowledge of the quantities $R_{xy} = E[\mathbf{x}\mathbf{y}^T]$ and $R_{yy} = E[\mathbf{y}\mathbf{y}^T]$. How does one determine these? Basically, applications fall into two classes:

1. Both $\mathbf{x}$ and $\mathbf{y}$ are available for processing and the objective is to cancel the correlations that may exist between them.

2. Only the signal $\mathbf{y}$ is available for processing and the objective is to estimate the signal $\mathbf{x}$ on the basis of $\mathbf{y}$.

In the first class of applications, there exist two basic design approaches:

a. *Block processing* (off-line) methods. The required correlations R_{xy} and R_{yy} are computed on the basis of two actual blocks of signal samples $\mathbf{x}$ and $\mathbf{y}$ by replacing statistical averages by time averages.

b. *Adaptive processing* (on-line) methods. The quantities R_{xy} and R_{yy} are "learned" gradually as the data $\mathbf{x}$ and $\mathbf{y}$ become available in real time. The processor H is continually updated in response to the incoming data, until it reaches its optimal value.

Both methods are *data adaptive*. The first is adaptive on a *block-by-block* basis, whereas the second on a *sample-by-sample* basis. Both methods depend heavily on the assumption of *stationarity*. In block processing methods, the replacement of ensemble averages by time averages is justified by the assumption of ergodicity, which requires stationarity. The requirement of stationarity can place serious limitations on the allowed length of the signal blocks $\mathbf{x}$ and $\mathbf{y}$.

Similarly, in adaptive processing methods, convergence to the optimal value of the processor H again requires stationarity. Adaptive methods offer, however, the possibility of tracking nonstationary changes of the environment, as long as such changes occur slowly enough to allow convergence between changes. Thus, the issue of the speed of convergence of adaptation algorithms is an important one.

In the second class of applications where $\mathbf{x}$ is not available for processing, one must have a specific model of the relationship between $\mathbf{x}$ and $\mathbf{y}$ from which R_{xy} and R_{yy} may be calculated. This is, for example, what is done in Kalman filtering.

Example 1.4.3: As an example of the relationship that might exist between $\mathbf{x}$ and $\mathbf{y}$, let

$$y_n = xc_n + v_n\,,\quad n = 1,2,\dots,M$$

where x and v_n are zero-mean, unit-variance, random variables, and c_n are known coefficients. It is further assumed that v_n are mutually uncorrelated, and also uncorrelated with x, so that $E[v_n v_m] = \delta_{nm}$, $E[xv_n] = 0$. We would like to determine the optimal linear estimate (1.4.5) of x, and the corresponding estimation error (1.4.4). In obvious matrix notation we have $\mathbf{y} = \mathbf{c}\,x + \mathbf{v}$, with $E[x\mathbf{v}] = 0$ and $E[\mathbf{v}\mathbf{v}^T] = I$, where I is the $M\times M$ unit matrix. We find

$$E[x\mathbf{y}^T] = E\big[x(x\mathbf{c}+\mathbf{v})^T\big] = \mathbf{c}^T E[x^2] + E[x\mathbf{v}^T] = \mathbf{c}^T$$
$$E[\mathbf{y}\mathbf{y}^T] = E\big[(x\mathbf{c}+\mathbf{v})(x\mathbf{c}+\mathbf{v})^T\big] = \mathbf{c}\mathbf{c}^T E[x^2] + E[\mathbf{v}\mathbf{v}^T] = \mathbf{c}\mathbf{c}^T + I$$

and therefore, $H = E[x\mathbf{y}^T]E[\mathbf{y}\mathbf{y}^T]^{-1} = \mathbf{c}^T(I+\mathbf{c}\mathbf{c}^T)^{-1}$. Using the matrix inversion lemma we may write $(I+\mathbf{c}\mathbf{c}^T)^{-1} = I - \mathbf{c}(1+\mathbf{c}^T\mathbf{c})^{-1}\mathbf{c}^T$, so that

$$H = \mathbf{c}^T\big[I - \mathbf{c}(1+\mathbf{c}^T\mathbf{c})^{-1}\mathbf{c}^T\big] = (1+\mathbf{c}^T\mathbf{c})^{-1}\mathbf{c}^T$$

The optimal estimate of x is then

$$\hat{x} = H\mathbf{y} = (1+\mathbf{c}^T\mathbf{c})^{-1}\mathbf{c}^T\mathbf{y} \tag{1.4.9}$$

The corresponding estimation error is computed by

$$E[e^2] = R_{ee} = R_{xx} - HR_{yy} = 1 - (1+\mathbf{c}^T\mathbf{c})^{-1}\mathbf{c}^T\mathbf{c} = (1+\mathbf{c}^T\mathbf{c})^{-1}$$

1.5 Regression Lemma

The regression lemma is a key result in the derivation of the Kalman filter. The optimum estimate and estimation error of a (zero-mean) random vector $\mathbf{x}$ based on a (zero-mean) vector of observations $\mathbf{y}_1$ are given by

$$\hat{\mathbf{x}}_1 = R_{xy_1}R_{y_1y_1}^{-1}\mathbf{y}_1 = E[\mathbf{x}\mathbf{y}_1^T]E[\mathbf{y}_1\mathbf{y}_1^T]^{-1}\mathbf{y}_1$$
$$\mathbf{e}_1 = \mathbf{x} - \hat{\mathbf{x}}_1$$
$$R_{e_1e_1} = E[\mathbf{e}_1\mathbf{e}_1^T] = R_{xx} - R_{xy_1}R_{y_1y_1}^{-1}R_{y_1x}$$

If the observation set is enlarged by adjoining to it a new set of observations $\mathbf{y}_2$, so that the enlarged observation vector is $\mathbf{y} = \begin{bmatrix}\mathbf{y}_1\\ \mathbf{y}_2\end{bmatrix}$, the corresponding estimate of $\mathbf{x}$ will given by,

$$\hat{\mathbf{x}} = R_{xy}R_{yy}^{-1}\mathbf{y} = \big[R_{xy_1}, R_{xy_2}\big]\begin{bmatrix}R_{y_1y_1} & R_{y_1y_2}\\ R_{y_2y_1} & R_{y_2y_2}\end{bmatrix}^{-1}\begin{bmatrix}\mathbf{y}_1\\ \mathbf{y}_2\end{bmatrix}$$

The *regression lemma* states that $\hat{\mathbf{x}}$ can be obtained by the following alternative expression of updating $\hat{\mathbf{x}}_1$ by the addition of a correction term,

$$\boxed{\hat{\mathbf{x}} = \hat{\mathbf{x}}_1 + R_{x\varepsilon_2}R_{\varepsilon_2\varepsilon_2}^{-1}\boldsymbol{\varepsilon}_2}\quad\text{(regression lemma)} \tag{1.5.1}$$

where $\boldsymbol{\varepsilon}_2$ is the innovations residual obtained by removing from $\mathbf{y}_2$ that part which is predictable from $\mathbf{y}_1$, that is,

$$\boldsymbol{\varepsilon}_2 = \mathbf{y}_2 - \hat{\mathbf{y}}_{2/1} = \mathbf{y}_2 - R_{y_2y_1}R_{y_1y_1}^{-1}\mathbf{y}_1$$

The improvement in using more observations is quantified by the following result, which shows that the mean-square error is reduced:

$$\mathbf{e} = \mathbf{x} - \hat{\mathbf{x}} \quad\Rightarrow\quad R_{ee} = R_{e_1e_1} - R_{x\varepsilon_2}R_{\varepsilon_2\varepsilon_2}^{-1}R_{\varepsilon_2x} \tag{1.5.2}$$

where we defined,

$$R_{x\varepsilon_2} = R_{\varepsilon_2x}^T = E[\mathbf{x}\boldsymbol{\varepsilon}_2^T]\,,\quad R_{\varepsilon_2\varepsilon_2} = E[\boldsymbol{\varepsilon}_2\boldsymbol{\varepsilon}_2^T]$$

The proof of Eq. (1.5.1) is straightforward and is left as an exercise. As a hint, the following property may be used,

$$\begin{bmatrix}\mathbf{y}_1\\ \mathbf{y}_2\end{bmatrix} = \begin{bmatrix}I & 0\\ H & I\end{bmatrix}\begin{bmatrix}\mathbf{y}_1\\ \boldsymbol{\varepsilon}_2\end{bmatrix},\quad \begin{bmatrix}R_{y_1y_1} & R_{y_1y_2}\\ R_{y_2y_1} & R_{y_2y_2}\end{bmatrix} = \begin{bmatrix}I & 0\\ H & I\end{bmatrix}\begin{bmatrix}R_{y_1y_1} & 0\\ 0 & R_{\varepsilon_2\varepsilon_2}\end{bmatrix}\begin{bmatrix}I & 0\\ H & I\end{bmatrix}^T$$

where $H = R_{y_2y_1}R_{y_1y_1}^{-1}$. A special case of this lemma is discussed next.

1.6 Gram-Schmidt Orthogonalization

In the previous section, we saw that any random vector $\mathbf{x}$ may be decomposed relative to another vector $\mathbf{y}$ into two parts, $\mathbf{x} = \hat{\mathbf{x}} + \mathbf{e}$, one part which is correlated with $\mathbf{y}$, and one which is not. These two parts are uncorrelated with each other since $R_{e\hat{x}} = E[\mathbf{e}\hat{\mathbf{x}}^T] = E[\mathbf{e}\mathbf{y}^T H^T] = E[\mathbf{e}\mathbf{y}^T]H^T = 0$. In a sense, they are orthogonal to each other. In this section, we will briefly develop such a geometrical interpretation.

The usefulness of the geometrical approach is threefold: First, it provides a very simple and intuitive framework in which to formulate and understand signal estimation problems. Second, through the Gram-Schmidt orthogonalization process, it provides the basis for making *signal models*, which find themselves in a variety of signal processing applications, such as speech synthesis, data compression, and modern methods of spectrum estimation. Third, again through the Gram-Schmidt construction, by decorrelating the given set of observations it provides the most convenient basis to work with, containing no redundancies. Linear estimates expressed in the decorrelated basis become computationally efficient.

Geometrical ideas may be introduced by thinking of the space of random variables under consideration as a *linear vector space* [7]. For example, in the previous section we dealt with the multicomponent random variables $\mathbf{x}$ and $\mathbf{y}$ consisting, say, of the random variables $\{x_1, x_2, \dots, x_N\}$ and $\{y_1, y_2, \dots, y_M\}$, respectively. In this case, the space of random variables under consideration is the set

$$\{x_1, x_2, \dots, x_N,\ y_1, y_2, \dots, y_M\} \tag{1.6.1}$$

Since any linear combination of random variables from this set is itself a random variable, the above set may be enlarged by adjoining to it all such possible linear combinations. This is the linear vector space *generated* or spanned by the given set of random variables. The next step is to convert this vector space into an *inner-product space* (a Hilbert space) by defining an inner product between any two random variables u and v as follows:

$$(u, v) = E[uv] \tag{1.6.2}$$

With this definition of an inner product, "orthogonal" means "uncorrelated." The *distance* between u and v is defined by the norm $\|u - v\|$ induced by the above inner product:

$$\|u - v\|^2 = E\big[(u - v)^2\big] \tag{1.6.3}$$

Mutually orthogonal (i.e., uncorrelated) random variables may be used to define *orthogonal bases*. Consider, for example, M mutually orthogonal random variables $\{\epsilon_1, \epsilon_2, \dots, \epsilon_M\}$, such that

$$(\epsilon_i, \epsilon_j) = E[\epsilon_i \epsilon_j] = 0\,, \quad \text{if } i \neq j \tag{1.6.4}$$

and let $Y = \{\epsilon_1, \epsilon_2, \dots, \epsilon_M\}$ be the linear subspace *spanned* by these M random variables. Without loss of generality, we may assume that the ϵ_is are linearly independent; therefore, they form a linearly independent and orthogonal basis for the subspace Y.

One of the standard results on linear vector spaces is the *orthogonal decomposition theorem* [8], which in our context may be stated as follows: Any random variable x may be decomposed uniquely, with respect to a subspace Y, into two mutually orthogonal parts. One part is *parallel* to the subspace Y (i.e., it lies in it), and the other is *perpendicular* to it. That is,

$$x = \hat{x} + e \qquad \text{with } \hat{x} \in Y \text{ and } e \perp Y \tag{1.6.5}$$

The component $\hat{x}$ is called the *orthogonal projection* of x onto the subspace Y. This decomposition is depicted in Fig. 1.6.1. The orthogonality condition $e \perp Y$ means that e must be orthogonal to every vector in Y; or equivalently, to every basis vector ϵ_i,

$$(e, \epsilon_i) = E[e\epsilon_i] = 0\,, \quad i = 1, 2, \dots, M \tag{1.6.6}$$

Since the component $\hat{x}$ lies in Y, it may be expanded in terms of the orthogonal basis in the form

$$\hat{x} = \sum_{i=1}^{M} a_i \epsilon_i$$

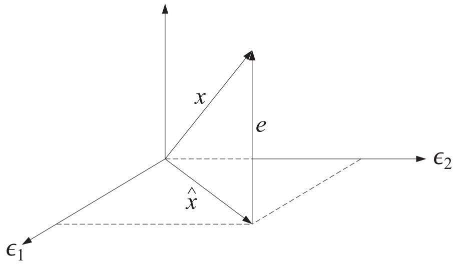


Fig. 1.6.1 Orthogonal decomposition with respect to $Y = \{\epsilon_1, \epsilon_2\}$.

The coefficients a_i can be determined using the orthogonality equations (1.6.6), as follows,

$$\begin{aligned}(x, \epsilon_i) &= (\hat{x} + e, \epsilon_i) = (\hat{x}, \epsilon_i) + (e, \epsilon_i) = (\hat{x}, \epsilon_i) \\ &= \left(\sum_{j=1}^{M} a_j \epsilon_j,\ \epsilon_i\right) = \sum_{j=1}^{M} a_j (\epsilon_j, \epsilon_i) = a_i (\epsilon_i, \epsilon_i)\end{aligned}$$

where in the last equality we used Eq. (1.6.4). Thus, $a_i = (x, \epsilon_i)(\epsilon_i, \epsilon_i)^{-1}$. or, $a_i = E[x\epsilon_i]E[\epsilon_i\epsilon_i]^{-1}$, and we can write Eq. (1.6.5) as

$$x = \hat{x} + e = \sum_{i=1}^{M} E[x\epsilon_i]E[\epsilon_i\epsilon_i]^{-1}\epsilon_i + e \tag{1.6.7}$$

Eq. (1.6.7) may also be written in a compact matrix form by introducing the M-vector,

$$\boldsymbol{\epsilon} = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_M \end{bmatrix}$$

the corresponding cross-correlation M-vector,

$$E[x\boldsymbol{\epsilon}] = \begin{bmatrix} E[x\epsilon_1] \\ E[x\epsilon_2] \\ \vdots \\ E[x\epsilon_M] \end{bmatrix}$$

and the correlation matrix $R_{\epsilon\epsilon} = E[\boldsymbol{\epsilon}\boldsymbol{\epsilon}^T]$, which is *diagonal* because of Eq. (1.6.4):

$$R_{\epsilon\epsilon} = E[\boldsymbol{\epsilon}\boldsymbol{\epsilon}^T] = \text{diag}\{E[\epsilon_1^2], E[\epsilon_2^2], \dots, E[\epsilon_M^2]\}$$

Then, Eq. (1.6.7) may be written as

$$x = \hat{x} + e = E[x\boldsymbol{\epsilon}^T]E[\boldsymbol{\epsilon}\boldsymbol{\epsilon}^T]^{-1}\boldsymbol{\epsilon} + e \tag{1.6.8}$$

The orthogonality equations (1.6.6) can be written as

$$R_{e\epsilon} = E[e\boldsymbol{\epsilon}^T] = 0 \tag{1.6.9}$$

Equations (1.6.8) and (1.6.9) represent the unique orthogonal decomposition of any random variable x relative to a linear subspace Y of random variables. If one has a collection of N random variables $\{x_1, x_2, \dots, x_N\}$, then each one may be orthogonally decomposed with respect to the same subspace Y, giving $x_i = \hat{x}_i + e_i$, $i = 1, 2, \dots, N$. These may be grouped together into a compact matrix form as

$$\mathbf{x} = \hat{\mathbf{x}} + \mathbf{e} = E[\mathbf{x}\boldsymbol{\epsilon}^T]E[\boldsymbol{\epsilon}\boldsymbol{\epsilon}^T]^{-1}\boldsymbol{\epsilon} + \mathbf{e} \tag{1.6.10}$$

where $\mathbf{x}$ stands for the column N-vector $\mathbf{x} = [x_1, x_2, \dots, x_N]^T$, and so on. This is identical to the correlation canceler decomposition of the previous section.

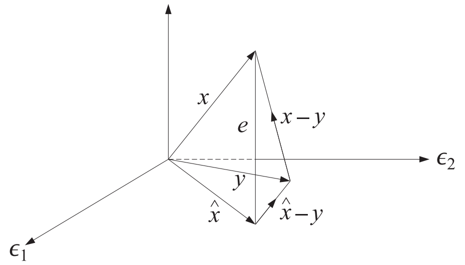
Next, we briefly discuss the *orthogonal projection theorem*. In Sec. 1.4, we noted the best linear estimator property of the correlation canceler decomposition. The same result may be understood geometrically by means of the orthogonal projection theorem, which states: The orthogonal projection $\hat{x}$ of a vector x onto a linear subspace Y is that vector in Y that lies closest to x with respect to the distance induced by the inner product of the vector space.

The theorem is a simple consequence of the orthogonal decomposition theorem and the Pythagorean theorem. Indeed, let $x = \hat{x} + e$ be the unique orthogonal decomposition of x with respect to Y, so that $\hat{x} \in Y$ and $e \perp Y$ and let y be an arbitrary vector in Y; noting that $(\hat{x} - y) \in Y$ and therefore $e \perp (\hat{x} - y)$, we have

$$\|x - y\|^2 = \|(\hat{x} - y) + e\|^2 = \|\hat{x} - y\|^2 + \|e\|^2$$

or, in terms of Eq. (1.6.3),

$$E\big[(x - y)^2\big] = E\big[(\hat{x} - y)^2\big] + E[e^2]$$



Since the vector y varies over the subspace Y, it follows that the above quantity will be minimized when $y = \hat{x}$. In summary, $\hat{x}$ represents the best approximation of x that can be made as a linear function of the random variables in Y in the minimum mean-square sense.

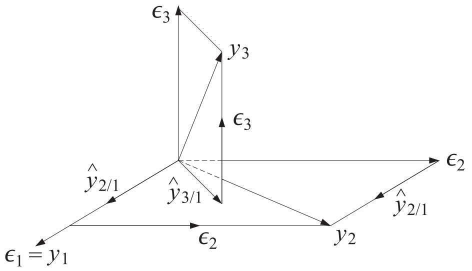
Above, we developed the orthogonal decomposition of a random variable relative to a linear subspace Y which was generated by means of an orthogonal basis $\epsilon_1, \epsilon_2, \dots, \epsilon_M$. In practice, the subspace Y is almost always defined by means of a nonorthogonal basis, such as a collection of random variables

$$Y = \{y_1, y_2, \dots, y_M\}$$

which may be mutually correlated. The subspace Y is defined again as the linear *span* of this basis. The *Gram-Schmidt orthogonalization* process is a recursive procedure of generating an orthogonal basis $\{\epsilon_1, \epsilon_2, \dots, \epsilon_M\}$ from $\{y_1, y_2, \dots, y_M\}$.

The basic idea of the method is this: Initialize the procedure by selecting $\epsilon_1 = y_1$. Next, consider y_2 and decompose it relative to ϵ_1. Then, the component of y_2 which is perpendicular to ϵ_1 is selected as ϵ_2, so that $(\epsilon_1, \epsilon_2) = 0$. Next, take y_3 and decompose it relative to the subspace spanned by $\{\epsilon_1, \epsilon_2\}$ and take the corresponding perpendicular component to be ϵ_3, and so on. For example, the first three steps of the procedure are

$$\begin{aligned}
\epsilon_1 &= y_1 \\
\epsilon_2 &= y_2 - E[y_2\epsilon_1]E[\epsilon_1\epsilon_1]^{-1}\epsilon_1 \\
\epsilon_3 &= y_3 - E[y_3\epsilon_1]E[\epsilon_1\epsilon_1]^{-1}\epsilon_1 - E[y_3\epsilon_2]E[\epsilon_2\epsilon_2]^{-1}\epsilon_2
\end{aligned}$$



At the nth iteration step

$$\epsilon_n = y_n - \sum_{i=1}^{n-1} E[y_n\epsilon_i]E[\epsilon_i\epsilon_i]^{-1}\epsilon_i\,, \quad n = 2, 3, \dots, M \tag{1.6.11}$$

The basis $\{\epsilon_1, \epsilon_2, \dots, \epsilon_M\}$ generated in this way is orthogonal by construction. The Gram-Schmidt process may be understood in terms of the hierarchy of subspaces:

$$\begin{aligned}
Y_1 &= \{\epsilon_1\} = \{y_1\} \\
Y_2 &= \{\epsilon_1, \epsilon_2\} = \{y_1, y_2\} \\
Y_3 &= \{\epsilon_1, \epsilon_2, \epsilon_3\} = \{y_1, y_2, y_3\} \\
&\vdots \\
Y_n &= \{\epsilon_1, \epsilon_2, \dots, \epsilon_n\} = \{y_1, y_2, \dots, y_n\}
\end{aligned}$$

for $n = 1, 2, \dots, M$, where each is a subspace of the next one and differs from the next by the addition of one more basis vector. The second term in Eq. (1.6.11) may be recognized now as the component of y_n parallel to the subspace Y_{n-1}. We may denote this as

$$\hat{y}_{n/n-1} = \sum_{i=1}^{n-1} E[y_n\epsilon_i]E[\epsilon_i\epsilon_i]^{-1}\epsilon_i \tag{1.6.12}$$

Then, Eq. (1.6.11) may be written as

$$\epsilon_n = y_n - \hat{y}_{n/n-1} \quad \text{or} \quad y_n = \hat{y}_{n/n-1} + \epsilon_n \tag{1.6.13}$$

which represents the orthogonal decomposition of y_n relative to the subspace Y_{n-1}. Since, the term $\hat{y}_{n/n-1}$ already lies in Y_{n-1}, we have the direct sum decomposition

$$Y_n = Y_{n-1} \oplus \{y_n\} = Y_{n-1} \oplus \{\epsilon_n\}$$

Introducing the notation

$$b_{ni} = E[y_n\epsilon_i]E[\epsilon_i\epsilon_i]^{-1}\,, \quad 1 \le i \le n-1 \tag{1.6.14}$$

and $b_{nn} = 1$, we may write Eq. (1.6.13) in the form

$$y_n = \sum_{i=1}^{n} b_{ni}\epsilon_i = \epsilon_n + \sum_{i=1}^{n-1} b_{ni}\epsilon_i = \epsilon_n + \hat{y}_{n/n-1} \tag{1.6.15}$$

for $1 \le n \le M$. And in matrix form,

$$\mathbf{y} = B\boldsymbol{\epsilon}\,, \quad \text{where} \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_M \end{bmatrix}, \quad \boldsymbol{\epsilon} = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_M \end{bmatrix} \tag{1.6.16}$$

and B is a *lower-triangular* matrix with matrix elements given by (1.6.14). Its main diagonal is unity. For example, for $M = 4$ we have

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ b_{21} & 1 & 0 & 0 \\ b_{31} & b_{32} & 1 & 0 \\ b_{41} & b_{42} & b_{43} & 1 \end{bmatrix} \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \epsilon_4 \end{bmatrix}$$

Both the matrix B and its inverse B^{-1} are unit lower-triangular matrices. The information contained in the two bases $\mathbf{y}$ and $\boldsymbol{\epsilon}$ is the same. Going from the basis $\mathbf{y}$ to the basis $\boldsymbol{\epsilon}$ removes all the redundant correlations that may exist in $\mathbf{y}$ and “distills” the essential information contained in $\mathbf{y}$ to its most basic form. Because the basis $\boldsymbol{\epsilon}$ is uncorrelated, every basis vector ϵ_i, $i = 1, 2, \dots, M$ will represent something different, or new. Therefore, the random variables ϵ_i are sometimes called the *innovations*, and the representation (1.6.16) of $\mathbf{y}$ in terms of $\boldsymbol{\epsilon}$, the *innovations representation.*

Since the correlation matrix $R_{\epsilon\epsilon} = E[\boldsymbol{\epsilon}\boldsymbol{\epsilon}^T]$ is diagonal, the transformation (1.6.16) corresponds to an LU (lower-upper) *Cholesky factorization* of the correlation matrix of $\mathbf{y}$, that is,

$$R_{yy} = E[\mathbf{y}\mathbf{y}^T] = BE[\boldsymbol{\epsilon}\boldsymbol{\epsilon}^T]B^T = BR_{\epsilon\epsilon}B^T \tag{1.6.17}$$

We note also the invariance of the projected vector $\hat{\mathbf{x}}$ of Eq. (1.6.10) under such linear change of basis:

$$\hat{\mathbf{x}} = E[\mathbf{x}\boldsymbol{\epsilon}^T]E[\boldsymbol{\epsilon}\boldsymbol{\epsilon}^T]^{-1}\boldsymbol{\epsilon} = E[\mathbf{x}\mathbf{y}^T]E[\mathbf{y}\mathbf{y}^T]^{-1}\mathbf{y} \tag{1.6.18}$$

This shows the equivalence of the orthogonal decompositions (1.6.10) to the correlation canceler decompositions (1.4.1). The computational efficiency of the $\boldsymbol{\epsilon}$ basis over the $\mathbf{y}$ basis is evident from the fact that the covariance matrix $E[\boldsymbol{\epsilon}\boldsymbol{\epsilon}^T]$ is diagonal, and

therefore, its inverse is trivially computed. We may also apply the property (1.6.18) to $\mathbf{y}$ itself. Defining the vectors

$$\boldsymbol{\epsilon}_{n-1} = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_{n-1} \end{bmatrix} \qquad \mathbf{y}_{n-1} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{n-1} \end{bmatrix}$$

we may write the projection $\hat{y}_{n/n-1}$ of y_n on the subspace Y_{n-1} given by Eq. (1.6.12) as follows:

$$\hat{y}_{n/n-1} = E[y_n\boldsymbol{\epsilon}_{n-1}^T]E[\boldsymbol{\epsilon}_{n-1}\boldsymbol{\epsilon}_{n-1}^T]^{-1}\boldsymbol{\epsilon}_{n-1} = E[y_n\mathbf{y}_{n-1}^T]E[\mathbf{y}_{n-1}\mathbf{y}_{n-1}^T]^{-1}\mathbf{y}_{n-1} \tag{1.6.19}$$

Eq. (1.6.13) is then written as

$$\epsilon_n = y_n - \hat{y}_{n/n-1} = y_n - E[y_n\mathbf{y}_{n-1}^T]E[\mathbf{y}_{n-1}\mathbf{y}_{n-1}^T]^{-1}\mathbf{y}_{n-1} \tag{1.6.20}$$

which provides a construction of ϵ_n directly in terms of the y_ns. We note that the quantity $\hat{y}_{n/n-1}$ is also the *best linear estimate* of y_n that can be made on the basis of the *previous* y_ns, $Y_{n-1} = \{y_1, y_2, \dots, y_{n-1}\}$. If the index n represents the time index, as it does for random signals, then $\hat{y}_{n/n-1}$ is the best *linear prediction* of y_n on the basis of its past; and ϵ_n is the corresponding prediction error.

The Gram-Schmidt process was started with the first element y_1 of $\mathbf{y}$ and proceeded forward to y_M. The process can just as well be started with y_M and proceed backward to y_1 (see Problem 1.15). It may be interpreted as *backward prediction*, or postdiction, and leads to the UL (rather than LU) factorization of the covariance matrix R_{yy}. In Sec. 1.8, we study the properties of such forward and backward orthogonalization procedures in some detail.

Example 1.6.1: Consider the three zero-mean random variables $\{y_1, y_2, y_3\}$ and let $R_{ij} = E[y_iy_j]$ for $i, j = 1, 2, 3$, denote their correlation matrix. Then, the explicit construction indicated in Eq. (1.6.20) can be carried out as follows. The required vectors $\mathbf{y}_{n-1}$ are:

$$\mathbf{y}_1 = [y_1]\,, \quad \mathbf{y}_2 = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

and hence

$$\begin{aligned}
E[y_2\mathbf{y}_1^T] &= E[y_2y_1] = R_{21} \\
E[\mathbf{y}_1\mathbf{y}_1^T] &= E[y_1y_1] = R_{11} \\
E[y_3\mathbf{y}_2^T] &= E[y_3[y_1, y_2]] = [R_{31}, R_{32}] \\
E[\mathbf{y}_2\mathbf{y}_2^T] &= E\left[\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}[y_1, y_2]\right] = \begin{bmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \end{bmatrix}
\end{aligned}$$

Therefore, Eq. (1.6.20) becomes

$$\begin{aligned}
\epsilon_1 &= y_1 \\
\epsilon_2 &= y_2 - \hat{y}_{2/1} = y_2 - R_{21}R_{11}^{-1}y_1 \\
\epsilon_3 &= y_3 - \hat{y}_{3/2} = y_3 - [R_{31}, R_{32}]\begin{bmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \end{bmatrix}^{-1}\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}
\end{aligned}$$

Example 1.6.2: The zero-mean random vector $\mathbf{y} = [y_1, y_2, y_3]^T$ has covariance matrix

$$R_{yy} = \begin{bmatrix} 1 & -1 & 1 \\ -1 & 3 & 3 \\ 1 & 3 & 12 \end{bmatrix}$$

Determine the innovations representation of $\mathbf{y}$ in two ways: using the Gram- Schmidt construction and using the results of Example 1.6.1.

Solution: Starting with $\epsilon_1 = y_1$, we find $E[y_2\epsilon_1] = R_{21} = -1$ and $E[\epsilon_1^2] = R_{11} = 1$. Therefore,

$$\epsilon_2 = y_2 - E[y_2\epsilon_1]E[\epsilon_1^2]^{-1}\epsilon_1 = y_2 + \epsilon_1 = y_2 + y_1$$

with a mean-square value $E[\epsilon_2^2] = E[y_2^2] + 2E[y_2y_1] + E[y_1^2] = 3 - 2 + 1 = 2$. Similarly, we find $E[y_3\epsilon_1] = R_{31} = 1$ and

$$E[y_3\epsilon_2] = E\big[y_3(y_2 + y_1)\big] = R_{32} + R_{31} = 3 + 1 = 4$$

Thus,

$$\epsilon_3 = y_3 - E[y_3\epsilon_1]E[\epsilon_1\epsilon_1]^{-1}\epsilon_1 - E[y_3\epsilon_2]E[\epsilon_2\epsilon_2]^{-1}\epsilon_2 = y_3 - \epsilon_1 - 2\epsilon_2$$

or,

$$\epsilon_3 = y_3 - y_1 - 2(y_2 + y_1) = y_3 - 2y_2 - 3y_1$$

Solving for the ys and writing the answer in matrix form we have

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 1 & 2 & 1 \end{bmatrix} \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \end{bmatrix} = B\boldsymbol{\epsilon}$$

The last row determines $E[\epsilon_3^2]$. Using the mutual orthogonality of the ϵ_is, we have

$$E[y_3^2] = E\big[(\epsilon_3 + 2\epsilon_2 + \epsilon_1)^2\big] = E[\epsilon_3^2] + 4E[\epsilon_2^2] + E[\epsilon_1^2] \quad \Rightarrow \quad 12 = E[\epsilon_3^2] + 8 + 1$$

which gives $E[\epsilon_3^2] = 3$. Using the results of Example 1.6.1, we have

$$\epsilon_3 = y_3 - [R_{31}, R_{32}]\begin{bmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \end{bmatrix}^{-1} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = y_3 - [1, 3]\begin{bmatrix} 1 & -1 \\ -1 & 3 \end{bmatrix}^{-1} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

The indicated matrix operations are computed easily and lead to the same expression for ϵ_3 found above. □

The innovations representation Eq. (1.6.16) and the Cholesky factorization (1.6.17) are also very useful for the purpose of simulating a random vector having a prescribed covariance matrix. The procedure is as follows: given $R = E[\mathbf{y}\mathbf{y}^T]$, find its Cholesky factor B and the diagonal matrix $R_{\epsilon\epsilon}$; then, using any standard random number generator, generate M independent random numbers $\boldsymbol{\epsilon} = [\epsilon_1, \epsilon_2, \dots, \epsilon_M]^T$ of mean zero and variances equal to the diagonal entries of $R_{\epsilon\epsilon}$, and perform the matrix operation $\mathbf{y} = B\boldsymbol{\epsilon}$ to obtain a realization of the random vector $\mathbf{y}$.

Conversely, if a number of independent realizations of $\mathbf{y}$ are available, $\{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_N\}$, we may form an estimate of the covariance matrix by the following expression, referred to as the *sample covariance matrix*

$$\hat{R} = \frac{1}{N}\sum_{n=1}^{N} \mathbf{y}_n\mathbf{y}_n^T \tag{1.6.21}$$

Example 1.6.3: In typical array processing applications, a linear array of, say, M equally spaced sensors measures the incident radiation field. This field may consist of a number of plane waves incident from different angles on the array plus background noise. The objective is to determine the number, angles of arrival, and strengths of the incident plane waves from measurements of the field at the sensor elements. At each time instant, the measurements at the M sensors may be assembled into the M-dimensional random vector $\mathbf{y}$, called an instantaneous *snapshot*. Thus, the correlation matrix $R = E[\mathbf{y}\mathbf{y}^T]$ measures the correlations that exist among sensors, that is, *spatial correlations*. In Chap. 14, we will consider methods of extracting the angle-of-arrival information from the covariance matrix R. Most of these methods require an estimate of the covariance matrix, which is typically given by Eq. (1.6.21) on the basis of N snapshots. □

How good an estimate of R is $\hat{R}$? First, note that it is an *unbiased* estimate:

$$E[\hat{R}] = \frac{1}{N}\sum_{n=1}^{N} E[\mathbf{y}_n\mathbf{y}_n^T] = \frac{1}{N}(NR) = R$$

Second, we show that it is *consistent*. The correlation between the various matrix elements of $\hat{R}$ is obtained as follows:

$$E[\hat{R}_{ij}\hat{R}_{kl}] = \frac{1}{N^2}\sum_{n=1}^{N}\sum_{m=1}^{N} E[y_{ni}y_{nj}y_{mk}y_{ml}]$$

where y_{ni} is the ith component of the nth vector $\mathbf{y}_n$. To get a simple expression for the covariance of $\hat{R}$, we will assume that $\mathbf{y}_n$, $n = 1, 2, \dots, N$ are independent zero-mean gaussian random vectors of covariance matrix R. This implies that [4,5]

$$E[y_{ni}y_{nj}y_{mk}y_{ml}] = R_{ij}R_{kl} + \delta_{nm}(R_{ik}R_{jl} + R_{il}R_{jk})$$

It follows that

$$E[\hat{R}_{ij}\hat{R}_{kl}] = R_{ij}R_{jk} + \frac{1}{N}(R_{ik}R_{jl} + R_{il}R_{jk}) \tag{1.6.22}$$

Writing $\Delta R = \hat{R} - E[\hat{R}] = \hat{R} - R$, we obtain for the covariance

$$E[\Delta R_{ij}\Delta R_{kl}] = \frac{1}{N}(R_{ik}R_{jl} + R_{il}R_{jk}) \tag{1.6.23}$$

Thus, $\hat{R}$ is a consistent estimator. The result of Eq. (1.6.23) is typical of the asymptotic results that are available in the statistical literature [4,5]. It will be used in Chap. 14 to obtain asymptotic results for linear prediction parameters and for the eigenstructure methods of spectrum estimation.

The sample covariance matrix (1.6.21) may also be written in an *adaptive*, or recursive form,

$$\hat{R}_N = \frac{1}{N}\sum_{n=1}^{N}\mathbf{y}_n\mathbf{y}_n^T = \frac{1}{N}\left[\sum_{n=1}^{N-1}\mathbf{y}_n\mathbf{y}_n^T + \mathbf{y}_N\mathbf{y}_N^T\right] = \frac{1}{N}\big[(N-1)\hat{R}_{N-1} + \mathbf{y}_N\mathbf{y}_N^T\big]$$

where we wrote $\hat{R}_N$ to explicitly indicate the dependence on N. A more intuitive way of writing this recursion is in the “predictor/corrector” form

$$\hat{R}_N = \hat{R}_{N-1} + \frac{1}{N}(\mathbf{y}_N\mathbf{y}_N^T - \hat{R}_{N-1}) \tag{1.6.24}$$

The term $\hat{R}_{N-1}$ may be thought of as a prediction of R based on $N-1$ observations, the Nth observation $\mathbf{y}_N\mathbf{y}_N^T$ may be thought of as an instantaneous estimate of R, and the term in the parenthesis as the prediction error that is used to correct the prediction. The function **sampcov** takes as input the old matrix $\hat{R}_{N-1}$, and the new observation $\mathbf{y}_N$, and outputs the updated matrix $\hat{R}_N$, overwriting the old one.

Example 1.6.4: Consider the 3×3 random vector $\mathbf{y}$ defined in Example 1.6.2. Using the innovations representation of $\mathbf{y}$, generate $N = 200$ independent vectors $\mathbf{y}_n$, $n = 1, 2, \dots, N$ and then compute the estimated sample covariance matrix (1.6.21) and compare it with the theoretical R. Compute the sample covariance matrix $\hat{R}$ recursively and plot its matrix elements as functions of the iteration number N.

Solution: Generate N independent 3-vectors $\boldsymbol{\epsilon}_n$, and compute $\mathbf{y}_n = B\boldsymbol{\epsilon}_n$. The estimated and theoretical covariance matrices are

$$\hat{R} = \begin{bmatrix} 0.995 & -1.090 & 0.880 \\ -1.090 & 3.102 & 2.858 \\ 0.880 & 2.858 & 11.457 \end{bmatrix}, \quad R = \begin{bmatrix} 1 & -1 & 1 \\ -1 & 3 & 3 \\ 1 & 3 & 12 \end{bmatrix}$$

Can we claim that this is a good estimate of R? Yes, because the deviations from R are consistent with the expected deviations given by Eq. (1.6.23). The standard deviation of the ijth matrix element is

$$\delta R_{ij} = \sqrt{E\big[(\Delta R_{ij})^2\big]} = \sqrt{(R_{ii}R_{jj} + R_{ij}^2)/N}$$

The estimated values $\hat{R}_{ij}$ fall within the intervals $R_{ij} - \delta R_{ij} \le \hat{R}_{ij} \le R_{ij} + \delta R_{ij}$, as can be verified by inspecting the matrices

$$R - \delta R = \begin{bmatrix} 0.901 & -1.146 & 0.754 \\ -1.146 & 2.691 & 2.534 \\ 0.754 & 2.534 & 10.857 \end{bmatrix}, \quad R + \delta R = \begin{bmatrix} 1.099 & -0.854 & 1.246 \\ -0.854 & 3.309 & 3.466 \\ 1.246 & 3.466 & 13.143 \end{bmatrix}$$

The recursive computation Eq. (1.6.24), implemented by successive calls to the function **sampcov**, is shown in Fig. 1.6.2, where only the matrix elements R_{11}, R_{12}, and R_{22} are plotted versus N. Such graphs give us a better idea of how fast the sample estimate $\hat{R}_N$ converges to the theoretical R. □

1.7 Partial Correlations

A concept intimately connected to the Gram-Schmidt orthogonalization is that of the partial correlation. It plays a central role in linear prediction applications.

Consider the Gram-Schmidt orthogonalization of a random vector $\mathbf{y}$ in the form $\mathbf{y} = B\boldsymbol{\epsilon}$, where B is a unit lower-triangular matrix, and $\boldsymbol{\epsilon}$ is a vector of mutually uncorrelated components. Inverting, we have

$$\boldsymbol{\epsilon} = A\mathbf{y} \tag{1.7.1}$$

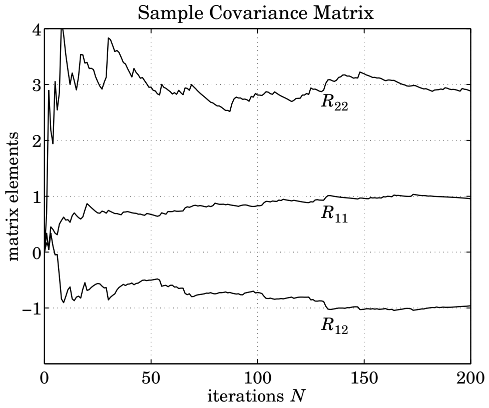


Fig. 1.6.2 Recursive computation of the sample covariance matrix.

where $A = B^{-1}$. Now, suppose the vector $\mathbf{y}$ is arbitrarily subdivided into three subvectors as follows:

$$\mathbf{y} = \begin{bmatrix} \mathbf{y}_0 \\ \mathbf{y}_1 \\ \mathbf{y}_2 \end{bmatrix}$$

where $\mathbf{y}_0, \mathbf{y}_1, \mathbf{y}_2$ do not necessarily have the same dimension. Then, the matrix equation (1.7.1) may also be decomposed in a block-compatible form:

$$\begin{bmatrix} \boldsymbol{\epsilon}_0 \\ \boldsymbol{\epsilon}_1 \\ \boldsymbol{\epsilon}_2 \end{bmatrix} = \begin{bmatrix} A_{00} & 0 & 0 \\ A_{11} & A_{10} & 0 \\ A_{22} & A_{21} & A_{20} \end{bmatrix} \begin{bmatrix} \mathbf{y}_0 \\ \mathbf{y}_1 \\ \mathbf{y}_2 \end{bmatrix} \tag{1.7.2}$$

where A_{00}, A_{10}, A_{20} are unit lower-triangular matrices. Since $\mathbf{y}$ has components that are generally correlated with each other, it follows that $\mathbf{y}_0$ will be correlated with $\mathbf{y}_1$, and $\mathbf{y}_1$ will be correlated with $\mathbf{y}_2$. Thus, through the intermediate action of $\mathbf{y}_1$, the vector $\mathbf{y}_0$ will be indirectly coupled with the vector $\mathbf{y}_2$. The question we would like to ask is this: Suppose the effect of the intermediate vector $\mathbf{y}_1$ were to be removed, then what would be the correlation that is left between $\mathbf{y}_0$ and $\mathbf{y}_2$? This is the *partial correlation*. It represents the "true" or "direct" influence of $\mathbf{y}_0$ on $\mathbf{y}_2$, when the indirect influence via $\mathbf{y}_1$ is removed. To remove the effect of $\mathbf{y}_1$, we project both $\mathbf{y}_0$ and $\mathbf{y}_2$ on the subspace spanned by $\mathbf{y}_1$ and then subtract these parts from both, that is, let

$$\mathbf{e}_0 = \mathbf{y}_0 - (\text{projection of } \mathbf{y}_0 \text{ on } \mathbf{y}_1)$$

$$\mathbf{e}_2 = \mathbf{y}_2 - (\text{projection of } \mathbf{y}_2 \text{ on } \mathbf{y}_1)$$

or,

$$\begin{aligned} \mathbf{e}_0 &= \mathbf{y}_0 - R_{01}R_{11}^{-1}\mathbf{y}_1 \\ \mathbf{e}_2 &= \mathbf{y}_2 - R_{21}R_{11}^{-1}\mathbf{y}_1 \end{aligned} \tag{1.7.3}$$

where we defined $R_{ij} = E[\mathbf{y}_i\mathbf{y}_j^T]$, for $i,j = 0,1,2$. We define the *partial correlation* (PARCOR) *coefficient* between $\mathbf{y}_0$ and $\mathbf{y}_2$, with the effect of the intermediate $\mathbf{y}_1$ removed, as follows:

$$\Gamma = E[\mathbf{e}_2\mathbf{e}_0^T]E[\mathbf{e}_0\mathbf{e}_0^T]^{-1} \tag{1.7.4}$$

Then, Γ may be expressed in terms of the entries of the matrix A as follows:

$$\Gamma = -A_{20}^{-1}A_{22} \tag{1.7.5}$$

To prove this result, we consider the last equation of (1.7.2):

$$\boldsymbol{\epsilon}_2 = A_{22}\mathbf{y}_0 + A_{21}\mathbf{y}_1 + A_{20}\mathbf{y}_2 \tag{1.7.6}$$

By construction, $\boldsymbol{\epsilon}_2$ is orthogonal to $\mathbf{y}_1$, so that $E[\boldsymbol{\epsilon}_2\mathbf{y}_1^T]= 0$. Thus we obtain the relationship:

$$\begin{aligned} E[\boldsymbol{\epsilon}_2\mathbf{y}_1^T] &= A_{22}E[\mathbf{y}_0\mathbf{y}_1^T]+A_{21}E[\mathbf{y}_1\mathbf{y}_1^T]+A_{20}E[\mathbf{y}_2\mathbf{y}_1^T] \\ &= A_{22}R_{01} + A_{21}R_{11} + A_{20}R_{21} = 0 \end{aligned} \tag{1.7.7}$$

Using Eqs. (1.7.3) and (1.7.7), we may express $\boldsymbol{\epsilon}_2$ in terms of $\mathbf{e}_0$ and $\mathbf{e}_2$, as follows:

$$\begin{aligned} \boldsymbol{\epsilon}_2 &= A_{22}(\mathbf{e}_0 + R_{01}R_{11}^{-1}\mathbf{y}_1)+A_{21}\mathbf{y}_1 + A_{20}(\mathbf{e}_2 + R_{21}R_{11}^{-1}\mathbf{y}_1) \\ &= A_{22}\mathbf{e}_0 + A_{20}\mathbf{e}_2 + (A_{22}R_{01} + A_{21}R_{11} + A_{20}R_{21})R_{11}^{-1}\mathbf{y}_1 \\ &= A_{22}\mathbf{e}_0 + A_{20}\mathbf{e}_2 \end{aligned} \tag{1.7.8}$$

Now, by construction, $\boldsymbol{\epsilon}_2$ is orthogonal to both $\mathbf{y}_0$ and $\mathbf{y}_1$, and hence also to $\mathbf{e}_0$, that is, $E[\boldsymbol{\epsilon}_2\mathbf{e}_0^T]= 0$. Using Eq. (1.7.8) we obtain

$$E[\boldsymbol{\epsilon}_2\mathbf{e}_0^T]= A_{22}E[\mathbf{e}_0\mathbf{e}_0^T]+A_{20}E[\mathbf{e}_2\mathbf{e}_0^T]= 0$$

from which (1.7.5) follows. It is interesting also to note that (1.7.8) may be written as

$$\boldsymbol{\epsilon}_2 = A_{20}\mathbf{e}$$

where $\mathbf{e} = \mathbf{e}_2 - \Gamma\mathbf{e}_0$ is the orthogonal complement of $\mathbf{e}_2$ relative to $\mathbf{e}_0$.

Example 1.7.1: An important special case of Eq. (1.7.5) is when $\mathbf{y}_0$ and $\mathbf{y}_2$ are selected as the first and last components of $\mathbf{y}$, and therefore $\mathbf{y}_1$ consists of all the intermediate components. For example, suppose $\mathbf{y} = [y_0, y_1, y_2, y_3, y_4]^T$. Then, the decomposition (1.7.2) can be written as follows:

$$\left[\begin{array}{c} \epsilon_0 \\ \hline \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \hline \epsilon_4 \end{array}\right] = \left[\begin{array}{c|ccc|c} 1 & 0 & 0 & 0 & 0 \\ \hline a_{11} & 1 & 0 & 0 & 0 \\ a_{22} & a_{21} & 1 & 0 & 0 \\ a_{33} & a_{32} & a_{31} & 1 & 0 \\ \hline a_{44} & a_{43} & a_{42} & a_{41} & 1 \end{array}\right] \left[\begin{array}{c} y_0 \\ \hline y_1 \\ y_2 \\ y_3 \\ \hline y_4 \end{array}\right] \tag{1.7.9}$$

where $\mathbf{y}_0, \mathbf{y}_1, \mathbf{y}_2$ are chosen as the vectors

$$\mathbf{y}_0 = [y_0]\,, \quad \mathbf{y}_1 = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}, \quad \mathbf{y}_2 = [y_4]$$

The matrices A_{20} and A_{22} are in this case the scalars $A_{20} = [1]$ and $A_{22} = [a_{44}]$. Therefore, the corresponding PARCOR coefficient (1.7.5) is

$$\Gamma = -a_{44}$$

Clearly, the first column $[1, a_{11}, a_{22}, a_{33}, a_{44}]$ of A contains all the lower order PARCOR coefficients, that is, the quantity

$$\gamma_p = -a_{pp}\,, \quad p = 1,2,3,4$$

represents the partial correlation coefficient between y_0 and y_p, with the effect of all the intermediate variables $y_1, y_2, \dots, y_{p-1}$ removed. □

We note the backward indexing of the entries of the matrix A in Eqs. (1.7.2) and (1.7.9). It corresponds to writing ϵ_n in a convolutional form

$$\epsilon_n = \sum_{i=0}^{n} a_{ni}y_{n-i} = \sum_{i=0}^{n} a_{n,n-i}y_i = y_n + a_{n1}y_{n-1} + a_{n2}y_{n-2} + \cdots + a_{nn}y_0 \tag{1.7.10}$$

and conforms to standard notation in linear prediction applications. Comparing (1.7.10) with (1.6.13), we note that the projection of y_n onto the subspace Y_{n-1} may also be expressed directly in terms of the correlated basis $Y_{n-1} = \{y_0, y_1, \dots, y_{n-1}\}$ as follows:

$$\hat{y}_{n/n-1} = -[a_{n1}y_{n-1} + a_{n2}y_{n-2} + \cdots + a_{nn}y_0] \tag{1.7.11}$$

An alternative expression was given in Eq. (1.6.19). Writing Eq. (1.7.10) in vector form, we have

$$\epsilon_n = [a_{nn}, \dots, a_{n1}, 1]\begin{bmatrix} y_0 \\ \vdots \\ y_{n-1} \\ y_n \end{bmatrix} = [1, a_{n1}, \dots, a_{nn}]\begin{bmatrix} y_n \\ y_{n-1} \\ \vdots \\ y_0 \end{bmatrix} \tag{1.7.12}$$

Thus, there are two possible definitions for the data vector $\mathbf{y}$ and corresponding weight vector $\mathbf{a}$. According to the first definition—which is what we used in Eqs. (1.7.1) and (1.7.9)—the vector $\mathbf{y}$ is indexed from the lowest to the highest index and the vector $\mathbf{a}$ is indexed in the reverse way. According to the second definition, $\mathbf{y}$ and $\mathbf{a}$ are exactly the *reverse*, or upside-down, versions of the first definition, namely, $\mathbf{y}$ is indexed backward from high to low, whereas $\mathbf{a}$ is indexed forward. If we use the second definition and write Eq. (1.7.12) in matrix form, we obtain the reverse of Eq. (1.7.9), that is

$$\boldsymbol{\epsilon}_{\rm rev} = \begin{bmatrix} \epsilon_4 \\ \epsilon_3 \\ \epsilon_2 \\ \epsilon_1 \\ \epsilon_0 \end{bmatrix} = \begin{bmatrix} 1 & a_{41} & a_{42} & a_{43} & a_{44} \\ 0 & 1 & a_{31} & a_{32} & a_{33} \\ 0 & 0 & 1 & a_{21} & a_{22} \\ 0 & 0 & 0 & 1 & a_{11} \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} y_4 \\ y_3 \\ y_2 \\ y_1 \\ y_0 \end{bmatrix} = U\mathbf{y}_{\rm rev} \tag{1.7.13}$$

Thus, the transformation between the correlated and decorrelated bases is now by means of a unit *upper-triangular* matrix U. It corresponds to the UL (rather than LU) factorization of the covariance matrix of the reversed vector $\mathbf{y}_{\rm rev}$. Writing $R_{\rm rev} = E[\mathbf{y}_{\rm rev}\mathbf{y}_{\rm rev}^T]$ and $D_{\rm rev} = E[\boldsymbol{\epsilon}_{\rm rev}\boldsymbol{\epsilon}_{\rm rev}^T]$, it follows from Eq. (1.7.13) that

$$D_{\rm rev} = UR_{\rm rev}U^T \tag{1.7.14}$$

The precise connection between the original basis and its reverse, and between their respective Cholesky factorizations, can be seen as follows. The operation of reversing a vector is equivalent to a linear transformation by the so-called *reversing* matrix J, consisting of ones along its antidiagonal and zeros everywhere else; for example, in the 5×5 case of Example 1.7.1,

$$J = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The reversed vectors will be $\mathbf{y}_{\rm rev} = J\mathbf{y}$ and $\boldsymbol{\epsilon}_{\rm rev} = J\boldsymbol{\epsilon}$. Using the property $J = J^T$, it follows that $R_{\rm rev} = JR_{yy}J$ and $D_{\rm rev} = JR_{\epsilon\epsilon}J$. Comparing Eq. (1.7.9) and Eq. (1.7.13) and using the property $J^2 = I$, we find,

$$\boldsymbol{\epsilon}_{\rm rev} = J\boldsymbol{\epsilon} = JA\mathbf{y} = (JAJ)(J\mathbf{y}) = (JAJ)\mathbf{y}_{\rm rev}\,, \quad \text{or,}$$

$$U = JAJ \tag{1.7.15}$$

Note that J acting on a matrix from the left reverses each column, whereas acting from the right, it reverses each row. Thus, U is obtained from A by reversing all its columns and then all its rows. Regardless of the choice of the vector $\mathbf{y}$, the Gram-Schmidt construction proceeds from the lowest to the highest index of $\mathbf{y}$, and therefore, it can be interpreted as predicting the present from the past. But whether this process leads to LU or UL factorization depends on whether $\mathbf{y}$ or its reverse is used as the basis. Of course, the choice of basis does not affect the computation of linear estimates. As we saw in Eq. (1.6.18), linear estimates are invariant under any linear change of basis; in particular,

$$\hat{x} = E[x\mathbf{y}^T]E[\mathbf{y}\mathbf{y}^T]^{-1}\mathbf{y} = E[x\mathbf{y}_{\rm rev}^T]E[\mathbf{y}_{\rm rev}\mathbf{y}_{\rm rev}^T]^{-1}\mathbf{y}_{\rm rev}$$

In this book, we use both representations $\mathbf{y}$ and $\mathbf{y}_{\rm rev}$, whichever is the most convenient depending on the context and application. For example, in discussing the classical Wiener filtering problem and Kalman filtering in Chap. 11, we find the basis $\mathbf{y}$ more natural. On the other hand, the basis $\mathbf{y}_{\rm rev}$ is more appropriate for discussing the lattice and direct-form realizations of FIR Wiener filters.

The ideas discussed in the last three sections are basic in the development of optimum signal processing algorithms, and will be pursued further in subsequent chapters. However, taking a brief look ahead, we point out how some of these concepts fit into the signal processing context:

1. The correlation canceling/orthogonal decompositions of Eqs. (1.4.1) and (1.6.10) for the basis of optimum Wiener and Kalman filtering.
2. The Gram-Schmidt process expressed by Eqs. (1.6.13) and (1.6.20) forms the basis of linear prediction and is also used in the development of the Kalman filter.
3. The representation $\mathbf{y} = B\boldsymbol{\epsilon}$ may be thought of as a signal model for synthesizing $\mathbf{y}$ by processing the uncorrelated (white noise) vector $\boldsymbol{\epsilon}$ through the linear filter B. The lower-triangular nature of B is equivalent to causality. Such signal models have a very broad range of applications, among which are speech synthesis and modern methods of spectrum estimation.
4. The inverse representation $\boldsymbol{\epsilon} = A\mathbf{y}$ of Eqs. (1.7.1) and (1.7.10) corresponds to the analysis filters of linear prediction. The PARCOR coefficients will turn out to be the reflection coefficients of the lattice filter realizations of linear prediction.
5. The Cholesky factorization (1.6.17) is the matrix analog of the spectral factorization theorem. It not only facilitates the solution of optimum Wiener filtering problems, but also the making of signal models of the type of Eq. (1.6.16).

1.8 Forward/Backward Prediction and LU/UL Factorization

The Gram-Schmidt orthogonalization procedure discussed in the previous sections was a *forward* procedure in the sense that the successive orthogonalization of the components of a random vector $\mathbf{y}$ proceeded forward from the first component to the last. It was given a linear prediction interpretation, that is, at each orthogonalization step, a prediction of the present component of $\mathbf{y}$ is made in terms of all the past ones. The procedure was seen to be mathematically equivalent to the LU Cholesky factorization of the covariance matrix $R = E[\mathbf{y}\mathbf{y}^T]$ (or, the UL factorization with respect to the reversed basis). We remarked in Sec. 1.6 (see also Problem 1.15) that if the Gram-Schmidt construction is started at the other end of the random vector $\mathbf{y}$ then the UL factorization of R is obtained (equivalently, the LU factorization in the reversed basis).

In this section, we discuss in detail such forward and backward Gram-Schmidt constructions and their relationship to *forward* and *backward* linear prediction and to LU and UL Cholesky factorizations, and show how to realize linear estimators in the forward and backward orthogonal bases.

Our main objective is to gain further insight into the properties of the basis of observations $\mathbf{y}$ and to provide a preliminary introduction to a large number of concepts and methods that have become standard tools in modern signal processing practice, namely, Levinson's and Schur's algorithms; fast Cholesky factorizations; lattice filters for linear prediction; lattice realizations of FIR Wiener filters; and fast recursive least squares adaptive algorithms. Although these concepts are fully developed in Chapters 12 and 16, we would like to show in this preliminary discussion how far one can go toward these goals *without* making any assumptions about any structural properties of the covariance matrix R, such as Toeplitz and stationarity properties, or the so-called *shift-invariance* property of adaptive least squares problems.

Forward/Backward Normal Equations

Let $\mathbf{y} = [y_a, \dots, y_b]^T$ be a random vector whose first and last components are y_a and y_b. Let $\hat{y}_b$ be the best linear estimate of y_b based on the *rest* of the vector $\mathbf{y}$, that is,

$$\hat{y}_b = E[y_b\bar{\mathbf{y}}^T]E[\bar{\mathbf{y}}\bar{\mathbf{y}}^T]^{-1}\bar{\mathbf{y}} \tag{1.8.1}$$

where $\bar{\mathbf{y}}$ is the upper part of $\mathbf{y}$, namely,

$$\mathbf{y} = \begin{bmatrix} y_a \\ \vdots \\ y_b \end{bmatrix} = \begin{bmatrix} \bar{\mathbf{y}} \\ y_b \end{bmatrix} \tag{1.8.2}$$

Similarly, let $\hat{y}_a$ be the best estimate of y_a based on the rest of $\mathbf{y}$, namely,

$$\hat{y}_a = E[y_a\tilde{\mathbf{y}}^T]E[\tilde{\mathbf{y}}\tilde{\mathbf{y}}^T]^{-1}\tilde{\mathbf{y}} \tag{1.8.3}$$

where $\tilde{\mathbf{y}}$ is the lower part of $\mathbf{y}$, that is,

$$\mathbf{y} = \begin{bmatrix} y_a \\ \vdots \\ y_b \end{bmatrix} = \begin{bmatrix} y_a \\ \tilde{\mathbf{y}} \end{bmatrix} \tag{1.8.4}$$

The decompositions (1.8.2) and (1.8.4) imply analogous decompositions of the covariance matrix $R = E[\mathbf{y}\mathbf{y}^T]$ as follows

$$R = \begin{bmatrix} \bar{R} & \mathbf{r}_b \\ \mathbf{r}_b^T & \rho_b \end{bmatrix} = \begin{bmatrix} \rho_a & \mathbf{r}_a^T \\ \mathbf{r}_a & \tilde{R} \end{bmatrix} \tag{1.8.5}$$

where

$$\begin{aligned} \tilde{R} &= E[\tilde{\mathbf{y}}\tilde{\mathbf{y}}^T]\,, \quad \mathbf{r}_a = E[y_a\tilde{\mathbf{y}}]\,, \quad \rho_a = E[y_a^2] \\ \bar{R} &= E[\bar{\mathbf{y}}\bar{\mathbf{y}}^T]\,, \quad \mathbf{r}_b = E[y_b\bar{\mathbf{y}}]\,, \quad \rho_b = E[y_b^2] \end{aligned} \tag{1.8.6}$$

We will refer to $\hat{y}_a$ and $\hat{y}_b$ as the forward and backward predictors, respectively. Since we have not yet introduced any notion of time in our discussion of random vectors, we will employ the terms forward and backward as convenient ways of referring to the above two estimates. In the present section, the basis $\mathbf{y}$ will be chosen according to the reversed-basis convention. As discussed in Sec. 1.7, LU becomes UL factorization in the reversed basis. By the same token, UL becomes LU factorization. Therefore, the term forward will be associated with UL and the term backward with LU factorization. The motivation for the choice of basis arises from the time series case, where the consistent usage of these two terms requires that $\mathbf{y}$ be reverse-indexed from high to low indices. For example, a typical choice of $\mathbf{y}$, relevant in the context of Mth order FIR Wiener filtering problems, is

$$\mathbf{y} = \begin{bmatrix} y_n \\ y_{n-1} \\ \vdots \\ y_{n-M} \end{bmatrix}$$

where n represents the time index. Therefore, estimating the first element, y_n, from the rest of $\mathbf{y}$ will be equivalent to prediction, and estimating the last element, y_{n-M}, from the rest of $\mathbf{y}$ will be equivalent to postdiction. Next, we introduce the forward and backward prediction coefficients by

$$\mathbf{a} = \begin{bmatrix} 1 \\ \boldsymbol{\alpha} \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} \boldsymbol{\beta} \\ 1 \end{bmatrix}, \quad \text{where} \quad \boldsymbol{\alpha} = -\tilde{R}^{-1}\mathbf{r}_a\,, \quad \boldsymbol{\beta} = -\bar{R}^{-1}\mathbf{r}_b \tag{1.8.7}$$

In this notation, the predictors (1.8.1) and (1.8.3) are written as

$$\hat{y}_a = -\boldsymbol{\alpha}^T\tilde{\mathbf{y}}\,, \quad \hat{y}_b = -\boldsymbol{\beta}^T\bar{\mathbf{y}} \tag{1.8.8}$$

The corresponding prediction errors are

$$e_a = y_a - \hat{y}_a = y_a + \boldsymbol{\alpha}^T\tilde{\mathbf{y}} = \mathbf{a}^T\mathbf{y}\,, \quad e_b = y_b - \hat{y}_b = y_b + \boldsymbol{\beta}^T\bar{\mathbf{y}} = \mathbf{b}^T\mathbf{y} \tag{1.8.9}$$

with mean square values

$$\begin{aligned} E_a &= E[e_a^2] = E\big[(\mathbf{a}^T\mathbf{y})(\mathbf{y}^T\mathbf{a})\big] = \mathbf{a}^T R\mathbf{a} \\ E_b &= E[e_b^2] = E\big[(\mathbf{b}^T\mathbf{y})(\mathbf{y}^T\mathbf{b})\big] = \mathbf{b}^T R\mathbf{b} \end{aligned} \tag{1.8.10}$$

Because the estimation errors are orthogonal to the observations that make up the estimates, that is, $E[e_b\bar{\mathbf{y}}] = 0$ and $E[e_a\tilde{\mathbf{y}}] = 0$, it follows that $E[\hat{y}_a e_a] = 0$ and $E[\hat{y}_b e_b] = 0$. Therefore, we can write $E[e_a^2] = E[y_a e_a]$ and $E[e_b^2] = E[y_b e_b]$. Thus, the minimized values of the prediction errors (1.8.10) can be written as

$$\begin{aligned} E_a &= E[y_a e_a] = E\big[y_a(y_a + \boldsymbol{\alpha}^T\tilde{\mathbf{y}})\big] = \rho_a + \boldsymbol{\alpha}^T\mathbf{r}_a = \rho_a - \mathbf{r}_a^T\tilde{R}^{-1}\mathbf{r}_a \\ E_b &= E[y_b e_b] = E\big[y_b(y_b + \boldsymbol{\beta}^T\bar{\mathbf{y}})\big] = \rho_b + \boldsymbol{\beta}^T\mathbf{r}_b = \rho_b - \mathbf{r}_b^T\bar{R}^{-1}\mathbf{r}_b \end{aligned} \tag{1.8.11}$$

By construction, the mean square estimation errors are positive quantities. This also follows from the positivity of the covariance matrix R. With respect to the block decompositions (1.8.5), it is easily shown that a necessary and sufficient condition for R to be positive definite is that $\bar{R}$ be positive definite and $\rho_b - \mathbf{r}_b^T\bar{R}^{-1}\mathbf{r}_b > 0$; alternatively, that $\tilde{R}$ be positive definite and $\rho_a - \mathbf{r}_a^T\tilde{R}^{-1}\mathbf{r}_a > 0$.

Equations (1.8.7) and (1.8.11) may be combined now into the more compact forms, referred to as the forward and backward *normal equations* of linear prediction,

$$R\mathbf{a} = E_a\mathbf{u}\,, \quad R\mathbf{b} = E_b\mathbf{v}\,, \qquad \text{where} \quad \mathbf{u} = \begin{bmatrix} 1 \\ \mathbf{0} \end{bmatrix}, \quad \mathbf{v} = \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix} \tag{1.8.12}$$

For example,

$$R\mathbf{b} = \begin{bmatrix} \bar{R} & \mathbf{r}_b \\ \mathbf{r}_b^T & \rho_b \end{bmatrix}\begin{bmatrix} \boldsymbol{\beta} \\ 1 \end{bmatrix} = \begin{bmatrix} \bar{R}\boldsymbol{\beta} + \mathbf{r}_b \\ \mathbf{r}_b^T\boldsymbol{\beta} + \rho_b \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ E_b \end{bmatrix} = E_b\mathbf{v}$$

and similarly,

$$R\mathbf{a} = \begin{bmatrix} \rho_a & \mathbf{r}_a^T \\ \mathbf{r}_a & \tilde{R} \end{bmatrix}\begin{bmatrix} 1 \\ \boldsymbol{\alpha} \end{bmatrix} = \begin{bmatrix} \rho_a + \mathbf{r}_a^T\boldsymbol{\alpha} \\ \mathbf{r}_a + \tilde{R}\boldsymbol{\alpha} \end{bmatrix} = \begin{bmatrix} E_a \\ \mathbf{0} \end{bmatrix} = E_a\mathbf{u}$$

Backward Prediction and LU Factorization

Next, we discuss the connection of the forward and backward predictors to the Gram-Schmidt procedure and to the Cholesky factorizations of the covariance matrix R. Consider an arbitrary unit lower triangular matrix $\bar{L}$ of the same dimension as $\bar{R}$ and form the larger unit lower triangular matrix whose bottom row is $\mathbf{b}^T = [\boldsymbol{\beta}^T, 1]$

$$L = \begin{bmatrix} \bar{L} & \mathbf{0} \\ \boldsymbol{\beta}^T & 1 \end{bmatrix} \tag{1.8.13}$$

Then, it follows from Eq. (1.8.12) that

$$LRL^T = \begin{bmatrix} \bar{L}\bar{R}\bar{L}^T & \mathbf{0} \\ \mathbf{0}^T & E_b \end{bmatrix} \tag{1.8.14}$$

Indeed, we have

$$LRL^T = \begin{bmatrix} \bar{L} & \mathbf{0} \\ \boldsymbol{\beta}^T & 1 \end{bmatrix} \begin{bmatrix} \bar{R} & \mathbf{r}_b \\ \mathbf{r}_b^T & \rho_b \end{bmatrix} L^T = \begin{bmatrix} \bar{L}\bar{R} & \bar{L}\mathbf{r}_b \\ \boldsymbol{\beta}^T\bar{R} + \mathbf{r}_b^T & \boldsymbol{\beta}^T\mathbf{r}_b + \rho_b \end{bmatrix} L^T = \begin{bmatrix} \bar{L}\bar{R} & \bar{L}\mathbf{r}_b \\ \mathbf{0}^T & E_b \end{bmatrix} L^T$$

$$= \begin{bmatrix} \bar{L}\bar{R}\bar{L}^T & \bar{L}\mathbf{r}_b + \bar{L}\bar{R}\boldsymbol{\beta} \\ \mathbf{0}^T & E_b \end{bmatrix} = \begin{bmatrix} \bar{L}\bar{R}\bar{L}^T & \mathbf{0} \\ \mathbf{0}^T & E_b \end{bmatrix}$$

Defining the transformed random vector $\mathbf{e}_b = L\mathbf{y}$, we have

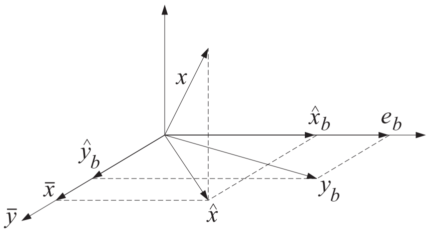
$$\mathbf{e}_b = L\mathbf{y} = \begin{bmatrix} \bar{L} & \mathbf{0} \\ \boldsymbol{\beta}^T & 1 \end{bmatrix} \begin{bmatrix} \bar{\mathbf{y}} \\ y_b \end{bmatrix} = \begin{bmatrix} \bar{L}\bar{\mathbf{y}} \\ \boldsymbol{\beta}^T\bar{\mathbf{y}} + y_b \end{bmatrix} = \begin{bmatrix} \bar{\mathbf{e}}_b \\ e_b \end{bmatrix} \tag{1.8.15}$$

where $\bar{\mathbf{e}}_b = \bar{L}\bar{\mathbf{y}}$. It follows that LRL^T is the covariance matrix of the transformed vector $\mathbf{e}_b$. The significance of Eq. (1.8.14) is that by replacing the $\mathbf{y}$ basis by $\mathbf{e}_b$ we have achieved partial decorrelation of the random vector $\mathbf{y}$. The new basis $\mathbf{e}_b$ is better to work with because it contains less redundancy than $\mathbf{y}$. For example, choosing $\bar{L}$ to be the identity matrix, $\bar{L} = \bar{I}$, Eqs. (1.8.14) and (1.8.15) become

$$LRL^T = \begin{bmatrix} \bar{R} & \mathbf{0} \\ \mathbf{0}^T & E_b \end{bmatrix}, \quad \mathbf{e}_b = \begin{bmatrix} \bar{\mathbf{y}} \\ e_b \end{bmatrix} \tag{1.8.16}$$

This represents the direct sum decomposition of the subspace spanned by $\mathbf{y}$ into the subspace spanned by $\bar{\mathbf{y}}$ and an orthogonal part spanned by e_b, that is,

$$\{\mathbf{y}\} = \{\bar{\mathbf{y}}, y_b\} = \{\bar{\mathbf{y}}\} \oplus \{e_b\}$$



The advantage of the new basis may be appreciated by considering the estimation of a random variable x in terms of $\mathbf{y}$. The estimate $\hat{x}$ may be expressed either in the $\mathbf{y}$ basis, or in the new basis $\mathbf{e}_b$ by

$$\hat{x} = E[x\mathbf{y}^T]E[\mathbf{y}\mathbf{y}^T]^{-1}\mathbf{y} = E[x\,\mathbf{e}_b^T]E[\mathbf{e}_b\mathbf{e}_b^T]^{-1}\mathbf{e}_b$$

Using the orthogonality between $\bar{\mathbf{y}}$ and e_b, or the block-diagonal property of the covariance matrix of $\mathbf{e}_b$ given by Eq. (1.8.16), we find

$$\hat{x} = E[x\bar{\mathbf{y}}^T]E[\bar{\mathbf{y}}\bar{\mathbf{y}}^T]^{-1}\bar{\mathbf{y}} + E[xe_b]E[e_b^2]^{-1}e_b = \bar{x} + \hat{x}_b$$

The two terms in $\hat{x}$ are recognized as the estimates of x based on the two orthogonal parts of the $\mathbf{y}$ basis. The first term still requires the computation of a matrix inverse, namely, $\bar{R}^{-1} = E[\bar{\mathbf{y}}\bar{\mathbf{y}}^T]^{-1}$, but the order of the matrix is reduced by one as compared with the original covariance matrix R. The same order-reduction procedure can now be applied to $\bar{R}$ itself, thereby reducing its order by one. And so on, by repeating the order-reduction procedure, the original matrix R can be completely diagonalized. This process is equivalent to performing Gram-Schmidt orthogonalization on $\mathbf{y}$ starting with y_a and ending with y_b. It is also equivalent to choosing $\bar{L}$ to correspond to the LU Cholesky factorization of $\bar{R}$. Then, the matrix L will correspond to the LU factorization of R. Indeed, if $\bar{L}$ is such that $\bar{L}\bar{R}\bar{L}^T = \bar{D}_b$, that is, a diagonal matrix, then

$$LRL^T = \begin{bmatrix} \bar{L}\bar{R}\bar{L}^T & \mathbf{0} \\ \mathbf{0}^T & E_b \end{bmatrix} = \begin{bmatrix} \bar{D}_b & \mathbf{0} \\ \mathbf{0}^T & E_b \end{bmatrix} = D_b \tag{1.8.17}$$

will itself be diagonal. The basis $\mathbf{e}_b = L\mathbf{y}$ will be completely decorrelated, having diagonal covariance matrix $E[\mathbf{e}_b\mathbf{e}_b^T] = D_b$. Thus, by successively solving backward prediction problems of lower and lower order we eventually orthogonalize the original basis $\mathbf{y}$ and obtain the LU factorization of its covariance matrix. By construction, the bottom row of L is the backward predictor $\mathbf{b}^T$. Similarly, the bottom row of $\bar{L}$ will be the backward predictor of order one less, and so on. In other words, the rows of L are simply the *backward predictors* of successive orders. The overall construction of L is illustrated by the following example.

Example 1.8.1: The random vector $\mathbf{y} = [y_a, y_c, y_b]^T$ has covariance matrix

$$R = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 3 & 2 \\ 0 & 2 & 3 \end{bmatrix}$$

By successively solving backward prediction problems of lower and lower order construct the LU factorization of R.

Solution: The backward prediction coefficients for predicting y_b are given by Eq. (1.8.7):

$$\boldsymbol{\beta} = -\bar{R}^{-1}\mathbf{r}_b = -\begin{bmatrix} 1 & 1 \\ 1 & 3 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 2 \end{bmatrix} = -\frac{1}{2}\begin{bmatrix} 3 & -1 \\ -1 & 1 \end{bmatrix}\begin{bmatrix} 0 \\ 2 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

Thus, $\mathbf{b}^T = [\boldsymbol{\beta}^T, 1] = [1, -1, 1]$. The estimation error is given by Eq. (1.8.11):

$$E_b = \rho_b + \boldsymbol{\beta}^T\mathbf{r}_b = 3 + [1, -1]\begin{bmatrix} 0 \\ 2 \end{bmatrix} = 1$$

Repeating the procedure on $\bar{R} = \begin{bmatrix} 1 & 1 \\ 1 & 3 \end{bmatrix}$, we find for the corresponding backward prediction coefficients, satisfying $\bar{R}\bar{\mathbf{b}} = \bar{E}_b\bar{\mathbf{v}}$, $\bar{\mathbf{v}} = [0, 1]^T$

$$\bar{\boldsymbol{\beta}} = -[1]^{-1}[1] = [-1], \quad \bar{\mathbf{b}}^T = [\bar{\boldsymbol{\beta}}^T, 1] = [-1, 1]$$

and $\bar{E}_b = \bar{\rho}_b + \bar{\boldsymbol{\beta}}^T\bar{\mathbf{r}}_b = 3 - 1\times 1 = 2$. The rows of L are the backward predictor coefficients, and the diagonal entries of D_b are the E_b. Thus,

$$L = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 1 & -1 & 1 \end{bmatrix}, \quad D_b = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

It is easily verified that $LRL^T = D_b$. Note that the first entry of D_b is always equal to ρ_a. Next, we obtain the same results by carrying out the Gram-Schmidt construction starting at y_a and ending with y_b. Starting with $\epsilon_1 = y_a$ and $E[\epsilon_1^2] = 1$, define

$$\epsilon_2 = y_c - E[y_c\epsilon_1]E[\epsilon_1^2]^{-1}\epsilon_1 = y_c - y_a$$

having $E[\epsilon_2^2] = E[y_c^2] - 2E[y_cy_a] + E[y_a^2] = 2$. Thus, the $\bar{\mathbf{e}}_b$ portion of the Gram-Schmidt construction will be

$$\bar{\mathbf{e}}_b = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix}\begin{bmatrix} y_a \\ y_c \end{bmatrix} = \bar{L}\bar{\mathbf{y}}$$

The last step of the Gram-Schmidt construction is

$$e_b = y_b - E[y_b\epsilon_1]E[\epsilon_1^2]^{-1}\epsilon_1 - E[y_b\epsilon_2]E[\epsilon_2^2]^{-1}\epsilon_2 = y_b - (y_c - y_a) = y_a - y_c + y_b$$

giving for the last row of L, $\mathbf{b}^T = [1, -1, 1]$. In the above step, we used

$$E[y_b\epsilon_2] = E[y_b(y_c - y_a)] = E[y_by_c] - E[y_by_a] = 2 - 0 = 2$$

and $E[y_b\epsilon_1] = E[y_by_a] = 0$. □

Linear Estimation in the Backward Basis

Equation (1.8.17) may be written in the form

$$R = L^{-1}D_bL^{-T} \tag{1.8.18}$$

where L^{-T} is the inverse of the transpose of L. Thus, L^{-1} and L^{-T} correspond to the conventional LU Cholesky factors of R. The computational advantage of this form becomes immediately obvious when we consider the inverse of R,

$$R^{-1} = L^TD_b^{-1}L \tag{1.8.19}$$

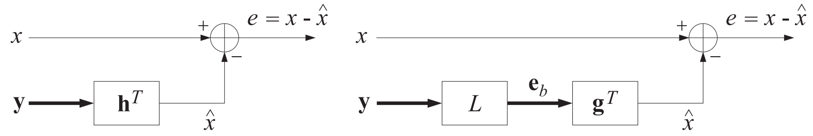
which shows that R^{-1} can be computed without any matrix inversion (the inverse of the diagonal matrix D_b is trivial). The design of linear estimators is simplified considerably in the $\mathbf{e}_b$ basis. The estimate of x is

$$\hat{x} = \mathbf{h}^T\mathbf{y} \tag{1.8.20}$$

where $\mathbf{h} = E[\mathbf{y}\mathbf{y}^T]^{-1}E[x\mathbf{y}] \equiv R^{-1}\mathbf{r}$. Writing $\mathbf{y} = L^{-1}\mathbf{e}_b$ and defining a new vector of estimation weights by $\mathbf{g} = L^{-T}\mathbf{h}$, we can rewrite Eq. (1.8.20) as

$$\hat{x} = \mathbf{h}^T\mathbf{y} = \mathbf{g}^T\mathbf{e}_b \tag{1.8.21}$$

The block diagram representations of the two realizations are shown below:



There are three major advantages of the representation of Eq. (1.8.21) over Eq. (1.8.20). First, to get the estimate $\hat{x}$ using (1.8.20), the processor has to linearly combine a lot of *redundant* information because the $\mathbf{y}$ basis is correlated, whereas the processor (1.8.21) linearly combines only the *non-redundant* part of the same information. This has important implications for the adaptive implementations of such processors. An adaptive processor that uses the representation (1.8.20) will tend to be slow in learning the statistics of the data vector $\mathbf{y}$ because it has to process all the redundancies in the data. Moreover, the more the redundancies, or equivalently, the higher the correlations in the data $\mathbf{y}$, the slower the speed of adaptation. On the other hand, an adaptive processor based on (1.8.21) should adapt very quickly. The preprocessing operation, $\mathbf{e}_b = L\mathbf{y}$, that decorrelates the data vector $\mathbf{y}$ can also be implemented adaptively. In time series applications, it is conveniently realized by means of a *lattice structure*. In adaptive array applications, it gives rise to the so-called *Gram-Schmidt preprocessor* implementations.

Second, the computation of $\mathbf{g}$ can be done efficiently without any matrix inversion. Given the LU factors of R as in Eq. (1.8.19) and the cross correlation vector $\mathbf{r}$, we may compute $\mathbf{g}$ by

$$\mathbf{g} = L^{-T}\mathbf{h} = L^{-T}R^{-1}\mathbf{r} = L^{-T}(L^TD_b^{-1}L)\mathbf{r} = D_b^{-1}L\mathbf{r} \tag{1.8.22}$$

If so desired, the original weights $\mathbf{h}$ may be recovered from $\mathbf{g}$ by

$$\mathbf{h} = L^T\mathbf{g} \tag{1.8.23}$$

The third advantage of the form Eq. (1.8.21) is that any lower-order portion of the weight vector $\mathbf{g}$ is already optimal for that order. Thus, the order of the estimator can be increased without having to redesign the lower-order portions of it. Recognizing that $L\mathbf{r} = LE[x\mathbf{y}] = E[x\,\mathbf{e}_b]$, we write Eq. (1.8.22) as

$$\mathbf{g} = D_b^{-1}E[x\,\mathbf{e}_b] = \begin{bmatrix} \bar{D}_b^{-1}E[x\,\bar{\mathbf{e}}_b] \\ E_b^{-1}E[xe_b] \end{bmatrix} \equiv \begin{bmatrix} \bar{\mathbf{g}} \\ g \end{bmatrix}$$

where we used the diagonal nature of D_b given in Eq. (1.8.17) and the decomposition (1.8.15). The estimate (1.8.21) can be written as

$$\hat{x} = \mathbf{g}^T\mathbf{e}_b = [\bar{\mathbf{g}}^T, g]\begin{bmatrix} \bar{\mathbf{e}}_b \\ e_b \end{bmatrix} = \bar{\mathbf{g}}^T\bar{\mathbf{e}}_b + ge_b \equiv \bar{x} + \hat{x}_b \tag{1.8.24}$$

It is clear that the two terms

$$\bar{x} = \bar{\mathbf{g}}^T\bar{\mathbf{e}}_b = E[x\,\bar{\mathbf{e}}_b^T]\bar{D}_b^{-1}\bar{\mathbf{e}}_b\,, \quad \hat{x}_b = ge_b = E[xe_b]E[e_b^2]^{-1}e_b \tag{1.8.25}$$

are the optimal estimates of x based on the two orthogonal parts of the subspace of observations, namely,

$$\{\mathbf{y}\} = \{\bar{\mathbf{y}}\} \oplus \{e_b\}\,, \quad \text{or,} \quad \{\mathbf{e}_b\} = \{\bar{\mathbf{e}}_b\} \oplus \{e_b\}$$

The first term, $\bar{x}$, is the same estimate of x based on $\bar{\mathbf{y}}$ that we considered earlier but now it is expressed in the diagonal basis $\bar{\mathbf{e}}_b = \bar{L}\bar{\mathbf{y}}$. The second term, $\hat{x}_b$, represents the improvement in that estimate that arises by taking into account one more observation, namely, y_b. It represents that part of x that cannot be estimated from $\bar{\mathbf{y}}$. And, it is computable only from that part of the new observation y_b that *cannot* be predicted from $\bar{\mathbf{y}}$, that is, e_b. The degree of improvement of $\hat{x}$ over $\bar{x}$, as measured by the mean-square estimation errors, can be computed explicitly in this basis. To see this, denote the estimation errors based on $\mathbf{y}$ and $\bar{\mathbf{y}}$ by

$$e = x - \hat{x} = x - \mathbf{g}^T\mathbf{e}_b\,, \quad \bar{e} = x - \bar{x} = x - \bar{\mathbf{g}}^T\bar{\mathbf{e}}_b$$

Then, Eq. (1.8.24) implies $e = x - \hat{x} = (x - \bar{x}) - \hat{x}_b$, or

$$e = \bar{e} - ge_b \tag{1.8.26}$$

Because e and $\mathbf{y}$, or $\mathbf{e}_b$, are orthogonal, we have $E[\hat{x}e] = 0$, which implies that

$$\mathcal{E} = E[e^2] = E[xe] = E\big[x(x - \mathbf{g}^T\mathbf{e}_b)\big] = E[x^2] - \mathbf{g}^T E[x\,\mathbf{e}_b]$$

Similarly, $\bar{\mathcal{E}} = E[\bar{e}^2] = E[x^2] - \bar{\mathbf{g}}^T E[x\,\bar{\mathbf{e}}_b]$. It follows that

$$\mathcal{E} = \bar{\mathcal{E}} - gE[xe_b] = \bar{\mathcal{E}} - g^2 E_b \tag{1.8.27}$$

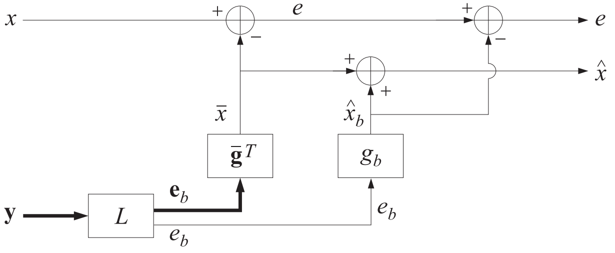
where we used $g = E[xe_b]E_b^{-1}$. The subtracted term represents the improvement obtained by including one more observation in the estimate. It follows from the above discussion that the lower-order portion $\bar{\mathbf{g}}$ of $\mathbf{g}$ is already optimal. This is not so in the $\mathbf{y}$ basis, that is, the lower-order portion of $\mathbf{h}$ is not equal to the lower-order optimal weights $\bar{\mathbf{h}} = \bar{R}^{-1}\bar{\mathbf{r}}$, where $\bar{\mathbf{r}} = E[x\bar{\mathbf{y}}]$. The explicit relationship between the two may be found as follows. Inserting the block decomposition Eq. (1.8.13) of L into Eq. (1.8.19) and using the lower-order result $\bar{R}^{-1} = \bar{L}^T\bar{D}_b^{-1}\bar{L}$, we may derive the following order-updating expression for R^{-1}

$$R^{-1} = \begin{bmatrix} \bar{R}^{-1} & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix} + \frac{1}{E_b}\mathbf{b}\mathbf{b}^T \tag{1.8.28}$$

Noting that $\bar{\mathbf{r}}$ is the lower-order part of $\mathbf{r}$, $\mathbf{r} = [\bar{\mathbf{r}}^T, r_b]^T$, where $r_b = E[xy_b]$, we obtain the following order-updating equation for the optimal $\mathbf{h}$

$$\mathbf{h} = R^{-1}\mathbf{r} = \begin{bmatrix} \bar{R}^{-1} & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix}\begin{bmatrix} \bar{\mathbf{r}} \\ r_b \end{bmatrix} + \frac{1}{E_b}(\mathbf{b}\mathbf{b}^T)\mathbf{r} = \begin{bmatrix} \bar{\mathbf{h}} \\ 0 \end{bmatrix} + c_b\mathbf{b} \tag{1.8.29}$$

where $c_b = (\mathbf{b}^T\mathbf{r})/E_b = (\boldsymbol{\beta}^T\bar{\mathbf{r}} + r_b)/E_b$. A block diagram realization that takes into account the order-recursive construction of the estimate (1.8.24) and estimation error (1.8.26) is shown below.



In Chap. 12, we discuss in greater detail the design procedure given by Eq. (1.8.22) and show how to realize Eqs. (1.8.21), or (1.8.24) and (1.8.26), by means of a *lattice structure*. In Chap. 16, we discuss the corresponding adaptive versions, leading to the so-called *adaptive lattice filters* for linear prediction and Wiener filtering, such as the gradient lattice and RLS lattice.

Forward Prediction and UL Factorization

Next, we turn our attention to the forward predictors defined in Eq. (1.8.12). They lead to UL (rather than LU) factorization of the covariance matrix. Considering an arbitrary unit upper-triangular matrix $\tilde{U}$ of the same dimension as $\tilde{R}$, we may form the larger unit upper-triangular matrix whose top row is the forward predictor $\mathbf{a}^T = [1, \boldsymbol{\alpha}^T]$

$$U = \begin{bmatrix} 1 & \boldsymbol{\alpha}^T \\ \mathbf{0} & \tilde{U} \end{bmatrix} \tag{1.8.30}$$

Then, it follows from Eq. (1.8.12) that

$$URU^T = \begin{bmatrix} E_a & \mathbf{0}^T \\ \mathbf{0} & \tilde{U}\tilde{R}\tilde{U}^T \end{bmatrix} \tag{1.8.31}$$

It follows that URU^T is the covariance matrix of the transformed vector

$$\mathbf{e}_a = U\mathbf{y} = \begin{bmatrix} 1 & \boldsymbol{\alpha}^T \\ \mathbf{0} & \tilde{U} \end{bmatrix}\begin{bmatrix} y_a \\ \tilde{\mathbf{y}} \end{bmatrix} = \begin{bmatrix} y_a + \boldsymbol{\alpha}^T\tilde{\mathbf{y}} \\ \tilde{U}\tilde{\mathbf{y}} \end{bmatrix} = \begin{bmatrix} e_a \\ \tilde{\mathbf{e}}_a \end{bmatrix} \tag{1.8.32}$$

Choosing $\tilde{U}$ to correspond to the UL factor of $\tilde{R}$, that is, $\tilde{U}\tilde{R}\tilde{U}^T = \tilde{D}_a$, where $\tilde{D}_a$ is diagonal, then Eq. (1.8.31) implies that U will correspond to the UL factor of R:

$$URU^T = \begin{bmatrix} E_a & \mathbf{0}^T \\ \mathbf{0} & \tilde{D}_a \end{bmatrix} = D_a \tag{1.8.33}$$

This is equivalent to Eq. (1.7.14). The basis $\mathbf{e}_a = U\mathbf{y}$ is completely decorrelated, with covariance matrix $E[\mathbf{e}_a\mathbf{e}_a^T] = D_a$. It is equivalent to Eq. (1.7.13). The rows of U are the *forward* predictors of successive orders. And therefore, the UL factorization of R is equivalent to performing the Gram-Schmidt construction starting at the endpoint y_b and proceeding to y_a. The following example illustrates the method.

Example 1.8.2: By successively solving forward prediction problems of lower and lower order, construct the UL factorization of the covariance matrix R of Example 1.8.1.

Solution: Using Eq. (1.8.7), we find

$$\boldsymbol{\alpha} = -\tilde{R}^{-1}\mathbf{r}_a = -\begin{bmatrix} 3 & 2 \\ 2 & 3 \end{bmatrix}^{-1}\begin{bmatrix} 1 \\ 0 \end{bmatrix} = -\frac{1}{5}\begin{bmatrix} 3 & -2 \\ -2 & 3 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} -3/5 \\ 2/5 \end{bmatrix}$$

Thus, $\mathbf{a}^T = [1, \boldsymbol{\alpha}^T] = [1, -3/5, 2/5]$. The estimation error is

$$E_a = \rho_a + \boldsymbol{\alpha}^T\mathbf{r}_a = 1 + [-3/5, 2/5]\begin{bmatrix} 1 \\ 0 \end{bmatrix} = \frac{2}{5}$$

Repeating the procedure on $\tilde{R} = \begin{bmatrix} 3 & 2 \\ 2 & 3 \end{bmatrix}$, we find the corresponding forward prediction coefficients, satisfying $\tilde{R}\tilde{\mathbf{a}} = \tilde{E}_a\tilde{\mathbf{u}}$, where $\tilde{\mathbf{u}} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$,

$$\tilde{\boldsymbol{\alpha}} = -[3]^{-1}[2] = -\frac{2}{3}, \quad \tilde{\mathbf{a}}^T = [1, \tilde{\boldsymbol{\alpha}}^T] = [1, -2/3]$$

and $\tilde{E}_a = \tilde{\rho}_a + \tilde{\boldsymbol{\alpha}}^T\tilde{\mathbf{r}}_a = 3 - (2/3)\times 2 = 5/3$. The rows of U are the forward predictor coefficients and the diagonal entries of D_a are the E_as:

$$U = \begin{bmatrix} 1 & -3/5 & 2/5 \\ 0 & 1 & -2/3 \\ 0 & 0 & 1 \end{bmatrix}, \quad D_a = \begin{bmatrix} 2/5 & 0 & 0 \\ 0 & 5/3 & 0 \\ 0 & 0 & 3 \end{bmatrix}$$

It is easily verified that $URU^T = D_a$. Note that the last entry of D_a is always equal to ρ_b. □

Equation (1.8.33) can be used to compute the inverse of R:

$$R^{-1} = U^T D_a^{-1} U \tag{1.8.34}$$

Using the lower-order result $\tilde{R}^{-1} = \tilde{U}^T\tilde{D}_a^{-1}\tilde{U}$ and the decomposition (1.8.30), we find the following order-updating equation for R^{-1}, analogous to Eq. (1.8.28):

$$R^{-1} = \begin{bmatrix} 0 & \mathbf{0}^T \\ \mathbf{0} & \tilde{R}^{-1} \end{bmatrix} + \frac{1}{E_a}\mathbf{a}\mathbf{a}^T \tag{1.8.35}$$

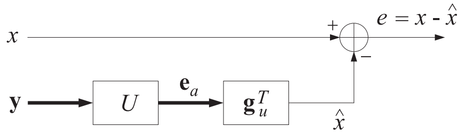
Denoting $\tilde{\mathbf{r}} = E[x\tilde{\mathbf{y}}]$ and $r_a = E[xy_a]$, we obtain the alternative order-update equation for $\mathbf{h}$, analogous to Eq. (1.8.29):

$$\mathbf{h} = R^{-1}\mathbf{r} = \begin{bmatrix} 0 & \mathbf{0}^T \\ \mathbf{0} & \tilde{R}^{-1} \end{bmatrix}\begin{bmatrix} r_a \\ \tilde{\mathbf{r}} \end{bmatrix} + \frac{1}{E_a}(\mathbf{a}^T\mathbf{r})\mathbf{a} = \begin{bmatrix} 0 \\ \tilde{\mathbf{h}} \end{bmatrix} + c_a\mathbf{a} \tag{1.8.36}$$

where $c_a = (\mathbf{a}^T\mathbf{r})/E_a = (r_a + \boldsymbol{\alpha}^T\tilde{\mathbf{r}})/E_a$, and $\tilde{\mathbf{h}} = \tilde{R}^{-1}\tilde{\mathbf{r}}$ is the lower-order optimal estimator for estimating x from $\tilde{\mathbf{y}}$. By analogy with Eq. (1.8.21), we could also choose to express the estimates in the $\mathbf{e}_a$ basis

$$\hat{x} = \mathbf{h}^T\mathbf{y} = \mathbf{h}^T U^{-1}\mathbf{e}_a = \mathbf{g}_u^T\mathbf{e}_a \tag{1.8.37}$$

where $\mathbf{g}_u = U^{-T}\mathbf{h}$. A realization is shown below.



The most important part of the realizations based on the diagonal bases $\mathbf{e}_a$ or $\mathbf{e}_a$ is the preprocessing part that decorrelates the $\mathbf{y}$ basis, namely, $\mathbf{e}_b = L\mathbf{y}$, or $\mathbf{e}_a = U\mathbf{y}$. We will see in Chapters 12 and 16 that this part can be done efficiently using the Levinson recursion and the *lattice structures* of linear prediction. The LU representation, based on the backward predictors, $\mathbf{e}_b = L\mathbf{y}$, is preferred because it is somewhat more conveniently realized in terms of the lattice structure than the UL representation $\mathbf{e}_a = U\mathbf{y}$.

Order Updates

So far, we studied the problems of forward and backward prediction separately from each other. Next, we would like to consider the two problems together and show how to construct the solution of the pair of equations (1.8.12) from the solution of a similar pair of lower order. This construction is the essence behind Levinson's algorithm for solving the linear prediction problem, both in the stationary and in the adaptive least squares cases. Consider the following pair of lower-order forward and backward predictors, defined in terms of the block decompositions (1.8.5) of R:

$$\bar{R}\bar{\mathbf{a}} = \bar{E}_a\bar{\mathbf{u}}, \quad \tilde{R}\tilde{\mathbf{b}} = \tilde{E}_b\tilde{\mathbf{v}} \tag{1.8.38}$$

where $\bar{\mathbf{u}}$ and $\tilde{\mathbf{v}}$ are unit vectors of dimension one less than those of Eq. (1.8.12). They are related to $\mathbf{u}$ and $\mathbf{v}$ through the decompositions

$$\mathbf{u} = \begin{bmatrix} \bar{\mathbf{u}} \\ 0 \end{bmatrix}, \quad \mathbf{v} = \begin{bmatrix} 0 \\ \tilde{\mathbf{v}} \end{bmatrix} \tag{1.8.39}$$

The basic result we would like to show is that the solution of the pair (1.8.12) may be constructed from the solution of the pair (1.8.38) by

$$\begin{aligned} \mathbf{a} &= \begin{bmatrix} \bar{\mathbf{a}} \\ 0 \end{bmatrix} - \gamma_b \begin{bmatrix} 0 \\ \tilde{\mathbf{b}} \end{bmatrix} \\ \mathbf{b} &= \begin{bmatrix} 0 \\ \tilde{\mathbf{b}} \end{bmatrix} - \gamma_a \begin{bmatrix} \bar{\mathbf{a}} \\ 0 \end{bmatrix} \end{aligned} \tag{1.8.40}$$

This result is motivated by Eq. (1.8.39), which shows that the right-hand sides of Eqs. (1.8.38) are already part of the right-hand sides of Eq. (1.8.12), and therefore, the solutions of Eq. (1.8.38) may appear as part of the solutions of (1.8.12). The prediction errors are updated by

$$E_a = (1 - \gamma_a\gamma_b)\bar{E}_a, \quad E_b = (1 - \gamma_a\gamma_b)\tilde{E}_b \tag{1.8.41}$$

where

$$\gamma_b = \frac{\Delta_a}{\tilde{E}_b}, \quad \gamma_a = \frac{\Delta_b}{\bar{E}_a} \tag{1.8.42}$$

The γs are known as the *reflection* or *PARCOR* coefficients. The quantities Δ_a and Δ_b are defined by

$$\Delta_a = \bar{\mathbf{a}}^T\mathbf{r}_b, \quad \Delta_b = \tilde{\mathbf{b}}^T\mathbf{r}_a \tag{1.8.43}$$

The two Δs are equal, $\Delta_a = \Delta_b$, as seen from the following considerations. Using the decompositions (1.8.5), we find

$$R\begin{bmatrix} \bar{\mathbf{a}} \\ 0 \end{bmatrix} = \begin{bmatrix} \bar{R} & \mathbf{r}_b \\ \mathbf{r}_b^T & \rho_b \end{bmatrix}\begin{bmatrix} \bar{\mathbf{a}} \\ 0 \end{bmatrix} = \begin{bmatrix} \bar{R}\bar{\mathbf{a}} \\ \mathbf{r}_b^T\bar{\mathbf{a}} \end{bmatrix} = \begin{bmatrix} \bar{E}_a\bar{\mathbf{u}} \\ \Delta_a \end{bmatrix}$$

$$R\begin{bmatrix} 0 \\ \tilde{\mathbf{b}} \end{bmatrix} = \begin{bmatrix} \rho_a & \mathbf{r}_a^T \\ \mathbf{r}_a & \tilde{R} \end{bmatrix}\begin{bmatrix} 0 \\ \tilde{\mathbf{b}} \end{bmatrix} = \begin{bmatrix} \mathbf{r}_a^T\tilde{\mathbf{b}} \\ \tilde{R}\tilde{\mathbf{b}} \end{bmatrix} = \begin{bmatrix} \Delta_b \\ \tilde{E}_b\tilde{\mathbf{v}} \end{bmatrix}$$

They may be written more conveniently as

$$R\begin{bmatrix}\bar{\mathbf{a}}\\0\end{bmatrix}=\begin{bmatrix}\bar{E}_a\bar{\mathbf{u}}\\\Delta_a\end{bmatrix}=\bar{E}_a\begin{bmatrix}\bar{\mathbf{u}}\\0\end{bmatrix}+\Delta_a\begin{bmatrix}\mathbf{0}\\1\end{bmatrix}=\bar{E}_a\mathbf{u}+\Delta_a\mathbf{v} \tag{1.8.44a}$$

$$R\begin{bmatrix}0\\\tilde{\mathbf{b}}\end{bmatrix}=\begin{bmatrix}\Delta_b\\\tilde{E}_b\tilde{\mathbf{v}}\end{bmatrix}=\Delta_b\begin{bmatrix}1\\\mathbf{0}\end{bmatrix}+\tilde{E}_b\begin{bmatrix}0\\\tilde{\mathbf{v}}\end{bmatrix}=\Delta_b\mathbf{u}+\tilde{E}_b\mathbf{v} \tag{1.8.44b}$$

Noting that $\mathbf{d}^T\mathbf{u}$ and $\mathbf{d}^T\mathbf{v}$ are equal to the first and last components of a vector $\mathbf{d}$, we have $[0,\tilde{\mathbf{b}}^T]\mathbf{u}=0$ and $[0,\tilde{\mathbf{b}}^T]\mathbf{v}=1$ because the first and last components of $[0,\tilde{\mathbf{b}}^T]$ are zero and one, respectively. Similarly, $[\bar{\mathbf{a}}^T,0]\mathbf{u}=1$ and $[\bar{\mathbf{a}}^T,0]\mathbf{v}=0$. Thus, multiplying Eq. (1.8.44a) from the left by $[0,\tilde{\mathbf{b}}^T]$ and Eq. (1.8.44b) by $[\bar{\mathbf{a}}^T,0]$, we find

$$[0,\tilde{\mathbf{b}}^T]R\begin{bmatrix}\bar{\mathbf{a}}\\0\end{bmatrix}=\Delta_a\,,\quad [\bar{\mathbf{a}}^T,0]R\begin{bmatrix}0\\\tilde{\mathbf{b}}\end{bmatrix}=\Delta_b \tag{1.8.45}$$

The equality of the Δs follows now from the fact that R is a symmetric matrix. Thus,

$$\Delta_a=\Delta_b\equiv\Delta \tag{1.8.46}$$

An alternative proof, based on partial correlations, will be given later. Equations (1.8.40) and (1.8.41) follow now in a straightforward fashion from Eq. (1.8.44). Multiplying the first part of Eq. (1.8.40) by R and using Eqs. (1.8.12) and (1.8.44), we find

$$E_a\mathbf{u}=R\mathbf{a}=R\begin{bmatrix}\bar{\mathbf{a}}\\0\end{bmatrix}-\gamma_bR\begin{bmatrix}0\\\tilde{\mathbf{b}}\end{bmatrix}$$

or,

$$E_a\mathbf{u}=(\bar{E}_a\mathbf{u}+\Delta_a\mathbf{v})-\gamma_b(\Delta_b\mathbf{u}+\tilde{E}_b\mathbf{v})=(\bar{E}_a-\gamma_b\Delta_b)\mathbf{u}+(\Delta_b-\gamma_b\tilde{E}_b)\mathbf{v}$$

which implies the conditions

$$E_a=\bar{E}_a-\gamma_b\Delta_b\,,\quad \Delta_a-\gamma_b\tilde{E}_b=0 \tag{1.8.47}$$

Similarly, multiplying the second part of the Eq. (1.8.40) by R, we obtain

$$E_b\mathbf{v}=(\Delta_b\mathbf{u}+\tilde{E}_b\mathbf{v})-\gamma_a(\bar{E}_a\mathbf{u}+\Delta_b\mathbf{v})=(\Delta_b-\gamma_a\bar{E}_a)\mathbf{u}+(\tilde{E}_b-\gamma_a\Delta_a)\mathbf{v}$$

which implies

$$E_b=\tilde{E}_b-\gamma_a\Delta_a\,,\quad \Delta_b-\gamma_a\bar{E}_a=0 \tag{1.8.48}$$

Equations (1.8.41) and (1.8.42) follow now from (1.8.47) and (1.8.48). By analogy with Eq. (1.8.9), we may now define the prediction errors corresponding to the lower-order predictors $\bar{\mathbf{a}}$ and $\tilde{\mathbf{b}}$ by

$$\bar{e}_a=\bar{\mathbf{a}}^T\bar{\mathbf{y}}\,,\quad \tilde{e}_b=\tilde{\mathbf{b}}^T\tilde{\mathbf{y}} \tag{1.8.49}$$

Using Eqs. (1.8.9) and (1.8.40), we find the following updating equations for the prediction errors

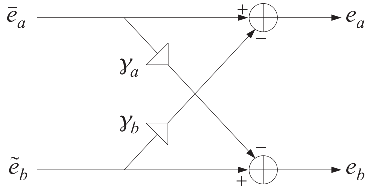
$$\mathbf{a}^T\mathbf{y}=[\bar{\mathbf{a}}^T,0]\begin{bmatrix}\bar{\mathbf{y}}\\y_b\end{bmatrix}-\gamma_b[0,\tilde{\mathbf{b}}^T]\begin{bmatrix}y_a\\\tilde{\mathbf{y}}\end{bmatrix}=\bar{\mathbf{a}}^T\bar{\mathbf{y}}-\gamma_b\tilde{\mathbf{b}}^T\tilde{\mathbf{y}}$$

$$\mathbf{b}^T\mathbf{y}=[0,\tilde{\mathbf{b}}^T]\begin{bmatrix}y_a\\\tilde{\mathbf{y}}\end{bmatrix}-\gamma_a[\bar{\mathbf{a}}^T,0]\begin{bmatrix}\bar{\mathbf{y}}\\y_b\end{bmatrix}=\tilde{\mathbf{b}}^T\tilde{\mathbf{y}}-\gamma_a\bar{\mathbf{a}}^T\bar{\mathbf{y}}$$

or,

$$e_a=\bar{e}_a-\gamma_b\tilde{e}_b\,,\quad e_b=\tilde{e}_b-\gamma_a\bar{e}_a \tag{1.8.50}$$

A lattice type realization ofEq. (1.8.50) is shown below. It forms the basis of the lattice structures of linear prediction discussed in Chapters 12 and 16.



The order-updating procedure is illustrated by the following example.

Example 1.8.3: Using Eq. (1.8.40), construct the forward and backward predictors **a** and **b** found previously in Examples 1.8.1 and 1.8.2.

Solution: The first part of Eq. (1.8.38), $\bar{R}\bar{\mathbf{a}}=\bar{E}_a\bar{\mathbf{u}}$ is solved as follows:

$$\begin{bmatrix}1&1\\1&3\end{bmatrix}\begin{bmatrix}1\\\bar{\alpha}\end{bmatrix}=\bar{E}_a\begin{bmatrix}1\\0\end{bmatrix}\quad\Rightarrow\quad\bar{\alpha}=-\frac{1}{3}\,,\quad\bar{E}_a=\frac{2}{3}$$

Therefore, $\bar{\mathbf{a}}=\begin{bmatrix}1\\-1/3\end{bmatrix}$. Similarly, $\tilde{R}\tilde{\mathbf{y}}=\tilde{E}_b\tilde{\mathbf{v}}$, is solved by

$$\begin{bmatrix}3&2\\2&3\end{bmatrix}\begin{bmatrix}\tilde{\beta}\\1\end{bmatrix}=\tilde{E}_b\begin{bmatrix}0\\1\end{bmatrix}\quad\Rightarrow\quad\tilde{\beta}=-\frac{2}{3}\,,\quad\tilde{E}_b=\frac{5}{3}$$

Hence, $\tilde{\mathbf{b}}=\begin{bmatrix}-2/3\\1\end{bmatrix}$. Next, we determine

$$\Delta=\bar{\mathbf{a}}^T\mathbf{r}_b=[1,-1/3]\begin{bmatrix}0\\2\end{bmatrix}=-\frac{2}{3}\,,\quad\gamma_b=\frac{\Delta}{\tilde{E}_b}=-\frac{2}{5}\,,\quad\gamma_a=\frac{\Delta}{\bar{E}_a}=-1$$

It follows from Eq. (1.8.40) that

$$\mathbf{a}=\begin{bmatrix}\bar{\mathbf{a}}\\0\end{bmatrix}-\gamma_b\begin{bmatrix}0\\\tilde{\mathbf{b}}\end{bmatrix}=\begin{bmatrix}1\\-1/3\\0\end{bmatrix}-\left(-\frac{2}{5}\right)\begin{bmatrix}0\\-2/3\\1\end{bmatrix}=\begin{bmatrix}1\\-3/5\\2/5\end{bmatrix}$$

$$\mathbf{b}=\begin{bmatrix}0\\\tilde{\mathbf{b}}\end{bmatrix}-\gamma_a\begin{bmatrix}\bar{\mathbf{a}}\\0\end{bmatrix}=\begin{bmatrix}0\\-2/3\\1\end{bmatrix}-(-1)\begin{bmatrix}1\\-1/3\\0\end{bmatrix}=\begin{bmatrix}1\\-1\\1\end{bmatrix}$$

and the prediction errors are found from Eq. (1.8.41)

$$E_a=\bar{E}_a(1-\gamma_a\gamma_b)=\frac{2}{3}(1-2/5)=\frac{2}{5}\,,\quad E_b=\tilde{E}_b(1-\gamma_a\gamma_b)=\frac{5}{3}(1-2/5)=1$$

Partial Correlation Interpretation

Next, we show that γ_a and γ_b are partial correlation coefficients in the sense of Sec. 1.7. Let $\mathbf{y}_c$ denote all the components of $\mathbf{y}$ that lie between y_a and y_b, so that

$$\mathbf{y} = \begin{bmatrix} y_a \\ \mathbf{y}_c \\ y_b \end{bmatrix}, \quad \bar{\mathbf{y}} = \begin{bmatrix} y_a \\ \mathbf{y}_c \end{bmatrix}, \quad \tilde{\mathbf{y}} = \begin{bmatrix} \mathbf{y}_c \\ y_b \end{bmatrix} \tag{1.8.51}$$

The forward predictor $\mathbf{a}$ was defined as the best estimator of y_a based on the rest of the vector $\mathbf{y}$. By the same token, $\bar{\mathbf{a}}$ is the best estimator of y_a based on the rest of $\bar{\mathbf{y}}$, that is, $\mathbf{y}_c$. Similarly, the backward predictor $\tilde{\mathbf{b}}$ defines the best estimator of y_b based on the rest of the vector $\tilde{\mathbf{y}}$; again, $\mathbf{y}_c$. Decomposing $\bar{\mathbf{a}}$ and $\tilde{\mathbf{b}}$ as

$$\bar{\mathbf{a}} = \begin{bmatrix} 1 \\ \bar{\boldsymbol{\alpha}} \end{bmatrix}, \quad \tilde{\mathbf{b}} = \begin{bmatrix} \tilde{\boldsymbol{\beta}} \\ 1 \end{bmatrix}$$

we may write the best estimates of y_a and y_b based on $\mathbf{y}_c$ as

$$\hat{y}_{a/c} = E[y_a\mathbf{y}_c^T]E[\mathbf{y}_c\mathbf{y}_c^T]^{-1}\mathbf{y}_c = -\bar{\boldsymbol{\alpha}}^T\mathbf{y}_c\,, \quad \hat{y}_{b/c} = E[y_b\mathbf{y}_c^T]E[\mathbf{y}_c\mathbf{y}_c^T]^{-1}\mathbf{y}_c = -\tilde{\boldsymbol{\beta}}^T\mathbf{y}_c$$

and the estimation errors

$$\bar{e}_a = \bar{\mathbf{a}}^T\bar{\mathbf{y}} = y_a - \hat{y}_{a/c}\,, \quad \tilde{e}_b = \tilde{\mathbf{b}}^T\tilde{\mathbf{y}} = y_b - \hat{y}_{b/c} \tag{1.8.52}$$

Thus, $\bar{e}_a$ and $\tilde{e}_b$ represent what is left of y_a and y_b after we project out their dependence on the intermediate vector $\mathbf{y}_c$. The direct influence of y_a on y_b, with the effect of $\mathbf{y}_c$ removed, is measured by the correlation $E[\bar{e}_a\tilde{e}_b]$. This correlation is equal to the quantity Δ defined in Eq. (1.8.46). This follows from Eq. (1.8.43)

$$\Delta_a = \bar{\mathbf{a}}^T\mathbf{r}_b = \bar{\mathbf{a}}^T E[y_b\bar{\mathbf{y}}] = E\big[y_b(\bar{\mathbf{a}}^T\bar{\mathbf{y}})\big] = E[y_b\bar{e}_a]$$

similarly,

$$\Delta_b = \tilde{\mathbf{b}}^T\mathbf{r}_a = \tilde{\mathbf{b}}^T E[y_a\tilde{\mathbf{y}}] = E\big[y_a(\tilde{\mathbf{b}}^T\tilde{\mathbf{y}})\big] = E[y_a\tilde{e}_b]$$

Now, because $\bar{e}_a$ is orthogonal to $\mathbf{y}_c$ and $\hat{y}_{b/c}$ is a linear combination of $\mathbf{y}_c$, it follows that $E[\hat{y}_{b/c}\bar{e}_a] = 0$. Similarly, because $\tilde{e}_b$ is orthogonal to $\mathbf{y}_c$ and $\hat{y}_{a/c}$ is linearly related to $\mathbf{y}_c$, it follows that $E[\hat{y}_{a/c}\tilde{e}_b] = 0$. Thus,

$$\Delta_a = E[y_b\bar{e}_a] = E\big[(y_b - \hat{y}_{b/c})\bar{e}_a\big] = E[\tilde{e}_b\bar{e}_a]$$

$$\Delta_b = E[y_a\tilde{e}_b] = E\big[(y_a - \hat{y}_{a/c})\tilde{e}_b\big] = E[\bar{e}_a\tilde{e}_b]$$

Therefore, Δ_a and Δ_b are equal

$$\Delta_a = \Delta_b = E[\bar{e}_a\tilde{e}_b] \tag{1.8.53}$$

This is an alternative proof of Eq. (1.8.46). It follows that γ_a and γ_b are normalized PARCOR coefficients in the sense of Sec. 1.7:

$$\gamma_b = \frac{E[\bar{e}_a\tilde{e}_b]}{E[\tilde{e}_b^2]}\,, \quad \gamma_a = \frac{E[\tilde{e}_b\bar{e}_a]}{E[\bar{e}_a^2]} \tag{1.8.54}$$

Using the Schwarz inequality for the inner product between two random variables, namely, $\big|E[uv]\big|^2 \leq E[u^2]E[v^2]$, we find the inequality

$$0 \leq \gamma_a\gamma_b = \frac{E[\bar{e}_a\tilde{e}_b]^2}{E[\tilde{e}_b^2]E[\bar{e}_a^2]} \leq 1 \tag{1.8.55}$$

This inequality also follows from Eq. (1.8.41) and the fact that E_a and $\bar{E}_a$ are positive quantities, both being mean square errors.

Example 1.8.4: For Example 1.8.1, compute the estimates $\hat{y}_{a/c}$ and $\hat{y}_{b/c}$ directly and compare them with the results of Example 1.8.3.

Solution: From the matrix elements of R we have $E[y_ay_b] = 1$, $E[y_by_c] = 2$, and $E[y_c^2] = 3$. Thus,

$$\hat{y}_{a/c} = E[y_ay_c]E[y_c^2]^{-1}y_c = \frac{1}{3}y_c\,, \quad \hat{y}_{b/c} = E[y_by_c]E[y_c^2]^{-1}y_c = \frac{2}{3}y_c$$

The corresponding errors will be

$$\bar{e}_a = y_a - \frac{1}{3}y_c = [1, -1/3]\bar{\mathbf{y}}\,, \quad \tilde{e}_b = y_b - \frac{2}{3}y_c = [-2/3, 1]\tilde{\mathbf{y}}$$

The results are identical to those of Example 1.8.3. □

Conventional Cholesky Factorizations

Equation (1.8.18) shows that the conventional Cholesky factor of R is given by the inverse matrix L^{-1}. A direct construction of the conventional Cholesky factor that avoids the computation of this inverse is as follows. Define

$$G_b = E[\mathbf{y}\mathbf{e}_b^T] \tag{1.8.56}$$

If we use $\mathbf{e}_b = L\mathbf{y}$ and $E[\mathbf{e}_b\mathbf{e}_b^T] = D_b$, it follows that

$$LG_b = LE[\mathbf{y}\mathbf{e}_b^T] = E[\mathbf{e}_b\mathbf{e}_b^T] = D_b$$

or,

$$G_b = L^{-1}D_b \tag{1.8.57}$$

Thus, G_b is a *lower-triangular* matrix. Its main diagonal consists of the diagonal entries of D_b. Solving for $L^{-1} = G_bD_b^{-1}$ and inserting in Eq. (1.8.18), we find the conventional LU factorization of R:

$$R = (G_bD_b^{-1})D_b(D_b^{-1}G_b^T) = G_bD_b^{-1}G_b^T \tag{1.8.58}$$

Similarly, the conventional UL factorization of R is obtained from Eq. (1.8.33) by defining the *upper-triangular* matrix

$$G_a = E[\mathbf{y}\mathbf{e}_a^T] \tag{1.8.59}$$

Using $\mathbf{e}_a = U\mathbf{y}$ and $E[\mathbf{e}_a\mathbf{e}_a^T] = D_a$, we find

$$UG_a = D_a \quad \Rightarrow \quad G_a = U^{-1}D_a \tag{1.8.60}$$

which yields the conventional UL factorization of R:

$$R = U^{-1}D_aU^{-T} = G_aD_a^{-1}G_a^T$$

The columns of the matrices G_a and G_b will be referred to as the forward and backward *gapped* functions. This terminology will be justified in Chap. 12. The decomposition of G_b into its columns can be done order-recursively using the decomposition (1.8.15). We have

$$G_b = E\big[\mathbf{y}[\bar{\mathbf{e}}_b^T, e_b]\big] \equiv [\bar{G}_b, \mathbf{g}_b] \tag{1.8.61}$$

where $\bar{G}_b = E[\mathbf{y}\bar{\mathbf{e}}_b^T]$ and $\mathbf{g}_b = E[\mathbf{y}e_b]$. Similarly, using Eq. (1.8.23) we find

$$G_a = E\big[\mathbf{y}[e_a, \tilde{\mathbf{e}}_a^T]\big] \equiv [\mathbf{g}_a, \tilde{G}_a] \tag{1.8.62}$$

where $\tilde{G}_a = E[\mathbf{y}\tilde{\mathbf{e}}_a^T]$ and $\mathbf{g}_a = E[\mathbf{y}e_a]$. Motivated by the lattice recursions (1.8.50), we are led to define the lower-order gapped functions

$$\tilde{\mathbf{g}}_b = E[\mathbf{y}\tilde{e}_b]\,, \quad \bar{\mathbf{g}}_a = E[\mathbf{y}\bar{e}_a]$$

It follows that the gapped functions $\mathbf{g}_a = E[\mathbf{y}e_a]$ and $\mathbf{g}_b = E[\mathbf{y}e_b]$ can be constructed order-recursively by the lattice-type equations

$$\begin{aligned}\mathbf{g}_a &= \bar{\mathbf{g}}_a - \gamma_b\tilde{\mathbf{g}}_b\\ \mathbf{g}_b &= \tilde{\mathbf{g}}_b - \gamma_a\bar{\mathbf{g}}_a\end{aligned} \tag{1.8.63}$$

The proof is straightforward. For example, $E[\mathbf{y}e_a] = E\big[\mathbf{y}(\bar{e}_a - \gamma_b\tilde{e}_b)\big]$. In Chap. 12 we will see that these equations are equivalent to the celebrated *Schur algorithm* for solving the linear prediction problem. In recent years, the Schur algorithm has emerged as an important signal processing tool because it admits efficient fixed-point and parallel processor implementations. Equations (1.8.63) are mathematically equivalent to the Levinson-type recursions (1.8.40). In fact, Eq. (1.8.40) can be derived from Eq. (1.8.63) as follows. Using $e_a = \mathbf{a}^T\mathbf{y}$ and $e_b = \mathbf{b}^T\mathbf{y}$, it follows that

$$\mathbf{g}_a = E[\mathbf{y}e_a] = E\big[\mathbf{y}(\mathbf{y}^T\mathbf{a})\big] = R\mathbf{a}\,, \quad \mathbf{g}_b = E[\mathbf{y}e_b] = E\big[\mathbf{y}(\mathbf{y}^T\mathbf{b})\big] = R\mathbf{b}$$

Similarly, we have

$$\bar{\mathbf{g}}_a = R\begin{bmatrix}\bar{\mathbf{a}}\\ 0\end{bmatrix}, \quad \tilde{\mathbf{g}}_b = R\begin{bmatrix}0\\ \tilde{\mathbf{b}}\end{bmatrix} \tag{1.8.64}$$

These are easily shown. For example,

$$R\begin{bmatrix}\bar{\mathbf{a}}\\ 0\end{bmatrix} = E\big[\mathbf{y}[\bar{\mathbf{y}}^T, y_b]\big]\begin{bmatrix}\bar{\mathbf{a}}\\ 0\end{bmatrix} = E[\mathbf{y}\bar{\mathbf{y}}^T]\bar{\mathbf{a}} = E[\mathbf{y}\bar{e}_a] = \bar{\mathbf{g}}_a$$

Therefore, the first part of Eq. (1.8.63) is equivalent to

$$R\mathbf{a} = R\begin{bmatrix}\bar{\mathbf{a}}\\ 0\end{bmatrix} - \gamma_bR\begin{bmatrix}0\\ \tilde{\mathbf{b}}\end{bmatrix}$$

Equation (1.8.40) follows now by canceling out the matrix factor R. One of the essential features of the Schur algorithm is that the reflection coefficients can also be computed from the knowledge of the lower-order gapped functions $\bar{\mathbf{g}}_a$ and $\tilde{\mathbf{g}}_b$, as follows. Using Eq. (1.8.64) and dotting Eq. (1.8.44) with the unit vectors $\mathbf{u}$ and $\mathbf{v}$, we find

$$\bar{E}_a = \mathbf{u}^T\bar{\mathbf{g}}_a\,, \quad \tilde{E}_b = \mathbf{v}^T\tilde{\mathbf{g}}_b\,, \quad \Delta = \mathbf{u}^T\tilde{\mathbf{g}}_b = \mathbf{v}^T\bar{\mathbf{g}}_a \tag{1.8.65}$$

Thus, Eq. (1.8.42) may be written as

$$\gamma_b = \frac{\mathbf{v}^T\bar{\mathbf{g}}_a}{\mathbf{v}^T\tilde{\mathbf{g}}_b}\,, \quad \gamma_b = \frac{\mathbf{u}^T\tilde{\mathbf{g}}_b}{\mathbf{u}^T\bar{\mathbf{g}}_a} \tag{1.8.66}$$

Summary

We have argued that the solution of the general linear estimation problem can be made more efficient by working with the decorrelated bases $\mathbf{e}_a$ or $\mathbf{e}_b$, which contain no redundancies. Linear prediction ideas come into play in this context because the linear transformations U and L that decorrelate the data vector $\mathbf{y}$ are constructible from the forward and backward linear prediction coefficients $\mathbf{a}$ and $\mathbf{b}$. Moreover, linear prediction was seen to be equivalent to the Gram-Schmidt construction and to the Cholesky factorization of the covariance matrix R. The order-recursive solutions of the linear prediction problem and the linear estimation problem, Eqs. (1.8.24) through (1.8.26), give rise to efficient lattice implementations with many desirable properties, such as robustness under coefficient quantization and modularity of structure admitting parallel VLSI implementations.

In this section, we intentionally did not make any additional assumptions about any structural properties of the covariance matrix R. To close the loop and obtain the efficient computational algorithms mentioned previously, we need to make additional assumptions on R. The simplest case is to assume that R has a Toeplitz structure. This case arises when $\mathbf{y}$ is a block of successive signal samples from a stationary time series. The Toeplitz property means that the matrix elements along each diagonal of R are the same. Equivalently, the matrix element R_{ij} depends only on the difference of the indices, that is, $R_{ij} = R(i-j)$. With respect to the subblock decomposition (1.8.5), it is easily verified that a necessary and sufficient condition for R to be Toeplitz is that

$$\tilde{R} = \bar{R}$$

This condition implies that the linear prediction solutions for $\tilde{R}$ and $\bar{R}$ must be the same, that is,

$$\tilde{\mathbf{b}} = \bar{\mathbf{b}}\,, \quad \tilde{\mathbf{a}} = \bar{\mathbf{a}}$$

Thus, from the forward and backward linear prediction solutions $\bar{\mathbf{a}}$ and $\bar{\mathbf{b}}$ of the lower-order Toeplitz submatrix $\bar{R}$, we first obtain $\tilde{\mathbf{b}} = \bar{\mathbf{b}}$ and then use Eq. (1.8.40) to get the linear prediction solution of the higher order matrix R. This is the essence behind Levinson's algorithm. It will be discussed further in Chap. 12.

In the nonstationary time series case, the matrix R is not Toeplitz. Even then one can obtain some useful results by means of the so-called *shift-invariance* property. In this case, the data vector $\mathbf{y}$ consists of successive signal samples starting at some arbitrary sampling instant n

$$\mathbf{y}(n)=\begin{bmatrix} y_n \\ y_{n-1} \\ \vdots \\ y_{n-M+1} \\ y_{n-M} \end{bmatrix} = \begin{bmatrix} \bar{\mathbf{y}}(n) \\ y_{n-M} \end{bmatrix} = \begin{bmatrix} y_n \\ \tilde{\mathbf{y}}(n) \end{bmatrix}$$

It follows that

$$\bar{\mathbf{y}}(n)=\begin{bmatrix} y_n \\ \vdots \\ y_{n-M+1} \end{bmatrix}, \quad \tilde{\mathbf{y}}(n)=\begin{bmatrix} y_{n-1} \\ \vdots \\ y_{n-M} \end{bmatrix}, \quad \text{or,} \quad \tilde{\mathbf{y}}(n)=\bar{\mathbf{y}}(n-1)$$

This implies that $\tilde{R}(n)=\bar{R}(n-1)$, and therefore

$$\tilde{\mathbf{a}}(n)=\bar{\mathbf{a}}(n-1)\,, \quad \tilde{\mathbf{b}}(n)=\bar{\mathbf{b}}(n-1)$$

Thus, order updating is coupled with time updating. These results are used in the development of the fast recursive least-squares adaptive filters, discussed in Chap. 16.

1.9 Random Signals

A random signal (random process, or stochastic process) is defined as a sequence of random variables $\{x_0, x_1, x_2, \dots, x_n, \dots\}$ where the index n is taken to be the time. The statistical description of so many random variables is very complicated since it requires knowledge of all the joint densities

$$p(x_0, x_1, x_2, \dots, x_n)\,, \quad \text{for } n = 0, 1, 2, \dots$$

If the mean $E[x_n]$ of the random signal is not zero, it can be removed by redefining a new signal $x_n - E[x_n]$. From now on, we will assume that this has been done, and shall work with zero-mean random signals. The *autocorrelation function* is defined as

$$R_{xx}(n,m)=E[x_n x_m]\,, \quad n,m = 0,1,2,\dots$$

Sometimes it will be convenient to think of the random signal as a (possibly infinite) random vector $\mathbf{x} = [x_0, x_1, x_2, \dots, x_n, \dots]^T$, and of the autocorrelation function as a (possibly infinite) matrix $R_{xx} = E[\mathbf{x}\mathbf{x}^T]$. R_{xx} is positive semi-definite and symmetric. The autocorrelation function may also be written as

$$R_{xx}(n+k, n)=E[x_{n+k}x_n] \tag{1.9.1}$$

It provides a *measure* of the influence of the sample x_n on the sample x_{n+k}, which lies in the future (if $k > 0$) by k units of time. The relative time separation k of the two samples is called the *lag*.

If the signal x_n is *stationary* (or wide-sense stationary), then the above average is independent of the absolute time n, and is a function only of the relative lag k; abusing somewhat the above notation, we may write in the case:

$$R_{xx}(k)=E[x_{n+k}x_n]=E[x_{n'+k}x_{n'}] \qquad \text{(autocorrelation)} \tag{1.9.2}$$

In other words, the self-correlation properties of a stationary signal x_n are same on the average, regardless of when this average is computed. In a way, the stationary random signal x_n looks the same for all times. In this sense, if we take two different blocks of data of length N, as shown in Fig. 1.9.1, we should expect the *average properties*, such as means and autocorrelations, extracted from these blocks of data to be roughly the same. The relative time separation of the two blocks as a whole should not matter.

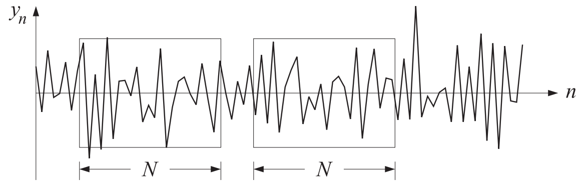


Fig. 1.9.1 Blocks of data from a stationary signal.

A direct consequence of stationarity is the *reflection-invariance* of the autocorrelation function $R_{xx}(k)$ of Eq. (1.9.2):

$$R_{xx}(k)=E[x_{n+k}x_n]=R_{xx}(-k) \tag{1.9.3}$$

One way to introduce a systematization of the various types of random signals is the Markov classification into zeroth-order Markov, first-order Markov, and so on. The simplest possible random signal is the zeroth-order Markov, or *purely random* signal, defined by the requirement that all the (zero-mean) random variables x_n be independent of each other and arise from a common density $p(x)$; this implies

$$p(x_0, x_1, x_2, \dots, x_n)=p(x_0)p(x_1)p(x_2)\cdots p(x_n)\cdots$$

$$R_{xx}(n,m)=E[x_n x_m]=0\,, \quad \text{for} \quad n \neq m$$

Such a random signal is stationary. The quantity $R_{xx}(n,n)$ is independent of n, and represents the variance of each sample:

$$R_{xx}(0)=E[x_n^2]=\sigma_x^2$$

In this case, the autocorrelation function $R_{xx}(k)$ may be expressed compactly as

$$R_{xx}(k)=E[x_{n+k}x_n]=\sigma_x^2\delta(k) \tag{1.9.4}$$

A purely random signal has no memory, as can be seen from the property

$$p(x_n, x_{n-1})=p(x_n)p(x_{n-1}) \quad \text{or,} \quad p(x_n|x_{n-1})=p(x_n)$$

that is, the occurrence of x_{n-1} at time instant $n-1$ does not in any way affect, or restrict, the values of x_n at the next time instant. Successive signal values are entirely independent of each other. Past values do not influence future values. No memory is retained from sample to sample; the next sample will take a value regardless of the value that the previous sample has already taken. Since successive samples are random, such a signal will exhibit very rapid time variations. But it will also exhibit slow time variations. Such time variations are best discussed in the frequency domain. This will lead directly to frequency concepts, power spectra, periodograms, and the like. It is expected that a purely random signal will contain all frequencies, from the very low to the very high, in equal proportions (white noise).

The next least complicated signal is the first-order Markov signal, which has memory only of one sampling instant. Such a signal remembers only the previous sample. It is defined by the requirement that

$$p(x_n|x_{n-1},x_{n-1},\dots,x_0)=p(x_n|x_{n-1})$$

which states that x_n may be influenced directly only by the previous sample value x_{n-1}, and not by the samples $x_{n-2},\dots,x_0$ that are further in the past. The complete statistical description of such random signal is considerably simplified. It is sufficient to know only the marginal densities $p(x_n)$ and the conditional densities $p(x_n|x_{n-1})$. Any other joint density may be constructed in terms of these. For instance,

$$\begin{aligned}
p(x_3,x_2,x_1,x_0) &= p(x_3|x_2,x_1,x_0)p(x_2,x_1,x_0) && \text{(by Bayes' rule)}\\
&= p(x_3|x_2)p(x_2,x_1,x_0) && \text{(by the Markov property)}\\
&= p(x_3|x_2)p(x_2|x_1,x_0)p(x_1,x_0)\\
&= p(x_3|x_2)p(x_2|x_1)p(x_1,x_0)\\
&= p(x_3|x_2)p(x_2|x_1)p(x_1|x_0)p(x_0)
\end{aligned}$$

1.10 Power Spectrum and Its Interpretation

The *power spectral density* of a stationary random signal x_n is defined as the double-sided z-transform of its autocorrelation function

$$S_{xx}(z)=\sum_{k=-\infty}^{\infty}R_{xx}(k)z^{-k} \tag{1.10.1}$$

where $R_{xx}(k)$is given by Eq. (1.9.2). If $R_{xx}(k)$ is strictly stable, the region of convergence of $S_{xx}(z)$ will include the unit circle in the complex z-plane. This allows us to define the *power spectrum* $S_{xx}(\omega)$ of the random signal x_n by setting $z=e^{j\omega}$ in Eq. (1.10.1). Abusing the notation somewhat, we have in this case

$$S_{xx}(\omega)=\sum_{k=-\infty}^{\infty}R_{xx}(k)e^{-j\omega k} \tag{1.10.2}$$

This quantity conveys very useful information. It is a measure of the frequency content of the signal x_n and of the distribution of the power of x_n over frequency. To see this, consider the inverse z-transform

$$R_{xx}(k)=\oint_{\text{u.c.}}S_{xx}(z)z^k\frac{dz}{2\pi jz} \tag{1.10.3}$$

where, since $R_{xx}(k)$ is stable, the integration contour may be taken to be the unit circle. Using $z=e^{j\omega}$, we find for the integration measure

$$\frac{dz}{2\pi jz}=\frac{d\omega}{2\pi}$$

Thus, Eq. (1.10.3) may also be written as an inverse Fourier transform

$$R_{xx}(k)=\int_{-\pi}^{\pi}S_{xx}(\omega)e^{j\omega k}\frac{d\omega}{2\pi} \tag{1.10.4}$$

In particular, the variance of x_n can be written as

$$R_{xx}(0)=\sigma_x^2=E[x_n^2]=\int_{-\pi}^{\pi}S_{xx}(\omega)\,\frac{d\omega}{2\pi} \tag{1.10.5}$$

Since the quantity $E[x_n^2]$ represents the average *total power* contained in x_n, it follows that $S_{xx}(\omega)$ will represent the *power per unit frequency interval.* A typical power spectrum is depicted in Fig. 1.10.1. As suggested by this figure, it is possible for the power to be mostly concentrated about some frequencies and not about others. The area under the curve represents the total power of the signal x_n.

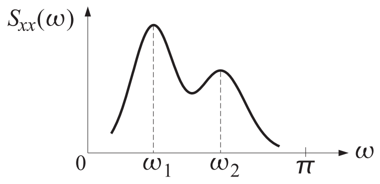
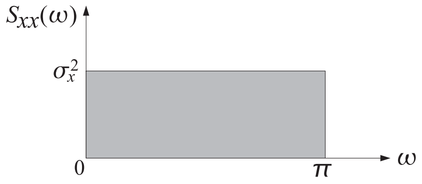


Fig. 1.10.1 Typical power spectrum.

If x_n is an uncorrelated (white-noise) random signal with a delta-function autocorrelation, given by Eq. (1.9.4), it will have a *flat* power spectrum with power level equal to the variance σ_x^2:

$$S_{xx}(\omega)=\sigma_x^2$$



Another useful concept is that of the *cross-correlation* and *cross-spectrum* between two stationary random sequences x_n and y_n. These are defined by

$$R_{yx}(k)=E[y_{n+k}x_n]\,,\quad S_{yx}(z)=\sum_{k=-\infty}^{\infty}R_{yx}(k)z^{-k} \tag{1.10.6}$$

Using stationarity, it is easy to show the reflection symmetry property

$$R_{yx}(k)=R_{xy}(-k) \tag{1.10.7}$$

that is analogous to Eq. (1.9.3). In the z-domain, the reflection symmetry properties (1.9.3) and (1.10.7) are translated into:

$$S_{xx}(z)=S_{xx}(z^{-1})\,,\quad S_{yx}(z)=S_{xy}(z^{-1}) \tag{1.10.8}$$

respectively; and also

$$S_{xx}(\omega)=S_{xx}(-\omega)\,,\quad S_{yx}(\omega)=S_{xy}(-\omega) \tag{1.10.9}$$

1.11 *Sample Autocorrelation and the Periodogram*

From now on we will work mostly with stationary random signals. If a block of N signal samples is available, we will assume that it is a segment from a stationary signal. The length N of the available data segment is an important consideration. For example, in computing frequency spectra, we know that high resolution in frequency requires a long record of data. However, if the record is too long the assumption of stationarity may no longer be justified. This is the case in many applications, as for example in speech and EEG signal processing. The speech waveform does not remain stationary for long time intervals. It may be assumed to be stationary only for short time intervals. Such a signal may be called *piece-wise stationary*. If it is divided into short segments of duration of approximately 20–30 milliseconds, then the portion of speech within each segment may be assumed to be a segment from a stationary signal. A typical piece-wise stationary signal is depicted in Fig. 1.11.1.

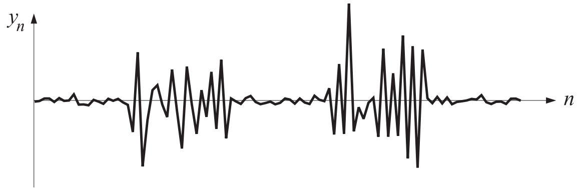


Fig. 1.11.1 Piece-wise stationary signal.

The main reason for assuming stationarity, or piece-wise stationarity, is that most of our methods of handling random signals depend heavily on this assumption. For example, the statistical autocorrelations based on the ensemble averages (1.9.2) may be replaced in practice by *time averages*. This can be justified only if the signals are stationary (actually, they must be ergodic). If the underlying signal processes are not stationary (and therefore definitely are not ergodic) we cannot use time averages. If a signal is piece-wise stationary and divided into stationary blocks, then for each such block, ensemble averages may be replaced by time averages. The time average approximation of an autocorrelation function is called the *sample autocorrelation* and is defined

as follows: Given a block of length N of measured signal samples

$$\boxed{y_0, y_1, y_2, \dots, y_{N-1}}$$

define

$$\hat{R}_{yy}(k)=\frac{1}{N}\sum_{n=0}^{N-1-k} y_{n+k}y_n\,,\quad \text{for}\quad 0\le k\le N-1 \tag{1.11.1}$$

and

$$\hat{R}_{yy}(k)=\hat{R}_{yy}(-k)\,,\quad \text{for}\quad -(N-1)\le k\le -1$$

The function **acf** takes as inputs two length-N signal blocks y_n, x_n, $n=0,1,\dots,N-1$, and computes their sample cross-correlation defined as

$$\hat{R}_{yx}(k)=\frac{1}{N}\sum_{k=0}^{N-1-k} y_{n+k}x_n\,,\quad k=0,1,\dots,N-1$$

This function may be used to compute either auto-correlations or cross-correlations. The *periodogram* is defined as the (double-sided) z-transform of the sample autocorrelation

$$\hat{S}_{yy}(z)=\sum_{k=-(N-1)}^{N-1}\hat{R}_{yy}(k)z^{-k} \tag{1.11.2}$$

It may be thought of as an approximation (estimate) of the true power spectral density $S_{yy}(z)$. It is easily shown that the periodogram may be expressed in terms of the z-transform of the data sequence itself, as

$$\hat{S}_{yy}(z)=\frac{1}{N}Y(z)Y(z^{-1}) \tag{1.11.3}$$

where

$$Y(z)=\sum_{n=0}^{N-1}y_n z^{-n} \tag{1.11.4}$$

As a concrete example, consider a length-3 signal $\mathbf{y}=[y_0,y_1,y_2]^T$. Then,

$$\begin{aligned}Y(z)Y(z^{-1}) &= (y_0+y_1z^{-1}+y_2z^{-2})(y_0+y_1z+y_2z^2)\\ &= (y_0^2+y_1^2+y_2^2)+(y_0y_1+y_1y_2)(z^{-1}+z)+(y_0y_2)(z^{-2}+z^2)\end{aligned}$$

from which we extract the inverse z-transform

$$\begin{aligned}\hat{R}_{xx}(0) &= \frac{1}{3}(y_0^2+y_1^2+y_2^2)\\ \hat{R}_{xx}(-1)=\hat{R}_{xx}(1) &= \frac{1}{3}(y_0y_1+y_1y_2)\\ \hat{R}_{xx}(-2)=\hat{R}_{xx}(2) &= \frac{1}{3}(y_0y_2)\end{aligned}$$

These equations may also be written in a nice matrix form, as follows

$$\underbrace{\begin{bmatrix} \hat{R}_{xx}(0) & \hat{R}_{xx}(1) & \hat{R}_{xx}(2) \\ \hat{R}_{xx}(1) & \hat{R}_{xx}(0) & \hat{R}_{xx}(1) \\ \hat{R}_{xx}(2) & \hat{R}_{xx}(1) & \hat{R}_{xx}(0) \end{bmatrix}}_{\hat{R}_{yy}} = \frac{1}{3}\underbrace{\begin{bmatrix} y_0 & y_1 & y_2 & 0 & 0 \\ 0 & y_0 & y_1 & y_2 & 0 \\ 0 & 0 & y_0 & y_1 & y_2 \end{bmatrix}}_{Y^T} \underbrace{\begin{bmatrix} y_0 & 0 & 0 \\ y_1 & y_0 & 0 \\ y_2 & y_1 & y_0 \\ 0 & y_2 & y_1 \\ 0 & 0 & y_2 \end{bmatrix}}_{Y}$$

or,

$$\hat{R}_{yy} = \frac{1}{3}Y^T Y$$

The matrix $\hat{R}_{yy}$ on the left is called the sample autocorrelation matrix. It is a *Toeplitz matrix*, that is, it has the same entry in each diagonal. The right hand side also shows that the autocorrelation matrix is positive definite. In the general case of a length-N sequence y_n, the matrix Y has N columns, each a down-shifted (delayed) version of the previous one, corresponding to a total of $N-1$ delays. This requires the length of each column to be $N+(N-1)$, that is, there are $2N-1$ rows. We will encounter again this matrix factorization in the least-squares design of waveshaping filters.

The sample autocorrelation may also be thought of as *ordinary convolution*. Note that $Y(z^{-1})$ represents the z-transform the original signal $\mathbf{y} = [y_0, y_1, \dots, y_{N-1}]^T$ reflected about the time origin. The reflected signal may be made causal by a delay of $N-1$ units of time. The reflected-delayed signal has some significance, and is known as the *reversed signal*. Its z-transform is the *reverse polynomial* of $Y(z)$

$$Y^R(z) = z^{-(N-1)}Y(z^{-1})$$

$$\begin{array}{lccccccccclcl} [& 0 & 0 & \cdots & 0 & y_0 & y_1 & \cdots & y_{N-2} & y_{N-1} &] & = & \text{original} \\ [& y_{N-1} & y_{N-2} & \cdots & y_1 & y_0 & 0 & \cdots & 0 & 0 &] & = & \text{reflected} \\ [& 0 & 0 & \cdots & 0 & y_{N-1} & y_{N-2} & \cdots & y_1 & y_0 &] & = & \text{reversed} \end{array}$$

The periodogram is expressed then in the form

$$\hat{S}_{xx}(z) = \frac{1}{N}Y(z)Y(z^{-1}) = \frac{1}{N}Y(z)Y^R(z)z^{N-1}$$

which implies that $\hat{R}_{yy}(k)$ may be obtained by convolving the original data sequence with the reversed sequence and then advancing the result in time by $N-1$ time units. This is seen by the following convolution table.

	y_0	y_1	y_2
y_2	y_2y_0	y_2y_1	y_2y_2
y_1	y_1y_0	y_1y_1	y_1y_2
y_0	y_0y_0	y_0y_1	y_0y_2

The *periodogram spectrum* is obtained by substituting $z = e^{j\omega}$

$$\hat{S}_{yy}(\omega) = \frac{1}{N}\left|Y(\omega)\right|^2 = \frac{1}{N}\left|\sum_{n=0}^{N-1} y_n e^{-j\omega n}\right|^2 \tag{1.11.5}$$

The periodogram spectrum (1.11.5) may be computed efficiently using FFT methods. The digital frequency ω in units of [radians/sample] is related to the physical frequency f in [Hz] by

$$\omega = 2\pi f T = \frac{2\pi f}{f_s}$$

where f_s is the sampling rate, and $T = 1/f_s$, the time interval between samples. The frequency resolution afforded by a length-N sequence is

$$\Delta\omega = \frac{2\pi}{N}, \quad \text{or,} \quad \Delta f = \frac{f_s}{N} = \frac{1}{NT} = \frac{1}{T_R} \quad \text{[Hz]}$$

where $T_R = NT$ is the duration of the data record in seconds. The periodogram spectrum suffers from two major drawbacks. First, the rectangular windowing of the data segment introduces significant *sidelobe leakage*. This can cause misinterpretation of sidelobe spectral peaks as being part of the true spectrum. And second, it is well-known that the periodogram spectrum is *not* a good (consistent) estimator of the true power spectrum $S_{yy}(\omega)$.

The development of methods to improve on the periodogram is the subject of *classical spectral analysis* [9–19]. We just mention, in passing, one of the most popular of such methods, namely, Welch's method [20]. The given data record of length N is subdivided into K shorter segments which may be overlapping or non-overlapping. If they are non-overlapping then each will have length $M = N/K$; if they are 50% overlapping then $M = 2N/(K+1)$. Each such segment is then windowed by a length-M data window, such as a Hamming window. The window reduces the sidelobe frequency leakage at the expense of resolution. The window $w(n)$ is typically normalized to have unit average energy, that is, $(1/M)\sum_{n=0}^{M-1} w^2(n) = 1$. The periodogram of each windowed segment is then computed by FFT methods and the K periodograms are averaged together to obtain the spectrum estimate

$$S(\omega) = \frac{1}{K}\sum_{i=1}^{K} S_i(\omega)$$

where $S_i(\omega)$ is the periodogram of the ith segment. The above subdivision into segments imitates ensemble averaging, and therefore, it results in a spectrum estimate of improved statistical stability. However, since each periodogram is computed from a length-M sequence, the frequency resolution is reduced from $\Delta\omega = 2\pi/N$ to roughly $\Delta\omega = 2\pi/M$ (for a well-designed window). Therefore, to maintain high frequency resolution (large M), as well as improved statistical stability of the spectrum estimate (large K), a long data record $N = MK$ is required—a condition that can easily come into conflict with stationarity. The so-called "modern methods" of spectrum estimation, which are based on parametric signal models, can provide high resolution spectrum estimates from short data records.

1.12 Filtering of Stationary Random Signals

In this section, we discuss the effect of linear filtering on random signals. The results are very basic and useful in suggesting guidelines for the design of signal processing systems for many applications, such as noise reduction, signal extraction, parametric spectrum estimation, and so on.

Suppose a stationary random signal x_n is sent into a linear filter defined by a transfer function $H(z)$, resulting in the the output random signal y_n

$$x_n \longrightarrow \boxed{H(z)} \longrightarrow y_n \qquad H(z)=\sum_n h_n z^{-n}$$

We would like to derive relationships between the autocorrelation functions of the input and output signals, and also between the corresponding power spectra. We assume, for now, that the signals x_n, y_n, h_n are real-valued. Using the input/output filtering equation in the z-domain,

$$\boxed{Y(z)=H(z)X(z)} \tag{1.12.1}$$

we determine first a relationship between the *periodograms* of the input and output signals. From the factorization of Eq. (1.11.3) and dropping the factor $1/N$ for convenience, we find

$$\begin{aligned}\hat{S}_{yy}(z) &= Y(z)Y(z^{-1}) \\ &= H(z)X(z)H(z^{-1})X(z^{-1}) = H(z)H(z^{-1})X(z)X(z^{-1}) \\ &= H(z)H(z^{-1})\hat{S}_{xx}(z) = S_{hh}(z)\hat{S}_{xx}(z)\end{aligned} \tag{1.12.2}$$

where we used the notation $S_{hh}(z)=H(z)H(z^{-1})$. This quantity is the z-transform of the *autocorrelation function* of the filter, that is,

$$\boxed{S_{hh}(z)=H(z)H(z^{-1})=\sum_{k=-\infty}^{\infty} R_{hh}(k)z^{-k}} \tag{1.12.3}$$

where $R_{hh}(k)$ is defined as

$$\boxed{R_{hh}(k)=\sum_n h_{n+k}h_n} \quad \text{(filter autocorrelation function)} \tag{1.12.4}$$

Equation (1.12.3) is easily verified by writing,

$$R_{hh}(k)=\sum_{i,j} h_i h_j \delta\big(k-(i-j)\big)$$

and taking z-transforms, or by writing $R_{hh}(k)=\sum_n h_{k+n}h_n = \sum_n h_{k-n}h_{-n}$, which is recognized as the convolution between the signals h_n and h_{-n} whose z-transforms are $H(z)$ and $H(z^{-1})$, respectively.

Taking inverse z-transforms of Eq. (1.12.2), we obtain the time-domain equivalent relationships between the input and output sample autocorrelations

$$\hat{R}_{yy}(k)=\sum_{m=-\infty}^{\infty} R_{hh}(k)\hat{R}_{xx}(k-m) = \text{convolution of } R_{hh} \text{ with } \hat{R}_{xx} \tag{1.12.5}$$

Similarly, we find for the cross-periodograms

$$\hat{S}_{yx}(z)=Y(z)X(z^{-1})=H(z)X(z)X(z^{-1})=H(z)\hat{S}_{xx}(z) \tag{1.12.6}$$

and also, replacing z by z^{-1},

$$\hat{S}_{xy}(z)=\hat{S}_{xx}(z)H(z^{-1}) \tag{1.12.7}$$

The above relationships between input and output periodogram spectra and sample autocorrelations remain the same for the *statistical* autocorrelations and power spectra. In the z-domain the power spectral densities are related by

$$\boxed{\begin{aligned}S_{yy}(z) &= H(z)H(z^{-1})S_{xx}(z) \\ S_{yx}(z) &= H(z)S_{xx}(z) \\ S_{xy}(z) &= S_{xx}(z)H(z^{-1})\end{aligned}} \tag{1.12.8}$$

Setting $z=e^{j\omega}$, we may also write Eq. (1.12.8) in terms of the corresponding power spectra:

$$\boxed{\begin{aligned}S_{yy}(\omega) &= |H(\omega)|^2 S_{xx}(\omega) \\ S_{yx}(\omega) &= H(\omega)S_{xx}(\omega) \\ S_{xy}(\omega) &= S_{xx}(\omega)H(-\omega)=S_{xx}(\omega)H(\omega)^*\end{aligned}} \tag{1.12.9}$$

In the time domain the correlation functions are related by

$$\boxed{\begin{aligned}R_{yy}(k) &= \sum_{m=-\infty}^{\infty} R_{hh}(m)R_{xx}(k-m) \\ R_{yx}(k) &= \sum_{m=-\infty}^{\infty} h_m R_{xx}(k-m)\end{aligned}} \tag{1.12.10}$$

The proofs of these are straightforward. For example, to show Eq. (1.12.10), we may use stationarity and the I/O convolutional equation,

$$y_n = \sum_m h_m x_{n-m}$$

to find

$$\begin{aligned}R_{yy}(k) &= E[y_{n+k}y_n] = E\left[\sum_i h_i x_{n+k-i} \sum_j h_j x_{n-j}\right] \\ &= \sum_{i,j} h_i h_j E[x_{n+k-i}x_{n-j}] = \sum_{i,j} h_i h_j R_{xx}\big(k-(i-j)\big) \\ &= \sum_{i,j,m} h_i h_j \delta\big(m-(i-j)\big)R_{xx}(k-m) = \sum_m R_{hh}(m)R_{xx}(k-m)\end{aligned}$$

The proof assumes that the transients introduced by the filter have died out and that the output signal is stationary. For a strictly stable filter, the stationarity of the output y_n (i.e., the fact that $E[y_{n+k}y_n]$ is independent of the absolute time n), becomes valid for large times n. To see the effect of such transients, consider a causal filter and assume that the input x_n is applied starting at $n = 0$. Then, the I/O equation reads:

$$y_n = \sum_{m=0}^{n} h_m x_{n-m}$$

and the corresponding output autocorrelation function becomes (for $n, k \geq 0$):

$$E[y_{n+k}y_n] = E\left[\sum_{i=0}^{n+k} h_i x_{n+k-i} \sum_{j=0}^{n} h_j x_{n-j}\right] = \sum_{i=0}^{n+k}\sum_{j=0}^{n} h_i h_j R_{xx}(k+j-i)$$

which does have an explicit n dependence. Assuming that the filter is strictly stable, the above expression will converge to Eq. (1.12.10) in the limit of large n. A further example of this property is discussed in Sec. 1.15.

The above filtering results can be applied to the special case of a zero-mean white-noise signal x_n of variance σ_x^2, which has a delta-function autocorrelation and a flat power spectrum, as shown in Fig. 1.12.1:

$$R_{xx}(k) = E[x_{n+k}x_n] = \sigma_x^2\delta(k)\,, \quad S_{xx}(z) = \sigma_x^2 \tag{1.12.11}$$

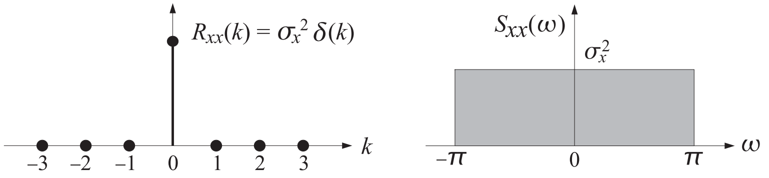


Fig. 1.12.1 Autocorrelation function and power spectrum of white noise.

Then, Eqs. (1.12.8) through (1.12.10) simplify as follows

$$\boxed{\begin{aligned} S_{yy}(z) &= H(z)H(z^{-1})\sigma_x^2 \\ S_{yx}(z) &= H(z)\sigma_x^2 \end{aligned}} \tag{1.12.12}$$

$$\boxed{\begin{aligned} S_{yy}(\omega) &= |H(\omega)|^2\sigma_x^2 \\ S_{yx}(\omega) &= H(\omega)\sigma_x^2 \end{aligned}} \tag{1.12.13}$$

$$\boxed{\begin{aligned} R_{yy}(k) &= \sigma_x^2 \sum_n h_{n+k}h_n = \sigma_x^2 R_{hh}(k) \\ R_{yx}(k) &= \sigma_x^2\, h_k \end{aligned}} \tag{1.12.14}$$

The filtering operation reshapes the flat white-noise spectrum of the input signal into a shape defined by the magnitude response $|H(\omega)|^2$ of the filter, and introduces self-correlations in the output signal given by the autocorrelation of the filter. The variance σ_y^2 of the output noise signal y_n is obtained from Eq. (1.10.5), that is,

$$\sigma_y^2 = E[y_n^2] = R_{yy}(0) = \frac{1}{2\pi}\int_{-\pi}^{\pi} S_{yy}(\omega)\,d\omega = \frac{1}{2\pi}\int_{-\pi}^{\pi}|H(\omega)|^2\sigma_x^2\,d\omega \tag{1.12.15}$$

The ratio σ_y^2/σ_x^2 is a measure of whether the filter attenuates or amplifies the input noise. We will refer to it as the *noise reduction ratio* (NRR). Using Eq. (1.12.15) and Parseval's identity, we may express it in the equivalent forms:

$$\boxed{\mathcal{R} = \frac{\sigma_y^2}{\sigma_x^2} = \sum_n h_n^2 = \frac{1}{2\pi}\int_{-\pi}^{\pi}|H(\omega)|^2\,d\omega} \quad \text{(noise reduction ratio)} \tag{1.12.16}$$

Example 1.12.1: As an example, consider the first-order Markov signal y_n defined as the output of the filter

$$y_n = ay_{n-1} + \epsilon_n\,, \quad H(z) = \frac{1}{1 - az^{-1}}\,, \quad |a| < 1$$

driven by white noise ϵ_n of variance σ_ϵ^2. The impulse response of the filter is

$$h_n = a^n u(n)\,, \quad u(n) = \text{unit step}$$

The output autocorrelation $R_{yy}(k)$ may be computed in two ways. First, in the time domain (assuming first that $k \geq 0$):

$$R_{yy}(k) = \sigma_\epsilon^2 \sum_{n=0}^{\infty} h_{n+k}h_n = \sigma_\epsilon^2 \sum_{n=0}^{\infty} a^{n+k}a^n = \sigma_\epsilon^2\, a^k \sum_{n=0}^{\infty} a^{2n} = \frac{\sigma_\epsilon^2 a^k}{1 - a^2}$$

And second, in the z-domain using power spectral densities and inverse z-transforms (again take $k \geq 0$):

$$\begin{aligned} S_{yy}(z) &= H(z)H(z^{-1})\sigma_\epsilon^2 = \frac{\sigma_\epsilon^2}{(1 - az^{-1})(1 - az)} \\ R_{yy}(k) &= \oint_{\text{u.c}} S_{yy}(z)z^k\frac{dz}{2\pi jz} = \oint_{\text{u.c.}} \frac{\sigma_\epsilon^2 z^k}{(z-a)(1-az)}\frac{dz}{2\pi j} \\ &= (\text{Residue at } z = a) = \frac{\sigma_\epsilon^2 a^k}{1 - a^2} \end{aligned}$$

In particular, we verify the following results to be used later:

$$\begin{aligned} R_{yy}(0) &= \frac{\sigma_\epsilon^2}{1 - a^2}\,, \quad R_{yy}(1) = \frac{\sigma_\epsilon^2 a}{1 - a^2} = aR_{yy}(0) \\ a &= \frac{R_{yy}(1)}{R_{yy}(0)}\,, \quad \sigma_\epsilon^2 = (1 - a^2)R_{yy}(0) \end{aligned}$$

It is interesting to note the exponentially decaying nature of $R_{yy}(k)$ with increasing lag k, as shown in Fig. 1.12.2.

Correlations exist between successive samples due the indirect influence of a given sample y_n on all future samples, as propagated by the difference equation. In going from one sampling instant to the next, the difference equation scales y_n by a factor a; therefore, we expect the correlations to decrease exponentially with increasing lag. □

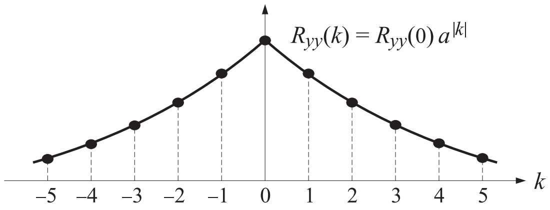


Fig. 1.12.2 Exponentially decaying autocorrelation.

Whenever the autocorrelation drops off very fast with increasing lag, it can be taken as an indication that there exists a stable difference equation model for the random signal. However, not all random signals have exponentially decaying autocorrelations. For example, a pure sinusoid with random phase

$$y_n = A\cos(\omega_0 n + \phi)$$

where ϕ is a uniformly-distributed random phase, has autocorrelation

$$R_{yy}(k) = \frac{1}{2}A^2\cos(\omega_0 k)$$

which never dies out. A particular realization of the random variable ϕ defines the entire realization of the time series y_n. Thus, as soon as ϕ is fixed, the entire y_n is fixed. Such random signals are called *deterministic*, since a few past values—e.g., three samples—of y_n are sufficient to determine all future values of y_n.

Finally we note that all of the filtering equations in Eqs. (1.12.8)–(1.12.10) can be considered to be special cases of the following more general result involving two filters $H_1(z)$ and $H_2(z)$ and two stationary input random signals $x_1(n)$ and $x_2(n)$, resulting in the output signals $y_1(n)$ and $y_2(n)$ as shown below:

$$x_1(n) \longrightarrow \boxed{H_1(z)} \longrightarrow y_1(n)$$
$$x_2(n) \longrightarrow \boxed{H_2(z)} \longrightarrow y_2(n)$$

Then, the corresponding cross-power spectral density of the output signals is given by:

$$\boxed{S_{y_1y_2}(z) = H_1(z)H_2(z^{-1})S_{x_1x_2}(z)} \tag{1.12.17}$$

where $S_{x_1x_2}(z)$ is the z-transform of $R_{x_1x_2}(k) = E\big[x_1(n+k)\,x_2(n)\big]$, etc.

1.13 Random Signal Models and Their Uses

Models that provide a characterization of the properties and nature of random signals are of primary importance in the design of optimum signal processing systems. This section offers an overview of such models and outlines their major applications. Many of the ideas presented here will be developed in greater detail in later chapters.

One of the most useful ways to model a random signal [21] is to consider it as being the *output* of a *causal and stable* linear filter $B(z)$ that is driven by a *stationary uncorrelated* (white-noise) sequence ϵ_n,

$$\epsilon_n \longrightarrow \boxed{B(z)} \longrightarrow y_n \qquad B(z) = \sum_{n=0}^{\infty} b_n z^{-n}$$

where $R_{\epsilon\epsilon}(k) = E[\epsilon_{n+k}\epsilon_n] = \sigma_\epsilon^2\delta(k)$. Assuming a causal input sequence ϵ_n, the output random signal y_n is obtained by convolving ϵ_n with the filter's impulse response b_n:

$$y_n = \sum_{i=0}^{n} b_{n-i}\epsilon_i \tag{1.13.1}$$

The stability of the filter $B(z)$ is essential as it guarantees the stationarity of the sequence y_n. This point will be discussed later on. By readjusting, if necessary, the value of σ_ϵ^2 we may assume that $b_0 = 1$. Then Eq. (1.13.1) corresponds exactly to the Gram-Schmidt form of Eqs. (1.6.15) and (1.6.16), where the matrix elements b_{ni} are given in terms of the impulse response of the filter $B(z)$:

$$b_{ni} = b_{n-i} \tag{1.13.2}$$

In this case, the structure of the matrix B is considerably simplified. Writing the convolutional equation (1.13.1) in matrix form

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ b_1 & 1 & 0 & 0 & 0 \\ b_2 & b_1 & 1 & 0 & 0 \\ b_3 & b_2 & b_1 & 1 & 0 \\ b_4 & b_3 & b_2 & b_1 & 1 \end{bmatrix} \begin{bmatrix} \epsilon_0 \\ \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \epsilon_4 \end{bmatrix} \tag{1.13.3}$$

we observe that the first column of B is the impulse response b_n of the filter. Each subsequent column is a down-shifted (delayed) version of the previous one, and each diagonal has the same entry (i.e., B is a Toeplitz matrix). The lower-triangular nature of B is equivalent to the assumed *causality* of the filter $B(z)$.

Such signal models are quite general. In fact, there is a general theorem by Wold that essentially guarantees the *existence* of such models for any stationary signal y_n [22,23]. Wold's construction of $B(z)$ is none other than the Gram-Schmidt construction of the orthogonalized basis ϵ_n. However, the practical usage of such models requires further that the transfer function $B(z)$ be *rational*, that is, the ratio of two polynomials in z^{-1}. In this case, the I/O convolutional equation (1.13.1) is most conveniently expressed as a *difference equation*.

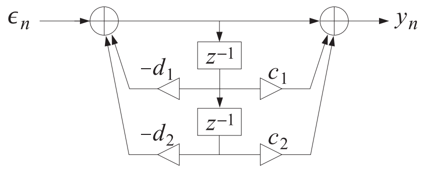
Example 1.13.1: Suppose

$$B(z) = \frac{1 + c_1 z^{-1} + c_2 z^{-2}}{1 + d_1 z^{-1} + d_2 z^{-2}} \tag{1.13.4}$$

Then Eq. (1.13.1) is equivalent to the difference equation

$$y_n = -d_1 y_{n-1} - d_2 y_{n-2} + \epsilon_n + c_1\epsilon_{n-1} + c_2\epsilon_{n-2} \tag{1.13.5}$$

which may be realized as follows



The filter $B(z)$ is called a *synthesis filter* and may be thought of as a random signal generator, or a signal model, for the random signal y_n. The numerator and denominator coefficients of the filter $B(z)$, and the variance σ_ϵ^2 of the input white noise, are referred to as the *model parameters*. For instance, in Example 1.13.1 the model parameters are $\{c_1, c_2, d_1, d_2, \sigma_\epsilon^2\}$.

Such parametric models have received a lot of attention in recent years. They are very common in speech and geophysical signal processing, image processing, EEG signal processing, spectrum estimation, data compression, and other time series analysis applications.

How are such models used? One of the main objectives in such applications has been to develop appropriate *analysis procedures* for extracting the model parameters on the basis of a given set of samples of the signal y_n. This is a *system identification* problem. The analysis procedures are designed to provide effectively the *best fit* of the data samples to a particular model. The procedures typically begin with a measured block of signal samples $\{y_0, y_1, \dots, y_N\}$—also referred to as an *analysis frame*—and through an appropriate analysis algorithm extract *estimates* of the model parameters. This is depicted in Fig. 1.13.1.

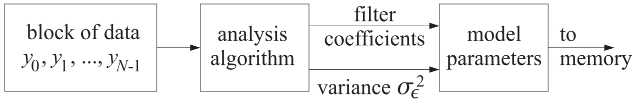


Fig. 1.13.1 Analysis procedure.

The given frame of samples $\{y_0, y_1, \dots, y_N\}$ is *represented* now by the set of model parameters extracted from it. Following the analysis procedure, the resulting model may be used in a variety of ways. The four major uses of such models are in:

1. Signal synthesis
2. Spectrum estimation
3. Signal classification
4. Data compression

We will discuss each of these briefly. To synthesize a particular realization of the random signal y_n, it is only necessary to recall from memory the appropriate model parameters, generate a random uncorrelated sequence ϵ_n having variance σ_ϵ^2, and send it through the filter $B(z)$. Such uncorrelated sequence may be *computer-generated* using a standard random number generator function. The synthetic signal will appear at the output of the filter. This is shown in Fig. 1.13.2.

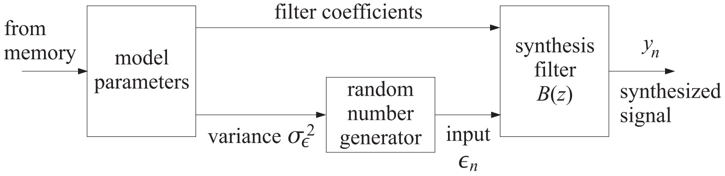


Fig. 1.13.2 Signal synthesis.

This is the basic principle behind most speech synthesis systems. In speech, the synthesis filter $B(z)$ represents a model of the transfer characteristics of the *vocal tract* considered as an acoustic tube. A typical analysis frame of speech has duration of 20 msec. If sampled at a 10-kHz sampling rate, it will consist of $N = 200$ samples. To synthesize a particular frame of 200 samples, the model parameters representing that frame are recalled from memory, and the synthesis filter is run for 200 sampling instances generating 200 output speech samples, which may be sent to a D/A converter. The next frame of 200 samples can be synthesized by recalling from memory *its* model parameters, and so on. Entire words or sentences can be synthesized in such a piece-wise, or frame-wise, manner.

A realistic representation of each speech frame requires the specification of two additional parameters besides the filter coefficients and σ_ϵ^2, namely, the *pitch* period and a *voiced/unvoiced* (V/UV) decision. Unvoiced sounds, such as the "sh" in the word "should", have a white-noise sounding nature, and are generated by the turbulent flow of air through constrictions of the vocal tract. Such sounds may be represented adequately by the above random signal models. On the other hand, voiced sounds, such as vowels, are pitched sounds, and have a pitch period associated with them. They may be assumed to be generated by the periodic excitation of the vocal tract by a train of impulses separated by the pitch period. The vocal tract responds to each of these impulses by producing its impulse response, resulting therefore in a quasi-periodic output which is characteristic of such sounds. Thus, depending on the type of sound, the nature of the generator of the excitation input to the synthesis filter will be different, namely, it will be a *random noise generator* for unvoiced sounds, and a *pulse train generator* for voiced sounds. A typical speech synthesis system that incorporates the above features is shown in Fig. 1.13.3.

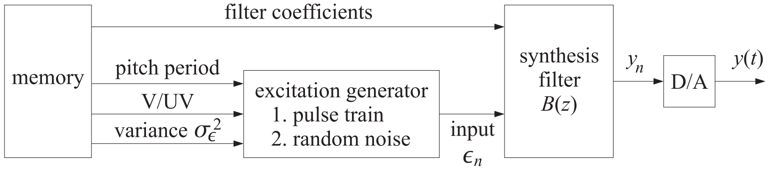


Fig. 1.13.3 Typical speech synthesis system.

Another major application of parametric models is to *spectrum estimation*. This is based on the property that

$$S_{yy}(\omega)=\sigma_\epsilon^2\,\big|B(\omega)\big|^2 \tag{1.13.6}$$

which will be proved later. It states that the spectral shape of the power spectrum $S_{yy}(\omega)$ of the signal y_n arises only from the spectral shape of the model filter $B(\omega)$. For example, the signal y_n generated by the model of Example 1.13.1 will have

$$S_{yy}(\omega)=\sigma_\epsilon^2\left|\frac{1+c_1e^{-j\omega}+c_2e^{-2j\omega}}{1+d_1e^{-j\omega}+d_2e^{-2j\omega}}\right|^2$$

This approach to spectrum estimation is depicted in Fig. 1.13.4. The parametric approach to spectrum estimation must be contrasted with the classical approach which is based on direct computation of the Fourier transform of the available data record, that is, the periodogram spectrum, or its improvements. The classical periodogram method is shown in Fig. 1.13.5. As we mentioned in the previous section, spectrum estimates based on such parametric models tend to have much better frequency resolution properties than the classical methods, especially when the length N of the available data record is short.

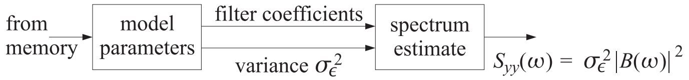

from memory
model parameters
filter coefficients
variance σ_ϵ^2
spectrum estimate
$S_{yy}(\omega)=\sigma_\epsilon^2|B(\omega)|^2$

Fig. 1.13.4 Spectrum estimation with parametric models.

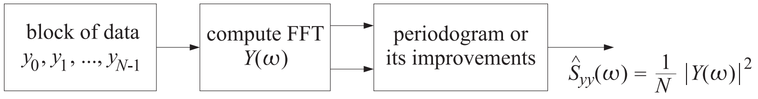

block of data
$y_0, y_1, \dots, y_{N\text{-}1}$
compute FFT
$Y(\omega)$
periodogram or its improvements
$\hat{S}_{yy}(\omega)=\frac{1}{N}|Y(\omega)|^2$

Fig. 1.13.5 Classical spectrum estimation.

In *signal classification* applications, such as speech recognition, speaker verification, or EEG pattern classification, the basic problem is to compare two available blocks of data samples and decide whether they belong to the same class or not. One of the two blocks might be a prestored and preanalyzed *reference template* against which the other block is to be compared. Instead of comparing the data records sample by sample, what are compared are the corresponding model parameters extracted from these blocks. In pattern recognition nomenclature, the vector of model parameters is the “feature vector.” The closeness of the two sets of model parameters to each other is decided on the basis of an appropriate *distance measure*. We will discuss examples of distance measures for speech and EEG signals in Chap. 12. This approach to signal classification is depicted in Fig. 1.13.6.

Next, we discuss the application of such models to *data compression*. The signal synthesis method described above is a form of data compression because instead of saving the N data samples y_n as such, what are saved are the corresponding model parameters which are typically much fewer in number than N. For example, in speech synthesis systems a savings of about a factor of 20 in memory may be achieved with this approach. Indeed, as we discussed above, a typical frame of speech consists of 200 samples, whereas the number of model parameters typically needed to represent this frame is about 10 to 15. The main limitation of this approach is that the reproduction of the original signal segment is not exact but depends on the particular realization of the computer-generated input sequence ϵ_n that drives the model. Speech synthesized in such manner is still intelligible, but it has lost some of its naturalness. Such signal synthesis methods are not necessarily as successful or appropriate in all applications. For example, in image processing, if one makes a parametric model of an image and attempts to “synthesize” it by driving the model with a computer-generated uncorrelated sequence, the reproduced image will bear no resemblance to the original image.

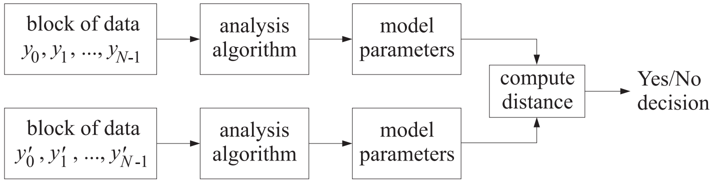


Fig. 1.13.6 Signal classification with parametric models.

For *exact* reproduction, both the model parameters and the entire sequence ϵ_n must be stored. This would still provide some form of data compression, as will be explained below. Such an approach to data compression is widely used in digital data transmission or digital data storage applications for all types of data, including speech and image data. The method may be described as follows: the given data record $\{y_0, y_1, \dots, y_{N-1}\}$ is subjected to an appropriate analysis algorithm to extract the model parameters, and then the segment is filtered through the *inverse filter*,

$$A(z)=\frac{1}{B(z)}\qquad y_n\longrightarrow\boxed{A(z)}\longrightarrow\epsilon_n \tag{1.13.7}$$

to provide the sequence ϵ_n. The inverse filter $A(z)$ is also known as the *whitening filter*, the *prediction-error* filter, or the *analysis filter*. The resulting sequence ϵ_n has a compressed dynamic range relative to y_n and therefore it requires fewer number of bits for the representation of each sample ϵ_n. A quantitative measure for the data compression gain is given by the ratio $G=\sigma_y^2/\sigma_\epsilon^2$, which is always greater than one. This can be seen easily using Eqs. (1.13.6) and (1.10.5)

$$\sigma_y^2=\int_{-\pi}^{\pi}S_{yy}(\omega)\,\frac{d\omega}{2\pi}=\sigma_\epsilon^2\int_{-\pi}^{\pi}\big|B(\omega)\big|^2\,\frac{d\omega}{2\pi}=\sigma_\epsilon^2\sum_{n=0}^{\infty}b_n^2$$

Since $b_0=1$, we find

$$G=\frac{\sigma_y^2}{\sigma_\epsilon^2}=\sum_{n=0}^{\infty}b_n^2=1+b_1^2+b_2^2+\cdots \tag{1.13.8}$$

The entire sequence ϵ_n and the model parameters are then transmitted over the data link, or stored in memory. At the receiving end, the original sequence y_n may be reconstructed exactly using the synthesis filter $B(z)$ driven by ϵ_n. This approach to data compression is depicted in Fig. 1.13.7. Not shown in Fig. 1.13.7 are the quantization and encoding operations that must be performed on ϵ_n in order to transmit it over the digital channel.

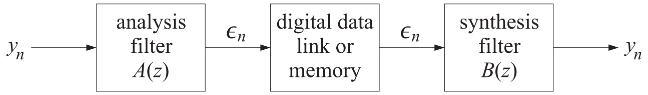


Fig. 1.13.7 Data compression.

Filtering the sequence y_n through the inverse filter requires that $A(z)$ be stable and causal. If we write $B(z)$ as the ratio of two polynomials

$$B(z)=\frac{N(z)}{D(z)} \tag{1.13.9}$$

then the stability and causality of $B(z)$ requires that the zeros of the polynomial $D(z)$ lie inside the unit circle in the complex z-plane; whereas the stability and causality of the inverse $A(z)=D(z)/N(z)$ requires the zeros of $N(z)$ to be inside the unit circle. Thus, both the poles and the zeros of $B(z)$ must be inside the unit circle. Such filters are called *minimal phase filters*. When $A(z)$ is stable and causal it may be expanded in the form

$$A(z)=\sum_{n=0}^{\infty}a_n z^{-n}=1+a_1z^{-1}+a_2z^{-2}+\cdots \tag{1.13.10}$$

and the I/O equation of Eq. (1.13.7) becomes

$$\epsilon_n=\sum_{i=0}^{n}a_i y_{n-i}=y_n+a_1y_{n-1}+a_2y_{n-2}+\cdots \tag{1.13.11}$$

for $n=0,1,2,\dots$. It may be written in matrix form $\boldsymbol{\epsilon}=A\mathbf{y}$ as

$$\begin{bmatrix}\epsilon_0\\ \epsilon_1\\ \epsilon_2\\ \epsilon_3\\ \epsilon_4\end{bmatrix}=\begin{bmatrix}1&0&0&0&0\\ a_1&1&0&0&0\\ a_2&a_1&1&0&0\\ a_3&a_2&a_1&1&0\\ a_4&a_3&a_2&a_1&1\end{bmatrix}\begin{bmatrix}y_0\\ y_1\\ y_2\\ y_3\\ y_4\end{bmatrix}$$

Both this matrix form and Eq. (1.13.11) are recognized as special cases of Eqs. (1.7.1) and (1.7.10). According to Eq. (1.7.11), the quantity

$$\hat{y}_{n/n-1}=-\big[a_1y_{n-1}+a_2y_{n-2}+\cdots+a_ny_0\big] \tag{1.13.12}$$

is the projection of y_n on the subspace spanned by $Y_{n-1}=\{y_{n-1},y_{n-2},\dots,y_0\}$. Therefore, it represents the best linear estimate of y_n on the basis of (all) its past values Y_{n-1}, that is, $\hat{y}_{n/n-1}$ is the *best prediction* of y_n from its (entire) past. Equation (1.13.11) gives the corresponding prediction error $\epsilon_n=y_n-\hat{y}_{n/n-1}$. We note here an interesting connection between linear prediction concepts and signal modeling concepts [21–25], that is, that the best linear predictor (1.13.12) determines the whitening filter $A(z)$ which, in turn, determines the generator model $B(z)=1/A(z)$ of y_n. In other words, solving the prediction problem also solves the modeling problem.

The above modeling approach to the representation of stationary time series, and its relationship to the Gram-Schmidt construction and linear prediction was initiate by Wold and developed further by Kolmogorov [22,24].

The most general model filter $B(z)$ given in Eq. (1.13.9) is called an *autoregressive moving average* (ARMA), or a pole-zero model. Two special cases of interest are the *moving average* (MA), or all-zero models, and the *autoregressive* (AR), or all-pole models. The MA model has a nontrivial numerator only, $B(z)=N(z)$, so that $B(z)$ is a finite polynomial:

$$B(z)=1+b_1z^{-1}+b_2z^{-2}+\cdots+b_Mz^{-M}\qquad\text{(MA model)}$$

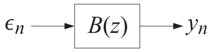
The AR model has a nontrivial denominator only, $B(z)=1/D(z)$, so that its inverse $A(z)=D(z)$ is a polynomial:

$$\begin{aligned}B(z)&=\frac{1}{1+a_1z^{-1}+a_2z^{-2}+\cdots+a_Mz^{-M}}\qquad\text{(AR model)}\\ A(z)&=1+a_1z^{-1}+a_2z^{-2}+\cdots+a_Mz^{-M}\end{aligned}$$

Autoregressive models are the most widely used models, because the analysis algorithms for extracting the model parameters $\{a_1,a_2,\dots,a_M;\sigma_\epsilon^2\}$ are fairly simple. In the sequel, we will concentrate mainly on such models.

1.14 *Filter Model of First Order Autoregressive Process*

To gain some understanding of filter models of the above type, we consider a very simple example of a first-order recursive filter $B(z)$ driven by a purely random sequence of variance σ_ϵ^2:



$$B(z)=\frac{1}{1-az^{-1}}$$

This serves also as a simple model for generating a first order Markov signal. The signal y_n is generated by the difference equation of the filter:

$$y_n=ay_{n-1}+\epsilon_n \tag{1.14.1}$$

Let the probability of the nth sample ϵ_n be $f(\epsilon_n)$. We would like to show that

$$p(y_n|y_{n-1},y_{n-2},\dots,y_1,y_0)=p(y_n|y_{n-1})=f(\epsilon_n)=f(y_n-ay_{n-1})$$

which not only shows the Markov property of y_n, but also how to compute the related conditional density. Perhaps the best way to see this is to start at $n=0$:

$$\begin{aligned}y_0&=\epsilon_0\qquad\text{(assuming zero initial conditions)}\\ y_1&=ay_0+\epsilon_1\\ y_2&=ay_1+\epsilon_2\,,\quad\text{etc.}\end{aligned}$$

To compute $p(y_2|y_1,y_0)$, suppose that y_1 and y_0 are both given. Since y_1 is given, the third equation above shows that the randomness left in y_2 arises from ϵ_2 only. Thus, $p(y_2|y_1)= f(\epsilon_2)$. From the first two equations it follows that specifying y_0 and y_1 is equivalent to specifying ϵ_0 and ϵ_1. Therefore, $p(y_2|y_1,y_0)= f(\epsilon_2|\epsilon_1,\epsilon_0)= f(\epsilon_2)$, the last equation following from the purely random nature of the sequence ϵ_n. We have shown that

$$p(y_2|y_1,y_0)= p(y_2|y_1)= f(\epsilon_2)= f(y_2-ay_1)$$

Using the results of Sec. 1.9, we also note

$$\begin{aligned} p(y_2,y_1,y_0) &= p(y_2|y_1)p(y_1|y_0)p(y_0) \\ &= f(\epsilon_2)f(\epsilon_1)f(\epsilon_0) \\ &= f(y_2-ay_1)f(y_1-ay_0)f(y_0) \end{aligned}$$

The solution of the difference equation (1.14.1) is obtained by convolving the impulse response of the filter $B(z)$

$$b_n = a^n u(n)\,, \qquad u(n)=\text{unit step}$$

with the input sequence ϵ_n as follows:

$$y_n = \sum_{i=0}^{n} b_i\epsilon_{n-i} = \sum_{i=0}^{n} a^i\epsilon_{n-i} \tag{1.14.2}$$

for $n = 0,1,2,\dots$. This is the innovations representation of y_n given by Eqs. (1.6.15), (1.6.16), and (1.13.1). In matrix form it reads:

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ a & 1 & 0 & 0 \\ a^2 & a & 1 & 0 \\ a^3 & a^2 & a & 1 \end{bmatrix} \begin{bmatrix} \epsilon_0 \\ \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \end{bmatrix} \tag{1.14.3}$$

The inverse equation, $\boldsymbol{\epsilon} = B^{-1}\mathbf{y} = A\mathbf{y}$, is obtained by writing Eq. (1.14.1) as $\epsilon_n = y_n - ay_{n-1}$. In matrix form, this reads

$$\begin{bmatrix} \epsilon_0 \\ \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -a & 1 & 0 & 0 \\ 0 & -a & 1 & 0 \\ 0 & 0 & -a & 1 \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix} \tag{1.14.4}$$

According to the discussion of Example 1.7.1, the partial correlation coefficients can be read off from the *first column* of this matrix. We conclude, therefore, that all partial correlation coefficients of order greater than two are zero. This property is in accordance with our intuition about first order Markov processes; due to the recursive nature of Eq. (1.14.1) a given sample, say y_n, will have an indirect influence on all future samples. However, the only direct influence is to the next sample.

Higher order autoregressive random signals can be generated by sending white noise through higher order filters. For example, the second-order difference equation

$$y_n = a_1y_{n-1} + a_2y_{n-2} + \epsilon_n \tag{1.14.5}$$

will generate a second-order Markov signal. In this case, the difference equation directly couples two successive samples, but not more than two. Therefore, all the partial correlations of order greater than three will be zero. This may be seen also by writing Eq. (1.14.5) in matrix form and inspecting the first column of the matrix A:

$$\begin{bmatrix} \epsilon_0 \\ \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \epsilon_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ -a_1 & 1 & 0 & 0 & 0 \\ -a_2 & -a_1 & 1 & 0 & 0 \\ 0 & -a_2 & -a_1 & 1 & 0 \\ 0 & 0 & -a_2 & -a_1 & 1 \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}$$

1.15 Stability and Stationarity

In this section we discuss the importance of stability of the signal generator filter $B(z)$. We demonstrate that the generated signal y_n will be stationary only if the generating filter is stable. And in this case, the sequence y_n will become stationary only after the transient effects introduced by the filter have died out.

To demonstrate these ideas, consider the lag-0 autocorrelation of our first order Markov signal

$$\begin{aligned} R_{yy}(n,n) &= E[y_n^2]= E[(ay_{n-1}+\epsilon_n)^2] \\ &= a^2E[y_{n-1}^2]+2aE[y_{n-1}\epsilon_n]+E[\epsilon_n^2]= a^2R_{yy}(n-1,n-1)+\sigma_\epsilon^2 \end{aligned} \tag{1.15.1}$$

where we set $\sigma_\epsilon^2 = E[\epsilon_n^2]$ and $E[y_{n-1}\epsilon_n]= 0$, which follows by using Eq. (1.14.2) to get

$$y_{n-1} = \epsilon_{n-1} + a\epsilon_{n-2} + \cdots + a^{n-1}\epsilon_0$$

and noting that ϵ_n is uncorrelated with all these terms, due to its white-noise nature. The above difference equation for $R_{yy}(n,n)$ can now be solved to get

$$R_{yy}(n,n)= E[y_n^2]= \frac{\sigma_\epsilon^2}{1-a^2} + \sigma_\epsilon^2\left(1-\frac{1}{1-a^2}\right)a^{2n} \tag{1.15.2}$$

where the initial condition was taken to be $E[y_0^2]= E[\epsilon_0^2]= \sigma_\epsilon^2$. If the filter is stable and causal, that is, $|a|<1$, then the second term in (1.15.2) tends to zero exponentially, and $R_{yy}(n,n)$ eventually loses its dependence on the absolute time n. For large n, it tends to the steady-state value

$$R_{yy}(0)= E[y_n^2]= \sigma_y^2 = \frac{\sigma_\epsilon^2}{1-a^2} \tag{1.15.3}$$

The same result is obtained, of course, by assuming stationarity from the start. The difference equation (1.15.1) can be written as

$$E[y_n^2]= a^2E[y_{n-1}^2]+\sigma_\epsilon^2$$

If y_n is assumed to be already stationary, then $E[y_n^2]= E[y_{n-1}^2]$. This implies the same steady-state solution as Eq. (1.15.3).

If the filter is unstable, that is, $|a| > 1$, then the second term of Eq. (1.15.2) diverges exponentially. The marginal case $a = 1$ is also unacceptable, but is of historical interest being the famous Wiener process, or random walk. In this case, the signal model is

$$y_n = y_{n-1} + \epsilon_n$$

and the difference equation for the variance becomes

$$R_{yy}(n,n) = R_{yy}(n-1,n-1) + \sigma_\epsilon^2$$

with solution

$$R_{yy}(n,n) = E[y_n^2] = (n+1)\sigma_\epsilon^2$$

In summary, for true stationarity to set in, the signal generator filter $B(z)$ must be *strictly* stable (all its poles must be strictly inside the unit circle).

1.16 Parameter Estimation

One of the most important practical questions is how to extract the model parameters, such as the above filter parameter a, from the actual data values. As an introduction to the analysis methods used to answer this question, let us suppose that the white noise input sequence ϵ_n is gaussian

$$f(\epsilon_n) = \frac{1}{\sqrt{2\pi}\sigma_\epsilon}\exp\big(-\frac{\epsilon_n^2}{2\sigma_\epsilon^2}\big)$$

and assume that a block of N measured values of the signal y_n is available

$$y_0, y_1, y_2, \dots, y_{N-1}$$

Can we extract the filter parameter a from this block of data? Can we also extract the variance σ_ϵ^2 of the driving white noise ϵ_n? If so, then instead of saving the N measured values $\{y_0, y_1, y_2, \dots, y_{N-1}\}$, we can save the extracted filter parameter a and the variance σ_ϵ^2. Whenever we want to synthesize our original sequence y_n, we will simply generate a white-noise input sequence ϵ_n of variance σ_ϵ^2, using a pseudorandom number generator routing, and then drive with it the signal model whose parameter a was previously extracted from the original data. Somehow, all the significant information contained in the original samples, has now been packed or compressed into the two numbers a and σ_ϵ^2.

One possible criterion for extracting the filter parameter a is the maximum likelihood (ML) criterion: The parameter a is selected so as to *maximize* the joint density

$$\begin{aligned} p(y_0, y_1, \dots, y_{N-1}) &= f(\epsilon_0)f(\epsilon_1)\cdots f(\epsilon_{N-1}) \\ &= \frac{1}{(\sqrt{2\pi}\sigma_\epsilon)^N}\exp\left[-\frac{1}{2\sigma_\epsilon^2}\sum_{n=1}^{N-1}(y_n - ay_{n-1})^2\right]\exp\left[-y_0^2/2\sigma_\epsilon^2\right] \end{aligned}$$

that is, the parameter a is selected so as to render the actual measured values $\{y_0, y_1, y_2, \dots, y_{N-1}\}$ most likely. The criterion is equivalent to minimizing the exponent with respect to a:

$$\mathcal{E}(a) = \sum_{n=1}^{N-1}(y_n - ay_{n-1})^2 + y_0^2 = \sum_{n=0}^{N-1} e_n^2 = \min \tag{1.16.1}$$

where we set $e_n = y_n - ay_{n-1}$, and $e_0 = y_0$. The minimization of Eq. (1.16.1) gives

$$\frac{\partial\mathcal{E}(a)}{\partial a} = -2\sum_{n=1}^{N-1}(y_n - ay_{n-1})y_{n-1} = 0\,, \quad \text{or,}$$

$$a = \frac{\sum_{n=1}^{N-1} y_n y_{n-1}}{\sum_{n=1}^{N-1} y_{n-1}^2} = \frac{y_0y_1 + y_1y_2 + \cdots + y_{N-2}y_{N-1}}{y_0^2 + y_1^2 + \cdots + y_{N-2}^2} \tag{1.16.2}$$

There is a potential problem with the above ML criterion for extracting the filter parameter a, namely, the parameter may turn out to have magnitude greater than one, which will correspond to an unstable filter generating the sequence y_n. This is easily seen from Eq. (1.16.2); whereas the numerator has dependence on the last sample y_{N-1}, the denominator does not. Therefore it is possible, for sufficiently large values of y_{N-1}, for the parameter a to be greater than one. There are other criteria for extracting the Markov model parameters that guarantee the stability of the resulting synthesis filters, such as the so-called autocorrelation method, or Burg's method. These will be discussed later on.

An alternative parameter estimation method is the *autocorrelation* or *Yule-Walker* method of extracting the model parameters from a block of data. We begin by expressing the model parameters in terms of output statistical quantities and then replace ensemble averages by time averages. Assuming stationarity has set in, we find

$$R_{yy}(1) = E[y_n y_{n-1}] = E\big[(ay_{n-1} + \epsilon_n)y_{n-1}\big] = aE[y_{n-1}^2] + E[\epsilon_n y_{n-1}] = aR_{yy}(0)$$

from which

$$a = \frac{R_{yy}(1)}{R_{yy}(0)}$$

The input parameter σ_ϵ^2 can be expressed as

$$\sigma_\epsilon^2 = (1 - a^2)\sigma_y^2 = (1 - a^2)R_{yy}(0)$$

These two equations may be written in matrix form as

$$\begin{bmatrix} R_{yy}(0) & R_{yy}(1) \\ R_{yy}(1) & R_{yy}(0) \end{bmatrix}\begin{bmatrix} 1 \\ -a \end{bmatrix} = \begin{bmatrix} \sigma_\epsilon^2 \\ 0 \end{bmatrix}$$

These are called the *normal equations* of linear prediction. Their generalization will be considered later on. These results are important because they allow the extraction of the signal model parameters directly in terms of *output* quantities, that is, from experimentally accessible quantities.

We may obtain estimates of the model parameters by replacing the theoretical autocorrelations by the corresponding *sample autocorrelations*, defined by Eq. (1.11.1):

$$\hat{a} = \frac{\hat{R}_{yy}(1)}{\hat{R}_{yy}(0)} = \frac{\displaystyle\frac{1}{N}\sum_{n=0}^{N-1-1} y_{n+1}y_n}{\displaystyle\frac{1}{N}\sum_{n=0}^{N-1} y_n y_n} = \frac{y_0y_1 + y_1y_2 + \cdots + y_{N-2}y_{N-1}}{y_0^2 + y_1^2 + \cdots + y_{N-2}^2 + y_{N-1}^2}$$

$$\hat{\sigma}_\epsilon^2 = (1-\hat{a}^2)\hat{R}_{yy}(0)$$

It is easily checked that the parameter $\hat{a}$, defined as above, is always of magnitude less than one; thus, the stability of the synthesis filter is guaranteed. Note the difference with the ML expression. The numerators are the same, but the denominators differ by an extra term. It is also interesting to note that the above expressions may be obtained by a *minimization criterion*; known as the autocorrelation method, or the Yule-Walker method:

$$\mathcal{E}(a) = \sum_{n=0}^{N} e_n^2 = \sum_{n=0}^{N}(y_n - ay_{n-1})^2 = \min \qquad (1.16.3)$$

This differs from the ML criterion (1.16.1) only in the range of summation for n. Whereas in the ML criterion the summation index n does not run off the ends of the data block, it does so in the Yule-Walker case. We may think of the block of data as having been extended to both directions by padding it with zeros

$$0,\dots,0,y_0,y_1,\dots,y_{N-1},0,0,\dots,0$$

The difference between this and the ML criterion arises from the last term in the sum

$$\mathcal{E}(a) = \sum_{n=0}^{N} e_n^2 = \sum_{n=1}^{N-1} e_n^2 + e_N^2 = \sum_{n=1}^{N-1}(y_n - ay_{n-1})^2 + (0 - ay_{N-1})^2$$

The Yule-Walker analysis algorithm for this first order example is summarized in Fig. 1.16.1.

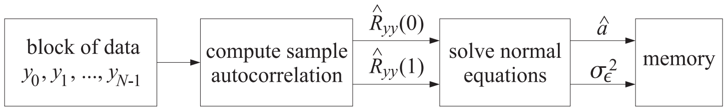


Fig. 1.16.1 Yule-Walker analysis method.

How good are $\hat{a}$ and $\hat{\sigma}_\epsilon^2$ as estimates of the model parameters a and σ_ϵ^2? It can be shown that they, and the maximum likelihood estimates of the previous section, are asymptotically unbiased and consistent. The corresponding variances are given for large N by [4–6]

$$E\big[(\Delta a)^2\big] = \frac{1-a^2}{N}\,, \quad E\big[(\Delta\sigma_\epsilon^2)^2\big] = \frac{2\sigma_\epsilon^4}{N} \qquad (1.16.4)$$

where $\Delta a = \hat{a} - a$ and $\Delta\sigma_\epsilon^2 = \hat{\sigma}_\epsilon^2 - \sigma_\epsilon^2$. Such asymptotic properties are discussed in greater detail in Chap. 14. Here, we present some simulation examples showing that (1.16.4) are adequate even for fairly small N.

Example 1.16.1: The following $N = 30$ signal samples of y_n have been generated by passing zero-mean white noise through the difference equation $y_n = ay_{n-1} + \epsilon_n$, with $a = 0.8$ and $\sigma_\epsilon^2 = 1$:

$$\begin{aligned} y_n = \{&2.583, 2.617, 2.289, 2.783, 2.862, 3.345, 2.704, 1.527, 2.096, 2.050, 2.314,\\ &0.438, 1.276, 0.524, -0.449, -1.736, -2.599, -1.633, 1.096, 0.348, 0.745,\\ &0.797, 1.123, 1.031, -0.219, 0.593, 2.855, 0.890, 0.970, 0.924\}\end{aligned}$$

Using the Yule-Walker method, we obtain the following estimates of the model parameters

$$\hat{a} = 0.806\,, \quad \sigma_\epsilon^2 = 1.17$$

Both estimates are consistent with the theoretically expected fluctuations about their means given by Eq. (1.16.4), falling within the one-standard deviation intervals $a \pm \delta a$ and $\sigma_\epsilon^2 \pm \delta\sigma_\epsilon^2$, where δa and $\delta\sigma_\epsilon^2$ are the square roots of Eq. (1.16.4). For $N = 30$, the numerical values of these intervals are: $0.690 \le \hat{a} \le 0.910$ and $0.742 \le \sigma_\epsilon^2 \le 1.258$. Given the theoretical and estimated model parameters, we can obtain the theoretical and estimated power spectral densities of y_n by

$$S_{\rm TH}(\omega) = \frac{\sigma_\epsilon^2}{\big|1 - ae^{-j\omega}\big|^2}\,, \quad S_{\rm YW}(\omega) = \frac{\hat{\sigma}_\epsilon^2}{\big|1 - \hat{a}e^{-j\omega}\big|^2}$$

The periodogram spectrum based on the given length-N data block is

$$S_{\rm PER}(\omega) = \frac{1}{N}\left|\sum_{n=0}^{N-1} y_n e^{-jn\omega}\right|^2$$

The three spectra are plotted in Fig. 1.16.2, in units of decibels; that is, $10\log_{10}S$, over the right half of the Nyquist interval $0 \le \omega \le \pi$. Note the excellent agreement of the Yule-Walker spectrum with the theoretical spectrum and the several sidelobes of the periodogram spectrum caused by the windowing of y_n.

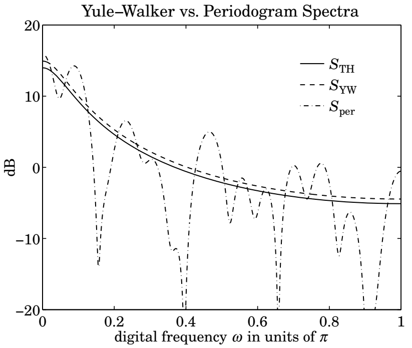


Fig. 1.16.2 Comparison of Yule-Walker and periodogram spectrum estimates.

Example 1.16.2: The purpose of this example is to demonstrate the reasonableness of the asymptotic variances, Eq. (1.16.4). For the first-order model defined in the previous example, we generated 100 different realizations of the length-30 signal block y_n. From each realization, we extracted the Yule-Walker estimates of the model parameters $\hat{a}$ and $\hat{\sigma}_\epsilon^2$. They are shown in Figs. 1.16.3 versus realization index, together with the corresponding asymptotic one-standard deviation intervals that were computed in the previous example.

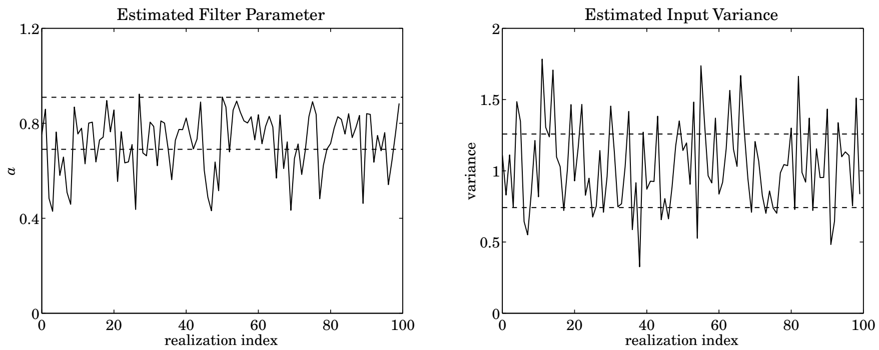


Fig. 1.16.3 Parameters a, σ_ϵ^2 estimated from 100 realizations of the length-30 data block y_n.

1.17 Linear Prediction and Signal Modeling

Linear prediction ideas are introduced in the context of our simple example by noting that the least-squares minimization criteria (1.16.1) and (1.16.3)

$$\mathcal{E}(a)=\sum_n e_n^2 = \text{minimum} \tag{1.17.1}$$

essentially force each e_n to be small. Thus, if we reinterpret

$$\hat{y}_n = ay_{n-1}$$

as the linear prediction of the sample y_n made on the basis of just the previous sample y_{n-1}, then $e_n = y_n - ay_{n-1} = y_n - \hat{y}_n$ may be thought of as the prediction error. The minimization criterion (1.17.1) essentially minimizes the prediction error in an average least-squares sense, thus attempting to make the best prediction possible.

As we mentioned in Sec. 1.13, the solution of the linear prediction problem provides the corresponding random signal generator model for y_n, which can be used, in turn, in a number of ways as outlined in Sec. 1.13. This is the main reason for our interest in linear prediction.

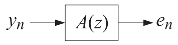
A more intuitive way to understand the connection between linear prediction and signal models is as follows: Suppose we have a predictor $\hat{y}_n$ of y_n which is not necessarily the best predictor. The predictor $\hat{y}_n$ is given as a linear combination of the past values $\{y_{n-1}, y_{n-2}, \dots\}$:

$$\hat{y}_n = -\big[a_1y_{n-1} + a_2y_{n-2} + \cdots\big] \tag{1.17.2}$$

The corresponding prediction error will be

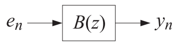
$$e_n = y_n - \hat{y}_n = y_n + a_1y_{n-1} + a_2y_{n-2} + \cdots \tag{1.17.3}$$

and it may be considered as the output of the prediction-error filter $A(z)$ (which is assumed to be stable and causal):



$$A(z) = 1 + a_1z^{-1} + a_2z^{-2} + \cdots$$

Suppose further that $A(z)$ has a stable and causal inverse filter



$$B(z) = \frac{1}{A(z)} = \frac{1}{1 + a_1z^{-1} + a_2z^{-2} + \cdots}$$

so that y_n may be expressed *causally* in terms of e_n, that is,

$$y_n = e_n + b_1e_{n-1} + b_2e_{n-2} + \cdots \tag{1.17.4}$$

Then, Eqs. (1.17.3) and (1.17.4) imply that the linear spaces generated by the random variables

$$\{y_{n-1}, y_{n-2}, \dots\} \quad \text{and} \quad \{e_{n-1}, e_{n-2}, \dots\}$$

are the same space. One can pass from one set to the other by a causal and causally invertible linear filtering operation.

Now, if the prediction $\hat{y}_n$ of y_n is the best possible prediction, then what remains after the prediction is made—namely, the error signal e_n—should be entirely *unpredictable* on the basis of the past values $\{y_{n-1}, y_{n-2}, \dots\}$. That is, e_n must be uncorrelated with all of these. But this implies that e_n must be uncorrelated with all $\{e_{n-1}, e_{n-2}, \dots\}$, and therefore e_n must be a white-noise sequence. It follows that $A(z)$ and $B(z)$ are the analysis and synthesis filters for the sequence y_n.

The least-squares minimization criteria of the type (1.17.1) that are based on time averages, provide a practical way to solve the linear prediction problem and hence also the modeling problem. Their generalization to higher order predictors will be discussed in Chap. 12.

1.18 Cramér–Rao Bound and Maximum Likelihood

The Cramér-Rao inequality [2–5,27] provides a lower bound for the variance of unbiased estimators of parameters. Thus, the best any parameter estimator can do is to meet its Cramér-Rao bound. Such estimators are called *efficient*. Parameter estimators based on the principle of *maximum likelihood*, such as the one presented in Sec. 1.16, have several nice properties, namely, as the number of observations becomes large, they are asymptotically unbiased, consistent, efficient, and are asymptotically normally distributed about the theoretical value of the parameter with covariance given by the Cramér-Rao bound.

In this section, we present a derivation of the Cramér-Rao inequality using correlation canceling methods and discuss its connection to maximum likelihood. Consider

N observations $Y = \{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_N\}$, where each observation is assumed to be an M-dimensional random vector. Based on these observations, we would like to estimate a number of (deterministic) parameters, assembled into a parameter vector $\boldsymbol{\lambda}$. We will write $p(Y, \boldsymbol{\lambda})$ to indicate the dependence of the joint probability density on $\boldsymbol{\lambda}$. As a concrete example, consider the case of N independent scalar observations drawn from a normal distribution with mean m and variance σ^2. The joint density is

$$p(Y, \boldsymbol{\lambda}) = (2\pi\sigma^2)^{-N/2} \exp\left[-\frac{1}{2\sigma^2} \sum_{n=1}^{N} (y_n - m)^2 \right] \tag{1.18.1}$$

For the parameter vector we may choose $\boldsymbol{\lambda} = [m, \sigma^2]^T$, if we want to estimate both the mean and variance.

The dependence of $p(Y, \boldsymbol{\lambda})$ on $\boldsymbol{\lambda}$ may be expressed in terms of the gradient with respect to $\boldsymbol{\lambda}$ of the log-likelihood function

$$\boldsymbol{\psi}(Y, \boldsymbol{\lambda}) \equiv \frac{\partial}{\partial \boldsymbol{\lambda}} \ln p(Y, \boldsymbol{\lambda}) = \frac{1}{p} \frac{\partial p}{\partial \boldsymbol{\lambda}} \tag{1.18.2}$$

Expectation values with respect to the joint density will, in general, depend on the parameter $\boldsymbol{\lambda}$. We have the following result for the expectation value of an arbitrary function $F(Y, \boldsymbol{\lambda})$:

$$\frac{\partial}{\partial \boldsymbol{\lambda}} E[F] = E\left[\frac{\partial F}{\partial \boldsymbol{\lambda}} \right] + E[F\boldsymbol{\psi}] \tag{1.18.3}$$

Writing $dY = d^M\mathbf{y}_1 d^M\mathbf{y}_2 \cdots d^M\mathbf{y}_N$ for the volume element over the space of observations, the proof of Eq. (1.18.3) follows from

$$\frac{\partial}{\partial \boldsymbol{\lambda}} \int pF\, dY = \int \frac{\partial}{\partial \boldsymbol{\lambda}} (pF)\, dY = \int p \frac{\partial F}{\partial \boldsymbol{\lambda}}\, dY + \int pF \frac{\partial \ln p}{\partial \boldsymbol{\lambda}}\, dY$$

Applying this property to $F = 1$, we find $E[\boldsymbol{\psi}] = 0$. Applying it to $\boldsymbol{\psi}$ itself, that is, $F = \boldsymbol{\psi}$, we find

$$J \equiv E[\boldsymbol{\psi}\boldsymbol{\psi}^T] = E[\Psi] \tag{1.18.4}$$

where

$$\Psi \equiv -\frac{\partial \boldsymbol{\psi}}{\partial \boldsymbol{\lambda}}$$

Eq. (1.18.4) is known as the *Fisher information matrix* based on Y. Component-wise, we have

$$J_{ij} = E[\psi_i \psi_j] = E[\Psi_{ij}]$$

where

$$\psi_i = \frac{\partial \ln p}{\partial \lambda_i}, \quad \Psi_{ij} = -\frac{\partial \psi_i}{\partial \lambda_j} = -\frac{\partial^2 \ln p}{\partial \lambda_i \partial \lambda_j}$$

Next, we derive the Cramér-Rao bound. Let $\hat{\boldsymbol{\lambda}}(Y)$ be any estimator of $\boldsymbol{\lambda}$ based on Y. Because $\hat{\boldsymbol{\lambda}}(Y)$ and $\boldsymbol{\psi}(Y, \boldsymbol{\lambda})$ both depend on Y, they will be correlated with each other. Using the correlation canceling methods of Sec. 1.4, we can remove these correlations by writing

$$\mathbf{e} = \hat{\boldsymbol{\lambda}} - E[\hat{\boldsymbol{\lambda}}\boldsymbol{\psi}^T] E[\boldsymbol{\psi}\boldsymbol{\psi}^T]^{-1} \boldsymbol{\psi}$$

Then, $\mathbf{e}$ will not be correlated with $\boldsymbol{\psi}$. Because $\boldsymbol{\psi}$ has zero mean, it follows that $E[\hat{\boldsymbol{\lambda}}] = E[\mathbf{e}]$. Working with the deviations about the corresponding means, namely, $\Delta\boldsymbol{\lambda} = \hat{\boldsymbol{\lambda}} - E[\hat{\boldsymbol{\lambda}}]$ and $\Delta\mathbf{e} = \mathbf{e} - E[\mathbf{e}]$, we have

$$\Delta\mathbf{e} = \Delta\boldsymbol{\lambda} - MJ^{-1}\boldsymbol{\psi} \tag{1.18.5}$$

where we denoted $M = E[\hat{\boldsymbol{\lambda}}\boldsymbol{\psi}^T]$. Following Eq. (1.4.4), we obtain for the covariance of $\Delta\mathbf{e}$

$$E[\Delta\mathbf{e}\Delta\mathbf{e}^T] = E[\Delta\boldsymbol{\lambda}\Delta\boldsymbol{\lambda}^T] - MJ^{-1}M^T \tag{1.18.6}$$

Thus, the difference of terms in the right-hand side is a positive semi-definite matrix. This may be expressed symbolically as $E[\Delta\mathbf{e}\Delta\mathbf{e}^T] \geq 0$, or, $E[\Delta\boldsymbol{\lambda}\Delta\boldsymbol{\lambda}^T] \geq MJ^{-1}M^T$. The quantity M depends on the *bias* of the estimator. For an *unbiased* estimator, M is the identity matrix, $M = I$, and we obtain the Cramér-Rao inequality

$$\text{cov}(\hat{\boldsymbol{\lambda}}) = E[\Delta\boldsymbol{\lambda}\Delta\boldsymbol{\lambda}^T] \geq J^{-1} \qquad \text{(Cramér-Rao)} \tag{1.18.7}$$

The dependence of M on the bias can be seen as follows. Because $\hat{\boldsymbol{\lambda}}(Y)$ has no explicit dependence on $\boldsymbol{\lambda}$, it follows from property (1.18.3) that

$$M = E[\hat{\boldsymbol{\lambda}}\boldsymbol{\psi}^T] = \frac{\partial}{\partial \boldsymbol{\lambda}} E[\hat{\boldsymbol{\lambda}}]$$

Define the bias of the estimator as the deviation of the mean from the true value of the parameter, that is, $E[\hat{\boldsymbol{\lambda}}] = \boldsymbol{\lambda} + \mathbf{b}(\boldsymbol{\lambda})$, where $\mathbf{b}(\boldsymbol{\lambda})$ is the bias

$$M = I + \frac{\partial \mathbf{b}}{\partial \boldsymbol{\lambda}} \equiv I + B$$

For an unbiased estimator, $B = 0$ and $M = I$. It follows from Eq. (1.18.6) that for the Cramér-Rao inequality to be satisfied as an equality, it is necessary that $\Delta\mathbf{e} = 0$ in Eq. (1.18.5), i.e., $\Delta\boldsymbol{\lambda} = MJ^{-1}\boldsymbol{\psi}$ and in the unbiased case, we obtain the condition $\boldsymbol{\psi} = J\Delta\boldsymbol{\lambda}$:

$$\frac{\partial}{\partial \boldsymbol{\lambda}} \ln p(Y, \boldsymbol{\lambda}) = J\Delta\boldsymbol{\lambda} = J[\hat{\boldsymbol{\lambda}}(Y) - \boldsymbol{\lambda}] \tag{1.18.8}$$

Estimators that satisfy this condition and thus, meet their Cramér-Rao bound, are called efficient.

Example 1.18.1: The log-likelihood function of Eq. (1.18.1) is

$$\ln p = -\frac{N}{2} \ln(2\pi) - \frac{N}{2} \ln \sigma^2 - \frac{1}{2\sigma^2} \sum_{n=1}^{N} (y_n - m)^2$$

The gradients with respect to the parameters m and σ^2 are

$$\begin{aligned} \frac{\partial \ln p}{\partial m} &= \frac{1}{\sigma^2} \sum_{n=1}^{N} (y_n - m) \\ \frac{\partial \ln p}{\partial \sigma^2} &= -\frac{N}{2\sigma^2} + \frac{1}{2\sigma^4} \sum_{n=1}^{N} (y_n - m)^2 \end{aligned} \tag{1.18.9}$$

The second derivatives are the matrix elements of the matrix Ψ:

$$\Psi_{mm} = -\frac{\partial^2 \ln p}{\partial m \partial m} = \frac{N}{\sigma^2}$$

$$\Psi_{m\sigma^2} = -\frac{\partial^2 \ln p}{\partial m \partial \sigma^2} = \frac{1}{\sigma^4}\sum_{n=1}^{N}(y_n - m)$$

$$\Psi_{\sigma^2\sigma^2} = -\frac{\partial^2 \ln p}{\partial \sigma^2 \partial \sigma^2} = -\frac{N}{2\sigma^4} + \frac{1}{\sigma^6}\sum_{n=1}^{N}(y_n - m)^2$$

Taking expectation values, we find the matrix elements of J

$$J_{mm} = \frac{N}{\sigma^2}\,, \quad J_{m\sigma^2} = 0\,, \quad J_{\sigma^2\sigma^2} = \frac{N}{2\sigma^4}$$

Therefore, the Cramér-Rao bound of any unbiased estimator of m and σ^2 will be

$$\begin{bmatrix} E[\Delta m \Delta m] & E[\Delta m \Delta \sigma^2] \\ E[\Delta \sigma^2 \Delta m] & E[\Delta \sigma^2 \Delta \sigma^2] \end{bmatrix} \geq \begin{bmatrix} \sigma^2/N & 0 \\ 0 & 2\sigma^4/N \end{bmatrix}$$

Example 1.18.2: We note that the sample mean $\hat{m}$ defined by Eq. (1.2.1) has variance equal to its Cramér-Rao bound, and therefore, it is an efficient estimator. It also satisfies condition (1.18.8). Writing $\sum_{n=1}^{N} y_n = N\hat{m}$, we obtain from Eq. (1.18.9)

$$\frac{\partial \ln p}{\partial m} = \frac{1}{\sigma^2}\sum_{n=1}^{N}(y_n - m) = \frac{1}{\sigma^2}\left[\sum_{n=1}^{N} y_n - Nm\right] = \frac{1}{\sigma^2}(N\hat{m} - Nm) = J_{mm}(\hat{m} - m)$$

We also note that the sample variance s^2 having variance $2\sigma^4/(N-1)$ meets its Cramér-Rao bound only asymptotically. The biased definition of the sample variance, Eq. (1.2.3), has variance given by Eq. (1.2.4). It is easily verified that it is *smaller* than its Cramér-Rao bound (1.18.7). But this is no contradiction because Eq. (1.18.7) is valid only for unbiased estimators. For a biased estimator, the lower bound $MJ^{-1}M^T$ must be used. Equation (1.2.4) does satisfy this bound. □

Next, we discuss the principle of maximum likelihood. The *maximum likelihood estimator* of a parameter $\boldsymbol{\lambda}$ is the value $\hat{\boldsymbol{\lambda}}$ that maximizes the joint density $p(Y, \boldsymbol{\lambda})$; i.e.,

$$p(Y, \boldsymbol{\lambda})\,\big|_{\boldsymbol{\lambda}=\hat{\boldsymbol{\lambda}}} = \text{maximum} \tag{1.18.10}$$

Equivalently,

$$\boldsymbol{\psi}(\hat{\boldsymbol{\lambda}}) = \frac{\partial}{\partial \boldsymbol{\lambda}} \ln p(Y, \boldsymbol{\lambda})\,\Big|_{\boldsymbol{\lambda}=\hat{\boldsymbol{\lambda}}} = 0 \tag{1.18.11}$$

In general, this equation is difficult to solve. However, the asymptotic properties of the solution for large N are simple enough to obtain. Assuming that $\hat{\boldsymbol{\lambda}}$ is near the true value of the parameter $\boldsymbol{\lambda}$ we may expand the gradient $\boldsymbol{\psi}$ about the true value:

$$\boldsymbol{\psi}(\hat{\boldsymbol{\lambda}}) \simeq \boldsymbol{\psi} + \frac{\partial \boldsymbol{\psi}(\boldsymbol{\lambda})}{\partial \boldsymbol{\lambda}}(\hat{\boldsymbol{\lambda}} - \boldsymbol{\lambda}) = \boldsymbol{\psi} - \Psi(\hat{\boldsymbol{\lambda}} - \boldsymbol{\lambda})$$

where we used the matrix Ψ defined in Eq. (1.18.4). For the maximum likelihood solution, the left-hand side is zero. Thus, solving for $\Delta\boldsymbol{\lambda} = \hat{\boldsymbol{\lambda}} - \boldsymbol{\lambda}$, we obtain

$$\Delta\boldsymbol{\lambda} = \Psi^{-1}\boldsymbol{\psi} \tag{1.18.12}$$

Assuming that the N observations are independent of each other, the joint density $p(Y, \boldsymbol{\lambda})$ factors into the marginal densities $\prod_{n=1}^{N} p(\mathbf{y}_n, \boldsymbol{\lambda})$. Therefore, the gradient $\boldsymbol{\psi}$ will be a sum of gradients

$$\boldsymbol{\psi} = \frac{\partial}{\partial \boldsymbol{\lambda}} \ln p = \sum_{n=1}^{N} \frac{\partial}{\partial \boldsymbol{\lambda}} \ln p(\mathbf{y}_n, \boldsymbol{\lambda}) = \sum_{n=1}^{N} \boldsymbol{\psi}_n$$

Similarly,

$$\Psi = -\frac{\partial \boldsymbol{\psi}}{\partial \boldsymbol{\lambda}} - \sum_{n=1}^{N} \frac{\partial \boldsymbol{\psi}_n}{\partial \boldsymbol{\lambda}} = \sum_{N=1}^{N} \Psi_n$$

Individual terms in these sums are mutually independent. Thus, from the law of large numbers, we can replace Ψ by its mean $\Psi \simeq E[\Psi] = J$, and Eq. (1.18.12) becomes

$$\Delta\boldsymbol{\lambda} = J^{-1}\boldsymbol{\psi} \tag{1.18.13}$$

This asymptotic equation contains essentially all the nice properties of the maximum likelihood estimator. First, from $E[\Psi] = 0$, it follows that $E[\Delta\boldsymbol{\lambda}] = 0$, or that $\hat{\boldsymbol{\lambda}}$ is *asymptotically unbiased*. Second, its asymptotic covariance agrees with the Cramér-Rao bound

$$E[\Delta\boldsymbol{\lambda}\Delta\boldsymbol{\lambda}^T] = J^{-1}E[\boldsymbol{\psi}\boldsymbol{\psi}^T]J^{-1} = J^{-1}JJ^{-1} = J^{-1}$$

Thus, $\hat{\boldsymbol{\lambda}}$ is *asymptotically efficient*. The same conclusion can be reached by noting that Eq. (1.18.13) is the same as condition (1.18.8). Third, $\hat{\boldsymbol{\lambda}}$ is *asymptotically consistent*, in the sense that its covariance tends to zero for large N. This follows from the fact that the information matrix for N independent observations is equal to N times the information matrix for one observation:

$$J = E[\Psi] = \sum_{n=1}^{N} E[\Psi_n] = NE[\Psi_1] = NJ_1$$

Therefore, $J^{-1} = J_1^{-1}/N$ tends to zero for large N. Fourth, because $\boldsymbol{\psi}$ is the sum of N independent terms, it follows from the vector version of the central limit theorem that $\boldsymbol{\psi}$ will be *asymptotically normally distributed*. Thus, so will be $\hat{\boldsymbol{\lambda}}$, with mean $\boldsymbol{\lambda}$ and covariance J^{-1}.

Example 1.18.3: Setting the gradients (1.18.9) to zero, we obtain the maximum likelihood estimates of the parameters m and σ^2. It is easily verified that they coincide with the sample mean and sample variance defined by Eqs. (1.2.1) and (1.2.3). □

Example 1.18.4: In many applications, the mean is known to be zero and only the variance needs to be estimated. For example, setting $m = 0$ in Eq. (1.18.1) we obtain the log-likelihood

$$\ln p = -\frac{N}{2}\ln(2\pi) - \frac{N}{2}\ln\sigma^2 - \frac{1}{2\sigma^2}\sum_{n=1}^{N} y_n^2$$

The maximum likelihood estimate of σ^2 is obtained from

$$\frac{\partial \ln p}{\partial \sigma^2} = -\frac{N}{2\sigma^2} + \frac{1}{2\sigma^4}\sum_{n=1}^{N} y_n^2 = 0$$

with solution

$$\hat{\sigma}^2 = \frac{1}{N}\sum_{n=1}^{N} y_n^2$$

It is easily verified that this is an *unbiased* estimate. It is the scalar version of Eq. (1.6.21). Using $E[y_n^2 y_m^2]= \sigma^4 + 2\delta_{nm}\sigma^4$, which is valid for independent zero-mean gaussian y_ns, we find for the variance of $\hat{\sigma}^2$

$$E[\Delta\sigma^2\Delta\sigma^2]= \frac{2\sigma^4}{N}\,, \quad \text{where} \quad \Delta\sigma^2 = \hat{\sigma}^2 - \sigma^2 \tag{1.18.14}$$

This agrees with the corresponding Cramér-Rao bound. Thus, $\hat{\sigma}^2$ is efficient. Equation (1.18.14) is the scalar version of Eq. (1.6.23). □

Example 1.18.5: Show that the multivariate sample covariance matrix, $\hat{R}$, given by Eq. (1.6.21) is the maximum likelihood estimate of R, assuming the mean is zero.

Solution: The log-likelihood function is, up to a constant

$$\ln p(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_N) = -\frac{N}{2}\ln(\det R) - \frac{1}{2}\sum_{n=1}^{N}\mathbf{y}_n^T R^{-1}\mathbf{y}_n$$

The second term may be written as the trace:

$$\sum_{n=1}^{N}\mathbf{y}_n^T R^{-1}\mathbf{y}_n = \mathrm{tr}[R^{-1}\sum_{n=1}^{N}\mathbf{y}_n\mathbf{y}_n^T] = N\,\mathrm{tr}[R^{-1}\hat{R}]$$

where we used $\sum_{n=1}^{N}\mathbf{y}_n\mathbf{y}_n^T = N\hat{R}$. Using the matrix property $\ln(\det R)= \mathrm{tr}(\ln R)$, we may write the log-likelihood in the form

$$\ln p = -\frac{N}{2}\,\mathrm{tr}[\ln R + R^{-1}\hat{R}]$$

The maximum likelihood solution for R satisfies $\partial\ln p/\partial R = 0$. To solve it, we find it more convenient to work with differentials. Using the two matrix properties

$$d\,\mathrm{tr}(\ln R)= \mathrm{tr}(R^{-1}dR)\,, \quad dR^{-1} = -R^{-1}(dR)R^{-1} \tag{1.18.15}$$

we obtain,

$$d\ln p = -\frac{N}{2}\,\mathrm{tr}[R^{-1}dR - R^{-1}(dR)R^{-1}\hat{R}] = -\frac{N}{2}\,\mathrm{tr}[R^{-1}(dR)R^{-1}(R-\hat{R})] \tag{1.18.16}$$

Because dR is arbitrary, the vanishing of $d\ln p$ implies $R = \hat{R}$. An alternative proof is to show that $f(R)\geq f(\hat{R})$, where $f(R)\equiv \mathrm{tr}(\ln R + R^{-1}\hat{R})$. This is shown easily using the inequality $x - 1 - \ln x \geq 0$, for $x \geq 0$, with equality reached at $x = 1$. □

In many applications, the desired parameter $\boldsymbol{\lambda}$ to be estimated appears only through the covariance matrix R of the observations $\mathbf{y}$, that is, $R = R(\boldsymbol{\lambda})$. For example, we will see in Chap. 14 that the covariance matrix of a plane wave incident on an array of two sensors in the presence of noise is given by

$$R = \begin{bmatrix} P+\sigma^2 & Pe^{jk} \\ Pe^{-jk} & P+\sigma^2 \end{bmatrix}$$

where possible parameters to be estimated are the power P and wavenumber k of the wave, and the variance σ^2 of the background noise. Thus, $\boldsymbol{\lambda} = [P, k, \sigma^2]^T$.

In such cases, we have the following general expression for the Fisher information matrix J, valid for independent zero-mean gaussian observations:

$$J_{ij} = \frac{N}{2}\,\mathrm{tr}\left[R^{-1}\frac{\partial R}{\partial\lambda_i}R^{-1}\frac{\partial R}{\partial\lambda_j}\right] \tag{1.18.17}$$

Writing $\partial_i = \partial/\partial\lambda_i$ for brevity, we have from Eq. (1.18.16)

$$\partial_i\ln p = -\frac{N}{2}\,\mathrm{tr}[R^{-1}\partial_i R\,R^{-1}(R-\hat{R})]$$

Differentiating once more, we find

$$\Psi_{ij} = -\partial_i\partial_j\ln p = \frac{N}{2}\,\mathrm{tr}[\partial_j(R^{-1}\partial_i R\,R^{-1})(R-\hat{R}) + R^{-1}\partial_i R\,R^{-1}\partial_j R]$$

Equation (1.18.17) follows now by taking expectation values $J_{ij} = E[\Psi_{ij}]$ and noting that the expectation value of the first term vanishes. This follows from the fact that $\hat{R}$ is an unbiased estimator of R and therefore, $E\big[\mathrm{tr}\big(F(R-\hat{R})\big)\big] = 0$, for any matrix F.

1.19 Minimum-Phase Signals and Filters

A *minimum-phase sequence* $\mathbf{a} = [a_0, a_1, \dots, a_M]$ has a z-transform with all its zeros inside the unit circle in the complex z-plane

$$A(z)= a_0 + a_1 z^{-1} + \cdots + a_M z^{-M} = a_0(1-z_1z^{-1})(1-z_2z^{-1})\cdots(1-z_Mz^{-1}) \tag{1.19.1}$$

with $|z_i| < 1$, $i = 1, 2, \dots, M$. Such a polynomial is also called a *minimum-delay* polynomial. Define the following related polynomials:

$$\begin{aligned}
A^*(z) &= a_0^* + a_1^* z^{-1} + \cdots + a_M^* z^{-M} = \text{complex-conjugated coefficients} \\
\bar{A}(z) &= a_0^* + a_1^* z + \cdots + a_M^* z^{M} = \text{conjugated and reflected} \\
A^R(z) &= a_M^* + a_{M-1}^* z^{-1} + \cdots + a_0^* z^{-M} = \text{reversed and conjugated}
\end{aligned}$$

We note the relationships:

$$\bar{A}(z)= A^*(z^{-1}) \quad \text{and} \quad A^R(z)= z^{-M}\bar{A}(z)= z^{-M}A^*(z^{-1}) \tag{1.19.2}$$

We also note that when we set $z = e^{j\omega}$ to obtain the corresponding frequency responses, $\bar{A}(\omega)$ becomes the complex conjugate of $A(\omega)$

$$\bar{A}(\omega)= A(\omega)^* \tag{1.19.3}$$

It is easily verified that all these polynomials have the *same* magnitude spectrum:

$$|A(\omega)|^2 = |\bar{A}(\omega)|^2 = |A^*(\omega)|^2 = |A^R(\omega)|^2 \tag{1.19.4}$$

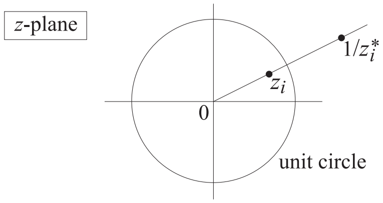
For example, in the case of a doublet $\mathbf{a} = (a_0, a_1)$ and its reverse $\mathbf{a}^R = (a_1^*, a_0^*)$, we verify explicitly

$$\begin{aligned}|A(\omega)|^2 = A(\omega)A(\omega)^* &= (a_0 + a_1 e^{-j\omega})(a_0^* + a_1^* e^{j\omega})\\ &= (a_1^* + a_0^* e^{-j\omega})(a_1 + a_0 e^{j\omega})\\ &= A^R(\omega)A^R(\omega)^* = |A^R(\omega)|^2\end{aligned}$$

Thus, on the basis the magnitude spectrum, one cannot distinguish the doublet $\mathbf{a} = (a_0, a_1)$ from its reverse $\mathbf{a}^R = (a_1^*, a_0^*)$. In the more general case of a polynomial of degree M, factored into doublets as in Eq. (1.19.1), we note that each doublet can be replaced by its reverse

$$(1, -z_i) \rightarrow (-z_i^*, 1) \quad \text{or} \quad (1 - z_i z^{-1}) \rightarrow (-z_i^* + z^{-1})$$

without affecting the overall magnitude spectrum $|A(\omega)|^2$. Since there are M such factors, there will be a total of 2^M different Mth degree polynomials, or equivalently, 2^M different length-$(M+1)$ sequences, all having the *same* magnitude spectrum. Every time a factor $(1 - z_i z^{-1})$ is reversed to become $(-z_i^* + z^{-1})$, the corresponding zero changes from $z = z_i$ to $z = 1/z_i^*$. If z_i is inside the unit circle, the $1/z_i^*$ is outside, as shown



To enumerate all these sequences, start by taking all zeros z_i to be inside the unit circle and successively keep reversing each factor until all 2^M possibilities have been exhausted. At the last step, all the factors will have been flipped, corresponding to all the zeros being outside the unit circle. The resulting polynomial and sequence are referred to as having *maximal phase*, or *maximal delay*. As an example consider the two doublets

$$\mathbf{a} = (2, 1) \quad \text{and} \quad \mathbf{b} = (3, 2)$$

and form the four different sequences, where $*$ denotes convolution:

$$\begin{aligned}\mathbf{c}_0 &= \mathbf{a} * \mathbf{b} &&= (2,1)*(3,2) = (6,7,2), && C_0(z) = A(z)B(z)\\ \mathbf{c}_1 &= \mathbf{a}^R * \mathbf{b} &&= (1,2)*(3,2) = (3,8,4), && C_1(z) = A^R(z)B(z)\\ \mathbf{c}_2 &= \mathbf{a} * \mathbf{b}^R &&= (2,1)*(2,3) = (4,8,3), && C_2(z) = A(z)B^R(z)\\ \mathbf{c}_3 &= \mathbf{a}^R * \mathbf{b}^R &&= (1,2)*(2,3) = (2,7,6), && C_3(z) = A(z)B(z)\end{aligned}$$

All four sequences have the same magnitude spectrum.

Partial Energy and Minimal Delay

Since the total energy of a sequence $\mathbf{a} = (a_0, a_1, \dots, a_M)$ is given by Parseval's equality

$$\sum_{m=0}^{M} |a_m|^2 = \int_{-\pi}^{\pi} |A(\omega)|^2 \frac{d\omega}{2\pi}$$

it follows that all of the above 2^M sequences, having the same magnitude spectrum, will also have the same *total energy*. However, the *distribution* of the total energy over time may be different. And this will allow an alternative characterization of the minimum phase sequences, first given by Robinson. Define the partial energy by

$$P_a(n) = \sum_{m=0}^{n} |a_m|^2 = |a_0|^2 + |a_1|^2 + \cdots + |a_n|^2, \quad n = 0, 1, \dots, M$$

then, for the above example, the partial energies for the four different sequences are given in the table

	$\mathbf{c}_0$	$\mathbf{c}_1$	$\mathbf{c}_2$	$\mathbf{c}_3$
$P(0)$	36	9	16	4
$P(1)$	85	73	80	53
$P(2)$	89	89	89	89

We note that $\mathbf{c}_0$ which has both its zeros inside the unit circle (i.e., minimal phase) is also the sequence that has most of its energy concentrated at the *earlier* times, that is, it makes its impact as early as possible, with *minimal delay*. In contrast, the maximal-phase sequence $\mathbf{c}_3$ has most of its energy concentrated at its tail thus, making most of its impact at the end, with maximal delay.

Invariance of the Autocorrelation Function

This section presents yet another characterization of the above class of sequences. It will be important in proving the minimum-phase property of the linear prediction filters.

The *sample autocorrelation* of a (possibly complex-valued) sequence $\mathbf{a} = (a_0, a_1, \dots, a_M)$ is defined by

$$\begin{aligned}R_{aa}(k) &= \sum_{n=0}^{M-k} a_{n+k} a_n^*, \quad \text{for} \quad 0 \le k \le M\\ R_{aa}(k) &= R_{aa}(-k)^*, \quad \text{for} \quad -M \le k \le -1\end{aligned} \tag{1.19.5}$$

It is easily verified that the corresponding power spectral density is factored as

$$S_{aa}(z) = \sum_{k=-M}^{M} R_{aa}(k) z^{-k} = A(z)\bar{A}(z) \tag{1.19.6}$$

The magnitude response is obtained by setting $z = e^{j\omega}$

$$S_{aa}(\omega) = |A(\omega)|^2 \tag{1.19.7}$$

with an inversion formula

$$R_{aa}(k)=\int_{-\pi}^{\pi}|A(\omega)|^2 e^{j\omega k}\,\frac{d\omega}{2\pi} \tag{1.19.8}$$

It follows from Eq. (1.19.8) that the above 2^M different sequences having the same magnitude spectrum, also have the same sample *autocorrelation*. They cannot be distinguished on the basis of their autocorrelation. Therefore, there are 2^M different spectral factorizations of $S_{aa}(z)$ of the form

$$S_{aa}(z)=A(z)\bar{A}(z) \tag{1.19.9}$$

but there is only one with minimum-phase factors. The procedure for obtaining it is straightforward: Find the zeros of $S_{aa}(z)$, which come in pairs z_i and $1/z_i^*$, thus, there are $2M$ such zeros. And, group those that lie inside the unit circle into a common factor. This defines $A(z)$ as a minimum phase polynomial.

Minimum-Delay Property

Here, we discuss the effect of flipping a zero from the inside to the outside of the unit circle, on the minimum-delay and minimum-phase properties of the signal. Suppose $A(z)$ is of degree M and has a zero z_1 inside the unit circle. Let $B(z)$ be the polynomial that results by flipping this zero to the outside; that is, $z_1 \to 1/z_1^*$

$$\begin{aligned} A(z) &= (1-z_1z^{-1})F(z) \\ B(z) &= (-z_1^*+z^{-1})F(z) \end{aligned} \tag{1.19.10}$$

where $F(z)$ is a polynomial of degree $M-1$. Both $A(z)$ and $B(z)$ have the same magnitude spectrum. We may think of this operation as sending $A(z)$ through an *allpass* filter

$$B(z)=\frac{-z_1^*+z^{-1}}{1-z_1z^{-1}}\,A(z)$$

In terms of the polynomial coefficients, Eq. (1.19.10) becomes

$$\begin{aligned} a_n &= f_n - z_1 f_{n-1} \\ b_n &= -z_1^* f_n + f_{n-1} \end{aligned} \tag{1.19.11}$$

for $n=0,1,\dots,M$, from which we obtain

$$|a_n|^2-|b_n|^2=\big(1-|z_1|^2\big)\big(|f_n|^2-|f_{n-1}|^2\big) \tag{1.19.12}$$

Summing to get the partial energies, $P_a(n)=\sum_{m=0}^{n}|a_m|^2$, we find

$$P_a(n)-P_b(n)=\big(1-|z_1|^2\big)|f_n|^2\,,\quad n=0,1,\dots,M \tag{1.19.13}$$

Thus, the partial energy of the sequence **a** remains greater than that of **b** for all times n; that is, $A(z)$ is of earlier delay than $B(z)$. The total energy is, of course, the same as follows from the fact that $F(z)$ is of degree $M-1$, thus, missing the Mth term or $f_M=0$. We have then

$$P_a(n)\ge P_b(n)\,,\quad n=0,1,\dots,M$$

and in particular

$$P_a(M)=P_b(M)\quad\text{and}\quad P_a(0)\ge P_b(0)$$

The last inequality can also be stated as $|a_0|\ge|b_0|$, and will be important in our proof of the minimum-phase property of the prediction-error filter of linear prediction.

Minimum-Phase Property

The effect of reversing the zero z_1 on the phase responses of $A(z)$ and $B(z)$ of Eq. (1.19.10) can be seen as follows. For $z=e^{j\omega}$, define the *phase lag* as the negative of the phase response

$$\begin{aligned} A(\omega) &= |A(\omega)|\,e^{j\mathrm{Arg}(\omega)} \\ \theta_A(\omega) &= -\mathrm{Arg}(\omega)=\text{phase-lag response} \end{aligned}$$

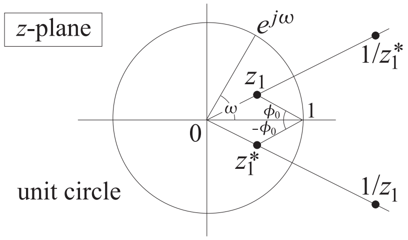
and similarly for $B(z)$. Since $A(\omega)$ and $B(\omega)$ have the same magnitude, they will differ only by a phase

$$\frac{A(\omega)}{B(\omega)}=e^{j(\theta_B-\theta_A)}=\frac{1-z_1e^{-j\omega}}{-z_1^*+e^{-j\omega}}=\frac{e^{j\omega}-z_1}{1-z_1^*e^{j\omega}}=e^{j\phi(\omega)}$$

where $\phi(\omega)$ is the phase-response of the all-pass factor $(e^{j\omega}-z_1)/(1-z_1^*e^{j\omega})$, so that $\theta_B(\omega)-\theta_A(\omega)=\phi(\omega)$. By taking derivatives with respect to ω in the above definition of $\phi(\omega)$, it can be easily shown that

$$\frac{d\phi(\omega)}{d\omega}=\frac{1-|z_1|^2}{\big|e^{j\omega}-z_1\big|^2}>0$$

which shows that $\phi(\omega)$ is an increasing function of ω. Thus, over the frequency interval $0\le\omega\le\pi$, we have $\phi(\omega)\ge\phi(0)$. It can be verified easily that $\phi(0)=-2\phi_0$, where ϕ_0 is the angle with the x-axis of the line between the points z_1 and 1, as shown in the figure below.



Thus, we have $\theta_B-\theta_A\ge\phi\ge-2\phi_0$. The angle ϕ_0 is positive, if z_1 lies within the upper half semi-circle, and negative, if it lies in the lower semi-circle; and, ϕ_0 is zero if z_1 lies on the real axis. If z_1 is real-valued, then $\theta_B\ge\theta_A$ for $0\le\omega\le\pi$. If z_1

is complex valued and we consider the combined effect of flipping the zero z_1 and its conjugate z_1^*, that is, $A(z)$ and $B(z)$ are given by

$$A(z) = (1 - z_1 z^{-1})(1 - z_1^* z^{-1})F(z)$$
$$B(z) = (-z_1^* + z^{-1})(-z_1 + z^{-1})F(z)$$

then, for the phase of the combined factor

$$e^{j\phi(\omega)} = \frac{e^{j\omega} - z_1}{1 - z_1^* e^{j\omega}} \cdot \frac{e^{j\omega} - z_1^*}{1 - z_1 e^{j\omega}}$$

we will have $\phi(\omega) \geq (-2\phi_0) + (2\phi_0) = 0$, so that $\theta_B(\omega) - \theta_A(\omega) = \phi(\omega) \geq 0$.

Thus, the phase lag of $A(z)$ remains smaller than that of $B(z)$. The phase-lag curve for the case when $A(z)$ has all its zeros inside the unit circle will remain below all the other phase-lag curves. The term *minimum-phase* strictly speaking means minimum phase lag (over $0 \leq \omega \leq \pi$).

1.20 Spectral Factorization Theorem

We finish our digression on minimum-phase sequences by quoting the spectral factorization theorem [5].

Any *rational* power spectral density $S_{yy}(z)$ of a (real-valued) stationary signal y_n can be factored in a minimum-phase form

$$S_{yy}(z) = \sigma_\epsilon^2 B(z) B(z^{-1}) \tag{1.20.1}$$

where

$$B(z) = \frac{N(z)}{D(z)} \tag{1.20.2}$$

with both $D(z)$ and $N(z)$ being minimum-phase polynomials; that is, having all their zeros inside the unit circle. By adjusting the overall constant σ_ϵ^2, both $D(z)$ and $N(z)$ may be taken to be *monic* polynomials. Then, they are *unique*.

This theorem guarantees the existence of a *causal and stable* random signal generator filter $B(z)$ for the signal y_n of the type discussed in Sec. 1.13:

$$\epsilon_n \longrightarrow \boxed{B(z)} \longrightarrow y_n$$

with ϵ_n white noise of variance σ_ϵ^2. The minimum-phase property of $B(z)$ also guarantees the stability and causality of the inverse filter $1/B(z)$, that is, the whitening filter

$$y_n \longrightarrow \boxed{1/B(z)} \longrightarrow \epsilon_n = \text{white noise}$$

The proof of the spectral factorization theorem is straightforward. Since $S_{yy}(z)$ is the power spectral density of a (real-valued) stationary process y_n, it will satisfy the symmetry conditions $S_{yy}(z) = S_{yy}(z^{-1})$. Therefore, if z_i is a zero then $1/z_i$ is also a zero, and if z_i is complex then the reality of $R_{yy}(k)$ implies that z_i^* will also be a zero. Thus, both z_i and $1/z_i^*$ are zeros. Therefore, the numerator polynomial of $S_{yy}(z)$ is of the type of Eq. (1.19.9) and can be factored into its minimum phase polynomials $N(z)N(z^{-1})$. This is also true of the denominator of $S_{yy}(z)$.

All sequential correlations in the original signal y_n arise from the *filtering* action of $B(z)$ on the white-noise input ϵ_n. This follows from Eq. (1.12.14):

$$R_{yy}(k) = \sigma_\epsilon^2 \sum_n b_{n+k} b_n\,, \quad B(z) = \sum_{n=0}^{\infty} b_n z^{-n} \tag{1.20.3}$$

Effectively, we have modeled the statistical autocorrelation $R_{yy}(k)$ by the sample autocorrelation of the impulse response of the synthesis filter $B(z)$. Since $B(z)$ is causal, such factorization corresponds to an LU, or Cholesky, factorization of the autocorrelation matrix.

This matrix representation can be seen as follows: Let B be the lower triangular Toeplitz matrix defined exactly as in Eq. (1.13.2)

$$b_{ni} = b_{n-i}$$

and let the autocorrelation matrix of y_n be

$$R_{yy}(i,j) = R_{yy}(i-j)$$

Then, the transposed matrix B^T will have matrix elements

$$(B^T)_{ni} = b_{i-n}$$

and Eq. (1.20.3) can be written in the form

$$\begin{aligned} R_{yy}(i,j) &= R_{yy}(i-j) = \sigma_\epsilon^2 \sum_n b_{n+i-j} b_n = \sigma_\epsilon^2 \sum_k b_{i-k} b_{j-k} \\ &= \sigma_\epsilon^2 \sum_k (B)_{ik} (B^T)_{kj} = \sigma_\epsilon^2 (BB^T)_{ij} \end{aligned}$$

Thus, in matrix notation

$$R_{yy} = \sigma_\epsilon^2 B B^T \tag{1.20.4}$$

This equation is a special case of the more general LU factorization of the Gram-Schmidt construction given by Eq. (1.6.17). Indeed, the assumption of stationarity implies that the quantity

$$\sigma_\epsilon^2 = E[\epsilon_n^2]$$

is independent of the time n, and therefore, the diagonal matrix $R_{\epsilon\epsilon}$ of Eq. (1.6.17) becomes a multiple of the identity matrix.

1.21 Minimum-Phase Property of the Prediction-Error Filter

The minimum-phase property of the prediction-error filter $A(z)$ of linear prediction is an important property because it guarantees the stability of the causal inverse synthesis filter $1/A(z)$. There are many proofs of this property in the literature [6–10]. Here, we would like to present a simple proof [11] which is based directly on the fact that the optimal prediction coefficients minimize the mean-square prediction error. Although we have only discussed first and second order linear predictors, for the purposes of this proof we will work with the more general case of an Mth order predictor defined by

$$\hat{y}_n = -[a_1 y_{n-1} + a_2 y_{n-2} + \cdots + a_M y_{n-M}]$$

which is taken to represent the best prediction of y_n based on the past M samples $Y_n = \{y_{n-1}, y_{n-2}, \dots, y_{n-M}\}$. The corresponding prediction error is

$$e_n = y_n - \hat{y}_n = y_n + a_1 y_{n-1} + a_2 y_{n-2} + \cdots + a_M y_{n-M}$$

The best set of prediction coefficients $\{a_1, a_2, \dots a_M\}$ is found by minimizing the mean-square prediction error

$$\begin{aligned} \mathcal{E}(a_1, a_2, \dots a_M) &= E[e_n^* e_n] = \sum_{m,k=0}^{M} a_m^* E[y_{n-m}^* y_{n-k}] a_k \\ &= \sum_{m,k=0}^{M} a_m^* R_{yy}(k-m) a_k = \sum_{m,k=0}^{M} a_m^* R_{yy}(m-k) a_k \end{aligned} \tag{1.21.1}$$

where we set $a_0 = 1$. For the proof of the minimum phase property, we do not need the explicit solution of this minimization problem; we only use the fact that the optimal coefficients minimize Eq. (1.21.1). The key to the proof is based on the observation that (3.7.1) can be written in the alternative form

$$\mathcal{E}(\mathbf{a}) = \sum_{k=-M}^{M} R_{yy}(k) R_{aa}(k) \tag{1.21.2}$$

where $R_{aa}(k)$ is the sample autocorrelation of the prediction-error filter sequence $\mathbf{a} = [1, a_1, a_2, \dots, a_M]^T$ as defined in Eq. (1.19.5). The equivalence of Eqs. (1.21.1) and (1.21.2) can be seen easily, either by rearranging the summation indices of (1.21.1), or by using the results of Problems 1.37 and 1.39.

Example 1.21.1: We demonstrate this explicitly for the $M = 2$ case. Using the definition (1.19.5) we have

$$\begin{aligned} R_{aa}(0) &= |a_0|^2 + |a_1|^2 + |a_2|^2 = 1 + |a_1|^2 + |a_2|^2 \\ R_{aa}(1) &= R_{aa}(-1)^* = a_1 a_0^* + a_2 a_1^* = a_1 + a_2 a_1^* \\ R_{aa}(2) &= R_{aa}(-2)^* = a_2 a_0^* = a_2 \end{aligned}$$

Since y_n is real-valued stationary, we have $R_{yy}(k) = R_{yy}(-k)$. Then, Eq. (1.21.1) becomes explicitly

$$\mathcal{E}(\mathbf{a}) = \sum_{m,k=0}^{M} a_m^* R_{yy}(m-k) a_k = [1, a_1^*, a_2^*] \begin{bmatrix} R_{yy}(0) & R_{yy}(1) & R_{yy}(2) \\ R_{yy}(1) & R_{yy}(0) & R_{yy}(1) \\ R_{yy}(0) & R_{yy}(1) & R_{yy}(2) \end{bmatrix} \begin{bmatrix} 1 \\ a_1 \\ a_2 \end{bmatrix}$$

$$= R_{yy}(0)[1 + a_1^* a_1 + a_2^* a_2] + R_{yy}(1)\big[(a_1 + a_2 a_1^*) + (a_1^* + a_2^* a_1)\big] + R_{yy}(2)[a_2 + a_2^*]$$

$$= R_{yy}(0) R_{aa}(0) + R_{yy}(1)\big[R_{aa}(1) + R_{aa}(-1)\big] + R_{yy}(2)\big[R_{aa}(2) + R_{aa}(-2)\big] \qquad \square$$

Let $\mathbf{a} = [1, a_1, a_2, \dots, a_M]^T$ be the optimal set of coefficients that minimizes $\mathcal{E}(\mathbf{a})$ and let z_i, $i = 1, 2 \dots, M$, be the zeros of the corresponding prediction-error filter:

$$1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_M z^{-M} = (1 - z_1 z^{-1})(1 - z_2 z^{-1}) \cdots (1 - z_M z^{-1}) \tag{1.21.3}$$

Reversing any one of the zero factors in this equation, that is, replacing $(1 - z_i z^{-1})$ by its reverse $(-z_i^* + z^{-1})$, results in a sequence that has the same sample autocorrelation as $\mathbf{a}$. As we have seen, there are 2^M such sequences, all with the same sample autocorrelation. We would like to show that among these, $\mathbf{a}$ is the one having the minimum-phase property.

To this end, let $\mathbf{b} = [b_0, b_1, \dots b_M]^T$ be any one of these 2^M sequences, and define the normalized sequence

$$\mathbf{c} = \mathbf{b}/b_0 = [1, b_1/b_0, b_2/b_0, \dots b_M/b_0]^T \tag{1.21.4}$$

Using the fact that $\mathbf{b}$ has the same sample autocorrelation as $\mathbf{a}$, we find for the sample autocorrelation of $\mathbf{c}$:

$$R_{cc}(k) = R_{bb}(k)/|b_0|^2 = R_{aa}(k)/|b_0|^2 \tag{1.21.5}$$

The performance index (1.21.2) evaluated at $\mathbf{c}$ is then

$$\mathcal{E}(\mathbf{c}) = \sum_{k=-M}^{M} R_{yy}(k) R_{cc}(k) = \sum_{k=-M}^{M} R_{yy}(k) R_{aa}(k)/|b_0|^2 \tag{1.21.6}$$

or,

$$\mathcal{E}(\mathbf{c}) = \mathcal{E}(\mathbf{a})/|b_0|^2 \tag{1.21.7}$$

Since $\mathbf{a}$ minimizes $\mathcal{E}$, it follows that $\mathcal{E}(\mathbf{c}) \geq \mathcal{E}(\mathbf{a})$. Therefore, Eq. (1.21.7) implies that

$$|b_0| \leq 1 \tag{1.21.8}$$

This must be true of all $\mathbf{b}$s in the above class. Eq. (1.21.8) then, immediately implies the minimum-phase property of $\mathbf{a}$. Indeed, choosing $\mathbf{b}$ to be that sequence obtained from (1.21.3) by reversing only the ith zero factor $(1 - z_i z^{-1})$ and not the other zero factors, it follows that

$$b_0 = -z_i^*$$

and therefore Eq. (1.21.8) implies that

$$|z_i| \leq 1 \tag{1.21.9}$$

which shows that all the zeros of $A(z)$ are inside the unit circle and thus, $A(z)$ has minimum phase. An alternative proof based on the Levinson recursion and Rouche's theorem of complex analysis will be presented in Chap. 12.

1.22 Computer Project – Adaptive AR(1) and AR(2) Models

This computer project, divided into separate parts, deals with adaptive AR models that are capable of tracking time-varying systems. It is also applied to the benchmark sunspot data, comparing the results with Yule's original application of an AR(2) model.

1. *Time-varying AR(1) model.* Consider the following AR(1), first-order, autoregressive signal model with a time-varying parameter:

$$y_n = a(n)y_{n-1} + \epsilon_n \tag{1.22.1}$$

where ϵ_n is zero-mean, unit-variance, white noise. The filter parameter $a(n)$ can be tracked by the following adaptation equations (which are equivalent to the exact recursive least-squares order-1 adaptive predictor):

$$\begin{aligned} R_0(n) &= \lambda R_0(n-1) + \alpha\, y_{n-1}^2 \\ R_1(n) &= \lambda R_1(n-1) + \alpha\, y_n y_{n-1} \\ \hat{a}(n) &= \frac{R_1(n)}{R_0(n)} \end{aligned}$$

where $\alpha = 1-\lambda$. The two filtering equations amount to sending the quantities y_{n-1}^2 and $y_n y_{n-1}$ through an exponential smoother. To avoid possible zero denominators, initialize R_0 to some small positive constant, $R_0(-1) = \delta$, such as $\delta = 10^{-3}$.

(a) Show that $\hat{a}(n)$ satisfies the recursion:

$$\boxed{\hat{a}(n) = \hat{a}(n-1) + \frac{\alpha}{R_0(n)} y_{n-1} e_{n/n-1}} \qquad e_{n/n-1} = y_n - \hat{a}(n-1)y_{n-1} \tag{1.22.2}$$

where $e_{n/n-1}$ is referred to as the a priori estimation (prediction) error.

(b) Using Eq. (1.22.1), generate a data sequence y_n, $n = 0, 1, \dots, N-1$ using the following time varying coefficient, sinusoidally switching from a positive value to a negative one (the synthesis filter switches from lowpass to highpass):

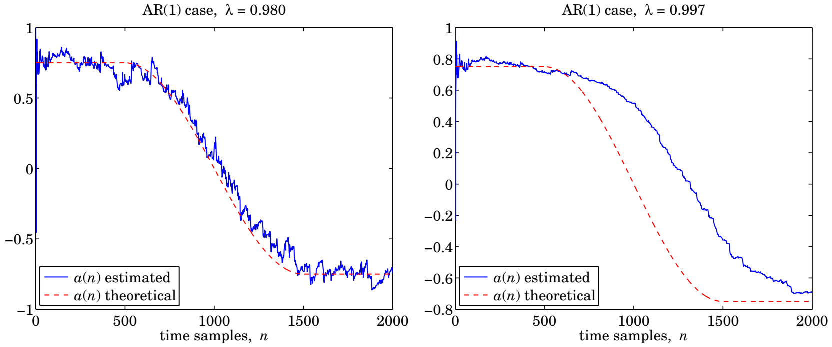
$$a(n) = \begin{cases} 0.75, & 0 \le n \le N_a - 1 \\ 0.75\cos\left(\pi\dfrac{n-N_a}{N_b-N_a}\right), & N_a \le n \le N_b \\ -0.75, & N_b + 1 \le n \le N-1 \end{cases}$$

Use the following numerical values:

$$N_a = 500, \quad N_b = 1500, \quad N = 2000$$

Calculate the estimated $\hat{a}(n)$ using the recursion (1.22.2) and plot it versus n together with the theoretical $a(n)$ using the parameter value $\lambda = 0.980$. Repeat using the value $\lambda = 0.997$. Comment on the tracking capability of the algorithm versus the accuracy of the estimate.

(c) Study the sensitivity of the algorithm to the initialization parameter δ.



2. *Tine-varying AR(2) model.* Next, consider an AR(2), second-order, model with time-varying coefficients:

$$y_n = -a_1(n)y_{n-1} - a_2(n)y_{n-2} + \epsilon_n \tag{1.22.3}$$

If the coefficients were stationary, then the theoretical Wiener solution for the prediction coefficients a_1 and a_2 would be:

$$\begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = -\begin{bmatrix} R_0 & R_1 \\ R_1 & R_0 \end{bmatrix}^{-1} \begin{bmatrix} R_1 \\ R_2 \end{bmatrix} = -\frac{1}{R_0^2 - R_1^2} \begin{bmatrix} R_0R_1 - R_1R_2 \\ R_0R_2 - R_1^2 \end{bmatrix} \tag{1.22.4}$$

where $R_k = E[y_n y_{n-k}]$. For a time-varying model, the coefficients can be tracked by replacing the theoretical autocorrelation lags R_k with their recursive, exponentially smoothed, versions:

$$\begin{aligned} R_0(n) &= \lambda R_0(n-1) + \alpha\, y_n^2 \\ R_1(n) &= \lambda R_1(n-1) + \alpha\, y_n y_{n-1} \\ R_2(n) &= \lambda R_2(n-1) + \alpha\, y_n y_{n-2} \end{aligned}$$

(a) Using Eq. (1.22.3), generate a non-stationary data sequence y_n by driving the second-order model with a unit-variance, zero-mean, white noise signal ϵ_n and using the following theoretical time-varying coefficients:

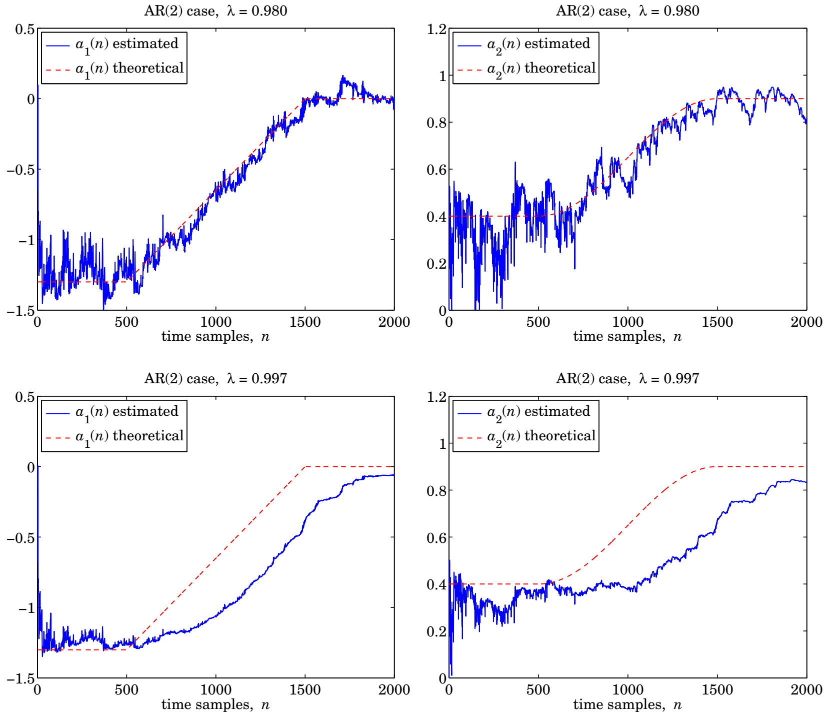
$$a_1(n) = \begin{cases} -1.3, & 0 \le n \le N_a - 1 \\ 1.3\dfrac{n-N_b}{N_b-N_a}, & N_a \le n \le N_b \\ 0, & N_b + 1 \le n \le N-1 \end{cases}$$

$$a_2(n) = \begin{cases} 0.4, & 0 \le n \le N_a - 1 \\ 0.65 - 0.25\cos\left(\pi\dfrac{n-N_a}{N_b-N_a}\right), & N_a \le n \le N_b \\ 0.9, & N_b + 1 \le n \le N-1 \end{cases}$$

Thus, the signal model for y_n switches continuously between the synthesis filters:

$$B(z)=\frac{1}{1-1.3z^{-1}+0.4z^{-2}} \quad\Rightarrow\quad B(z)=\frac{1}{1+0.9z^{-2}}$$

(b) Compute the adaptive coefficients $\hat{a}_1(n)$ and $\hat{a}_2(n)$ using the two forgetting factors $\lambda = 0.980$ and $\lambda = 0.997$. Plot the adaptive coefficients versus n, together with the theoretical time-varying coefficients and discuss the tracking capability of the adaptive processor.



3. *AR(2) modeling of sunspot data.* Next, we will apply the adaptive method of part-2 to some real data. The file `sunspots.dat` contains the yearly mean number of sunspots for the years 1700–2008. To unclutter the resulting graphs, we will use only the data for the last 200 years, over 1809–2008. These can be read into MATLAB as follows:

```
Y = loadfile('sunspots.dat');
i = find(Y(:,1)==1809);
y = Y(i:end,2);                % number of sunspots
N = length(y);                 % here, N=200
m = mean(y); y = y-m;          % zero-mean data
```

where the last line determines the mean of the data block and subtracts it from the data. The mean m will be restored at the end.

Yule was the first to introduce the concept of an autoregressive signal model and applied it to the sunspot time series assuming a second-order model. The so-called Yule-Walker method is a block processing method in which the entire (zero-mean) data block is used to estimate the autocorrelation lags R_0, R_1, R_2 using sample autocorrelations:

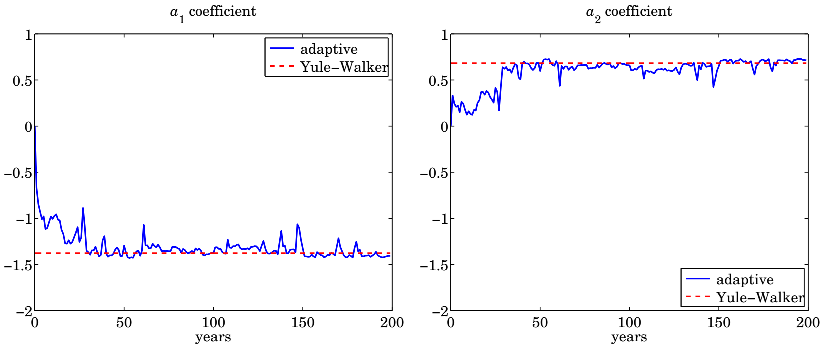
$$\hat{R}_0=\frac{1}{N}\sum_{n=0}^{N-1}y_n^2\,,\quad \hat{R}_1=\frac{1}{N}\sum_{n=0}^{N-2}y_{n+1}y_n\,,\quad \hat{R}_2=\frac{1}{N}\sum_{n=0}^{N-3}y_{n+2}y_n$$

Then, the model parameters a_1, a_2 are estimated using Eq. (1.22.4):

$$\begin{bmatrix}\hat{a}_1\\ \hat{a}_2\end{bmatrix}=-\begin{bmatrix}\hat{R}_0 & \hat{R}_1\\ \hat{R}_1 & \hat{R}_0\end{bmatrix}^{-1}\begin{bmatrix}\hat{R}_1\\ \hat{R}_2\end{bmatrix}\qquad \text{(Yule-Walker method)}$$

(a) First, compute the values of $\hat{a}_1, \hat{a}_2$ based on the given length-200 data block.

(b) Then, apply the adaptive algorithm of the part-2 with $\lambda = 0.99$ to determine the adaptive versions $a_1(n), a_2(n)$ and plot them versus n, and add on these graphs the straight lines corresponding to the Yule-Walker estimates $\hat{a}_1, \hat{a}_2$.

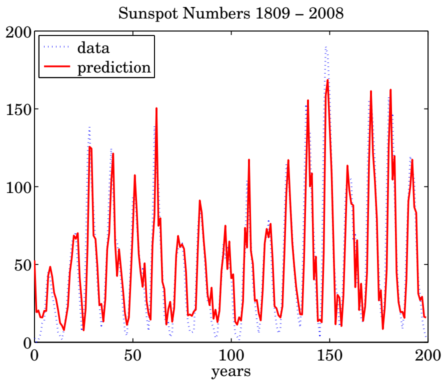


(c) At each time instant n, the value of y_n can be predicted by either of the two formulas:

$$\hat{y}_{n/n-1} = -a_1(n)y_{n-1} - a_2(n)y_{n-2}$$

$$\hat{y}_{n/n-1} = -\hat{a}_1 y_{n-1} - \hat{a}_2 y_{n-2}$$

On the same graph, plot y_n and $\hat{y}_{n/n-1}$ for the above two alternatives. The case of the adaptive predictor is shown below.



(d) Repeat the above questions using $\lambda = 0.95$ and discuss the effect of reducing λ.

(e) Apply a length-200 Hamming window w_n to the (zero-mean) data y_n and calculate the corresponding periodogram spectrum,

$$S_{\text{per}}(\omega)=\frac{1}{N}\left|\sum_{n=0}^{N-1} w_n y_n e^{-j\omega n}\right|^2$$

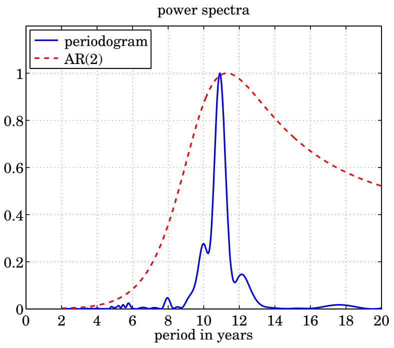
as a function of the yearly period $p = 2\pi/\omega$, over the range $2 \le p \le 20$ years. For the same p's or ω's calculate also the AR(2) spectrum using the Yule-Walker coefficients $\hat{a}_1, \hat{a}_2$:

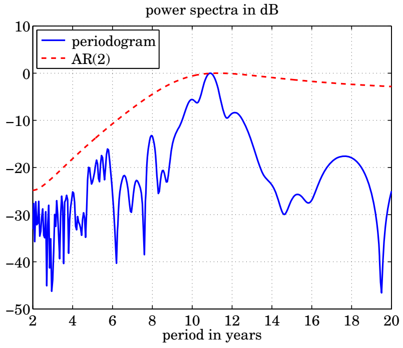
$$S_{\text{AR}}(\omega)=\frac{\sigma_\epsilon^2}{\left|1+\hat{a}_1 e^{-j\omega}+\hat{a}_2 e^{-2j\omega}\right|^2}$$

where σ_ϵ^2 can be calculated by

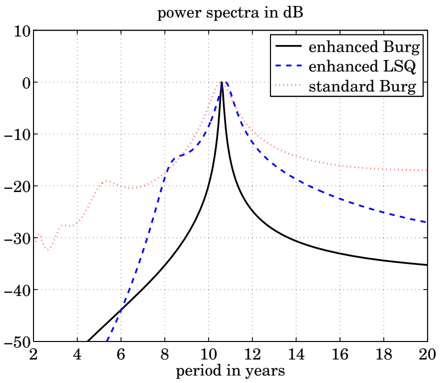
$$\sigma_\epsilon^2 = \hat{R}_0 + \hat{a}_1\hat{R}_1 + \hat{a}_2\hat{R}_2$$

Normalize the spectra $S_{\text{per}}(\omega), S_{\text{AR}}(\omega)$ to unity maxima and plot them versus period p on the same graph. Note that both predict the presence of an approximate 11-year cycle, which is also evident from the time data.





We will revisit this example later on by applying SVD methods to get sharper peaks. An example of the improved results is shown below.



1.23 Problems

1.1 Two dice are available for throwing. One is fair, but the other bears only sixes. One die is selected as follows: A coin is tossed. If the outcome is tails then the fair die is selected, but if the outcome is heads, the biased die is selected. The coin itself is not fair, and the probability of bearing heads or tails is $1/3$ or $2/3$, respectively. A die is now selected according to this procedure and tossed twice and the number of sixes is noted.

Let x be a random variable that takes on the value 0 when the fair die is selected or 1 if the biased die is selected. Let y be a random variable denoting the number of sixes obtained in the two tosses; thus, the possible values of y are $0, 1, 2$.

(a) For all possible values of x and y, compute $p(y|x)$, that is, the probability that the number of sixes will be y, given that the x die was selected.

(b) For each y, compute $p(y)$, that is, the probability that the number of sixes will be y, regardless of which die was selected.

(c) Compute the mean number of sixes $E[y]$.

(d) For all values of x and y, compute $p(x|y)$, that is, the probability that we selected die x, given that we already observed a y number of sixes.

1.2 *Inversion Method.* Let $F(x)$ be the cumulative distribution of a probability density $p(x)$. Suppose u is a uniform random number in the interval $[0, 1)$. Show that the solution of the equation $F(x) = u$, or equivalently, $x = F^{-1}(u)$, generates a random number x distributed according to $p(x)$. This is the inversion method of generating random numbers from uniform random numbers.

1.3 *Computer Experiment.* Let x be a random variable with the exponential probability density

$$p(x)=\frac{1}{\mu}e^{-x/\mu}$$

Show that x has mean μ and variance μ^2. Determine the cumulative distribution function $F(x)$ of x. Determine the inverse formula $x = F^{-1}(u)$ for generating x from a uniform

u. Take $\mu = 2$. Using the inversion formula and a uniform random number generator routine, generate a block of 200 random numbers x distributed according to $p(x)$. Compute their sample mean and sample variance, Eqs. (1.2.1) and (1.2.3), and compare them with their theoretical values. Do the estimated values fall within the standard deviation intervals defined by Eqs. (1.2.2) and (1.2.4)?

1.4 The Rayleigh probability density finds application in fading communication channels

$$p(r) = \frac{r}{\sigma^2} e^{-r^2/2\sigma^2}\,, \quad r \geq 0$$

Using the inversion method, $r = F^{-1}(u)$, show how to generate a Rayleigh-distributed random variable r from a uniform u.

1.5 (a) Following the notation of Sec. 1.4, show the matrix identity, where $H = R_{xy}R_{yy}^{-1}$

$$\left[\begin{array}{c|c} I_N & -H \\ \hline 0 & I_M \end{array}\right]\left[\begin{array}{c|c} R_{xx} & R_{xy} \\ \hline R_{yx} & R_{yy} \end{array}\right]\left[\begin{array}{c|c} I_N & -H \\ \hline 0 & I_M \end{array}\right]^T = \left[\begin{array}{c|c} R_{xx} - R_{xy}R_{yy}^{-1}R_{yx} & 0 \\ \hline 0 & R_{yy} \end{array}\right]$$

(b) Rederive the correlation canceling results of Eqs. (1.4.3) and (1.4.4) using this identity.

1.6 Using the matrix identity of Problem 1.5, derive directly the result of Example 1.4.1, that is, $E[\mathbf{x}|\mathbf{y}] = R_{xy}R_{yy}^{-1}\mathbf{y}$. Work directly with probability densities;

1.7 Show that the orthogonal projection $\hat{\mathbf{x}}$ of a vector $\mathbf{x}$ onto another vector $\mathbf{y}$, defined by Eq. (1.4.5) or Eq. (1.6.18), is a linear function of $\mathbf{x}$, that is, show

$$\widehat{A_1\mathbf{x}_1 + A_2\mathbf{x}_2} = A_1\hat{\mathbf{x}}_1 + A_2\hat{\mathbf{x}}_2$$

1.8 Suppose x consists of two components $x = s + n_1$, a desired component s, and a noise component n_1. Suppose that y is a related noise component n_2 to which we have access, $y = n_2$. The relationship between n_1 and n_2 is assumed to be linear, $n_1 = Fn_2$. For example, s might represent an electrocardiogram signal which is contaminated by 60 Hz power frequency pick-up noise n_1; then, a reference 60 Hz noise $y = n_2$, can be obtained from the wall outlet.

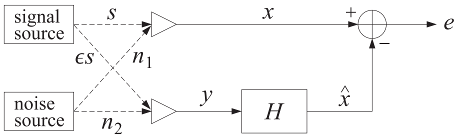
(a) Show that the correlation canceler is $H = F$, and that complete cancellation of n_1 takes place.

(b) If $n_1 = Fn_2 + v$, where v is uncorrelated with n_2 and s, show that $H = F$ still, and n_1 is canceled completely. The part v remains unaffected.

1.9 *Signal Cancellation Effects.* In the previous problem, we assumed that the reference signal y did not contain any part related to the desired component s. There are applications, however, where both the signal and the noise components contribute to both x and y, as for example in antenna sidelobe cancellation. Since the reference signal y contains part of s, the correlation canceler will act also to cancel part of the useful signal s from the output. To see this effect, consider a simple one-dimensional example

$$x = s + n_1$$

$$y = n_2 + \epsilon s$$



with $n_1 = Fn_2$, where we assume that y contains a small part proportional to the desired signal s. Assume that n_2 and s are uncorrelated. Show that the output e of the correlation canceler will contain a reduced noise component n_1 as well as a partially canceled signal s, as follows:

$$e = as + bn_1\,, \quad \text{where} \quad a = 1 - \frac{F\epsilon(1 + F\epsilon G)}{1 + F^2\epsilon^2 G}\,, \quad b = -\epsilon F G a$$

and G is a signal to noise ratio $G = E[s^2]/E[n_1^2]$. Note that when $\epsilon = 0$, then $a = 1$ and $b = 0$, as it should.

1.10 Consider a special case of Example 1.4.3 defined by $c_n = 1$, so that $y_n = x + v_n$, $n = 1, 2, \dots, M$. This represents the noisy measurement of a constant x. By comparing the corresponding mean-square estimation errors $E[e^2]$, show that the optimal estimate of x given in Eq. (1.4.9) is indeed better than the straight average estimate:

$$\hat{x}_{\text{av}} = \frac{y_1 + y_2 + \cdots + y_M}{M}$$

1.11 *Recursive Estimation.* Consider the subspace $Y_n = \{y_1, y_2, \dots, y_n\}$ for $n = 1, 2, \dots, M$, as defined in Sec. 1.6. Eq. (1.6.18) defines the estimate $\hat{\mathbf{x}}$ of a random vector $\mathbf{x}$ based on the largest one of these subspaces, namely, Y_M.

(a) Show that this estimate can also be generated recursively as follows:

$$\hat{\mathbf{x}}_n = \hat{\mathbf{x}}_{n-1} + \mathbf{G}_n(y_n - \hat{y}_{n/n-1})$$

for $n = 1, 2, \dots, M$, and initialized by $\hat{\mathbf{x}}_0 = 0$ and $\hat{y}_{1/0} = 0$, where $\hat{\mathbf{x}}_n$ denotes the best estimate of $\mathbf{x}$ based on the subspace Y_n and $\mathbf{G}_n$ is a gain coefficient given by $\mathbf{G}_n = E[\mathbf{x}\epsilon_n]E[\epsilon_n\epsilon_n]^{-1}$. (*Hint:* Note $\hat{\mathbf{x}}_n = \sum_{i=1}^{n} E[\mathbf{x}\epsilon_i]E[\epsilon_i\epsilon_i]^{-1}\epsilon_i$.)

(b) Show that the innovations $\epsilon_n = y_n - \hat{y}_{n/n-1}$ is orthogonal to $\hat{\mathbf{x}}_{n-1}$, that is, show that $E[\hat{\mathbf{x}}_{n-1}\epsilon_n] = 0$ for $n = 1, 2, \dots, M$.

(c) Let $\mathbf{e}_n = \mathbf{x} - \hat{\mathbf{x}}_n$ be the corresponding estimation error of $\mathbf{x}$ with respect to the subspace Y_n. Using Eq. (1.4.4), show that its covariance matrix can be expressed in the ϵ-basis as follows

$$R_{e_n e_n} = R_{xx} - \sum_{i=1}^{n} E[\mathbf{x}\epsilon_i]E[\epsilon_i\epsilon_i]^{-1}E[\epsilon_i\mathbf{x}^T]$$

(d) The above recursive construction represents a successive improvement of the estimate of $\mathbf{x}$, as more and more y_ns are taken into account; that is, as the subspaces Y_n are successively enlarged. Verify that $\hat{\mathbf{x}}_n$ is indeed a better estimate than $\hat{\mathbf{x}}_{n-1}$ by showing that the mean-square estimation error $R_{e_n e_n}$ is smaller than the mean-square error $R_{e_{n-1}e_{n-1}}$. This is a very intuitive result; the more information we use the better the estimate.

Such recursive updating schemes are the essence of Kalman filtering. In that context, $\mathbf{G}_n$ is referred to as the "Kalman gain."

1.12 The recursive updating procedure given in Problem 1.11 is useful only if the gain coefficient $\mathbf{G}_n$ can be computed at each iteration n. For that, a knowledge of the relationship between $\mathbf{x}$ and y_n is required. Consider the case of Example 1.4.3 where $y_n = c_n x + v_n$; define the vectors

$$\mathbf{c}_n = [c_1, c_2, \dots, c_n]^T\,, \quad \mathbf{y}_n = [y_1, y_2, \dots, y_n]^T\,, \quad \text{for} \quad n = 1, 2, \dots, M$$

and let $\hat{x}_n$ and $e_n = x - \hat{x}_n$ be the estimate of x on the basis of Y_n and the corresponding estimation error.

(a) Using Eq. (1.4.9), show that

$$\hat{x}_n = \frac{1}{1+\mathbf{c}_n^T\mathbf{c}_n}\mathbf{c}_n^T\mathbf{y}_n \quad \text{and} \quad E[e_n^2]= E[xe_n]= \frac{1}{1+\mathbf{c}_n^T\mathbf{c}_n}$$

(b) Using Eq. (1.6.19), compute $\hat{y}_{n/n-1}$ and show that it may be expressed in the form

$$\hat{y}_{n/n-1} = c_n\hat{x}_{n-1} = \frac{c_n}{1+\mathbf{c}_{n-1}^T\mathbf{c}_{n-1}}\mathbf{c}_{n-1}^T\mathbf{y}_{n-1}$$

(c) Let $e_{n-1} = x - \hat{x}_{n-1}$ be the estimation error based on Y_{n-1}. Writing

$$\epsilon_n = y_n - \hat{y}_{n/n-1} = (c_n x + v_n) - c_n\hat{x}_{n-1} = c_n e_{n-1} + v_n$$

show that

$$E[\epsilon_n\epsilon_n] = (1+\mathbf{c}_n^T\mathbf{c}_n)(1+\mathbf{c}_{n-1}^T\mathbf{c}_{n-1})^{-1}$$
$$E[x\epsilon_n] = c_n(1+\mathbf{c}_{n-1}^T\mathbf{c}_{n-1})^{-1}$$

(d) Show that the estimate $\hat{x}_n$ of x can be computed recursively by

$$\hat{x}_n = \hat{x}_{n-1} + G_n(y_n - \hat{y}_{n/n-1}), \quad \text{where} \quad G_n = c_n(1+\mathbf{c}_n^T\mathbf{c}_n)^{-1}$$

1.13 Rederive the recursive updating equation given in Problem 1.12(d), without any reference to innovations or projections, by simply manipulating Eq. (1.4.9) algebraically, and writing it in recursive form.

1.14 *Computer Experiment.* A three-component random vector $\mathbf{y}$ has autocorrelation matrix

$$R = E[\mathbf{y}\mathbf{y}^T] = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 6 & 14 \\ 3 & 14 & 42 \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

Carry out the Gram-Schmidt orthogonalization procedure to determine the innovations representation $\mathbf{y} = B\boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} = [\epsilon_1, \epsilon_2, \epsilon_3]^T$ is a vector of uncorrelated components. The vector $\mathbf{y}$ can be simulated by generating a zero-mean gaussian vector of uncorrelated components $\boldsymbol{\epsilon}$ of the appropriate variances and constructing $\mathbf{y} = B\boldsymbol{\epsilon}$. Generate $N = 50$ such vectors $\mathbf{y}_n$, $n = 1, 2, \dots, N$ and compute the corresponding sample covariance matrix $\hat{R}$ given by Eq. (1.6.21). Compare it with the theoretical R. Is $\hat{R}$ consistent with the standard deviation intervals (1.6.23)? Repeat for $N = 100$.

1.15 The Gram-Schmidt orthogonalization procedure for a subspace $Y = \{y_1, y_2, \dots, y_M\}$ is initialized at the leftmost random variable y_1 by $\epsilon_1 = y_1$ and progresses to the right by successively orthogonalizing y_2, y_3, and so on. It results in the lower triangular representation $\mathbf{y} = B\boldsymbol{\epsilon}$. The procedure can just as well be started at the rightmost variable y_M and proceed backwards as follows:

$$\eta_M = y_M$$
$$\eta_{M-1} = y_{M-1} - (\text{projection of } y_{M-1} \text{ on } \eta_M)$$
$$\eta_{M-2} = y_{M-2} - (\text{projection of } y_{M-2} \text{ on } \{\eta_M, \eta_{M-1}\})$$

and so on. Show that the resulting uncorrelated vector $\boldsymbol{\eta} = [\eta_1, \eta_2, \dots, \eta_M]^T$ is related to $\mathbf{y} = [y_1, y_2, \dots, y_M]^T$ by a linear transformation

$$\mathbf{y} = U\boldsymbol{\eta}$$

where U is a unit *upper-triangular* matrix. Show also that this corresponds to a UL (rather than LU) Cholesky factorization of the covariance matrix R_{yy}.

1.16 Since "orthogonal" means "uncorrelated," the Gram-Schmidt orthogonalization procedure can also be understood as a correlation canceling operation. Explain how Eq. (1.6.20) may be thought of as a special case of the correlation canceler defined by Eqs. (1.4.1) and (1.4.2). What are $\mathbf{x}, \mathbf{y}, \mathbf{e}$, and H, in this case? Draw the correlation canceler diagram of Fig. 1.4.1 as it applies here, showing explicitly the components of all the vectors.

1.17 Using Eq. (1.7.11), show that the vector of coefficients $[a_{n1}, a_{n2}, \dots, a_{nn}]^T$ can be expressed explicitly in terms of the $\mathbf{y}$-basis as follows:

$$\begin{bmatrix} a_{n1} \\ a_{n2} \\ \vdots \\ a_{nn} \end{bmatrix} = -E[\mathbf{y}_{n-1}\mathbf{y}_{n-1}^T]^{-1}E[y_n\mathbf{y}_{n-1}], \quad \text{where} \quad \mathbf{y}_{n-1} = \begin{bmatrix} y_{n-1} \\ y_{n-2} \\ \vdots \\ y_0 \end{bmatrix}$$

1.18 Show that the mean-square estimation error of y_n on the basis of Y_{n-1}—that is, $E[\epsilon_n^2]$, where $\epsilon_n = y_n - \hat{y}_{n/n-1}$—can be expressed as

$$E[\epsilon_n^2] = E[\epsilon_n y_n] = E[y_n^2] - E[y_n\mathbf{y}_{n-1}^T]E[\mathbf{y}_{n-1}\mathbf{y}_{n-1}^T]^{-1}E[y_n\mathbf{y}_{n-1}]$$

1.19 Let $\mathbf{a}_n = [1, a_{n1}, a_{n2}, \dots, a_{nn}]^T$ for $n = 1, 2, \dots, M$. Show that the results of the last two problems can be combined into one enlarged matrix equation

$$E[\mathbf{y}_n\mathbf{y}_n^T]\mathbf{a}_n = E[\epsilon_n^2]\mathbf{u}_n$$

where $\mathbf{u}_n$ is the unit-vector $\mathbf{u}_n = [1, 0, 0, \dots, 0]^T$ consisting of one followed by n zeros, and $\mathbf{y}_n = [y_n, y_{n-1}, \dots, y_1, y_0]^T = [y_n, \mathbf{y}_{n-1}^T]^T$.

1.20 The quantity $\hat{y}_{n/n-1}$ of Eq. (1.6.19) is the best estimate of y_n based on all the previous ys, namely, $Y_{n-1} = \{y_0, y_1, \dots, y_{n-1}\}$. This can be understood in three ways: First, in terms of the orthogonal projection theorem as we demonstrated in the text. Second, in terms of the correlation canceler interpretation as suggested in Problem 1.16. And third, it may be proved directly as follows. Let $\hat{y}_{n/n-1}$ be given as a linear combination of the previous ys as in Eq. (1.7.11); the coefficients $[a_{n1}, a_{n2}, \dots, a_{nn}]^T$ are to be chosen optimally to minimize the estimation error ϵ_n given by Eq. (1.7.10) in the mean-square sense. In terms of the notation of Problem 1.19, Eq. (1.7.10) and the mean-square error $E[\epsilon_n^2]$ can be written in the compact vectorial form

$$\epsilon_n = \mathbf{a}_n^T\mathbf{y}_n, \quad \mathcal{E}(\mathbf{a}_n) = E[\epsilon_n^2] = \mathbf{a}_n^T E[\mathbf{y}_n\mathbf{y}_n^T]\mathbf{a}_n$$

The quantity $\mathcal{E}(\mathbf{a}_n)$ is to be minimized with respect to $\mathbf{a}_n$. The minimization must be subject to the constraint that the first entry of the vector $\mathbf{a}_n$ be unity. This constraint can be expressed in vector form as

$$\mathbf{a}_n^T\mathbf{u}_n = 1$$

where $\mathbf{u}_n$ is the unit vector defined in Problem 1.19. Incorporate this constraint with a Lagrange multiplier λ and minimize the performance index

$$\mathcal{E}(\mathbf{a}_n) = \mathbf{a}_n^T E[\mathbf{y}_n\mathbf{y}_n^T]\mathbf{a}_n + \lambda(1 - \mathbf{a}_n^T\mathbf{u}_n)$$

with respect to $\mathbf{a}_n$, then fix λ by enforcing the constraint, and finally show that the resulting solution of the minimization problem is identical to that given in Problem 1.19.

1.21 Show that the normal equations (1.8.12) can also be obtained by minimizing the performance indices (1.8.10) with respect to $\mathbf{a}$ and $\mathbf{b}$, subject to the constraints that the first element of $\mathbf{a}$ and the last element of $\mathbf{b}$ be unity. (*Hint:* These constraints are expressible in the form $\mathbf{u}^T\mathbf{a} = 1$ and $\mathbf{v}^T\mathbf{b} = 1$.)

1.22 Using Eq. (1.8.16), show that E_b can be expressed as the ratio of the two determinants $E_b = \det R / \det \bar{R}$.

1.23 Show Eqs. (1.8.28) and (1.8.35).

1.24 A random signal $x(n)$ is defined as a linear function of time by

$$x(n) = an + b$$

where a and b are independent zero-mean gaussian random variables of variances σ_a^2 and σ_b^2, respectively.

(a) Compute $E\big[x(n)^2\big]$.

(b) Is $x(n)$ a stationary process? Is it ergodic? Explain.

(c) For each fixed n, compute the probability density $p\big(x(n)\big)$.

(d) For each fixed n and m $(n \neq m)$, compute the conditional probability density function $p\big(x(n)|x(m)\big)$ of $x(n)$ given $x(m)$. (*Hint:* $x(n) - x(m) = (n-m)b$.)

1.25 Compute the sample autocorrelation of the sequences

(a) $y_n = 1$, for $0 \le n \le 10$.

(b) $y_n = (-1)^n$, for $0 \le n \le 10$.

in two ways: First in the time domain, using Eq. (1.11.1), and then in the z-domain, using Eq. (1.11.3) and computing its inverse z-transform.

1.26 *FFT Computation of Autocorrelations.* In many applications, a fast computation of sample autocorrelations or cross-correlations is required, as in the matched filtering operations in radar data processors. A fast way to compute the sample autocorrelation $\hat{R}_{yy}(k)$ of a length-N data segment $\mathbf{y} = [y_0, y_1, \dots, y_{N-1}]^T$ is based on Eq. (1.11.5) which can be computed using FFTs. Performing an inverse FFT on Eq. (1.11.5), we find the computationally efficient formula

$$\hat{R}_{yy}(k) = \frac{1}{N}\text{IFFT}\big[\,|\text{FFT}(\mathbf{y})|^2\big] \tag{P.1}$$

To avoid wrap-around errors introduced by the IFFT, the length N' of the FFT must be selected to be greater than the length of the function $\hat{R}_{yy}(k)$. Since $\hat{R}_{yy}(k)$ is double-sided with an extent $-(N-1) \le k \le (N-1)$, it will have length equal to $2N-1$. Thus, we must select $N' \ge 2N-1$. To see the wrap-around effects, consider the length-4 signal $\mathbf{y} = [1, 2, 2, 1]^T$.

(a) Compute $\hat{R}_{yy}(k)$ using the time-domain definition.

(b) Compute $\hat{R}_{yy}(k)$ according to Eq. (P.1) using 4-point FFTs.

(c) Repeat using 8-point FFTs.

1.27 *Computer Experiment.*

(a) Generate 1000 samples $x(n)$, $n = 0, 1, \dots, 999$, of a zero-mean, unit-variance, white gaussian noise sequence.

(b) Compute and plot the first 100 lags of its sample autocorrelation, that is, $\hat{R}_{yy}(k)$, for $k = 0, 1, \dots, 99$. Does $\hat{R}_{yy}(k)$ look like a delta function $\delta(k)$?

(c) Generate 10 different realizations of the length-1000 sequence $x(n)$, and compute 100 lags of the corresponding sample autocorrelations. Define an average autocorrelation by

$$\hat{R}(k) = \frac{1}{10}\sum_{i=1}^{10}\hat{R}_i(k)\,, \quad k = 0, 1, \dots, 99,$$

where $\hat{R}_i(k)$ is the sample autocorrelation of the ith realization of $x(n)$. Plot $\hat{R}(k)$ versus k. Do you notice any improvement?

1.28 A 500-millisecond record of a stationary random signal is sampled at a rate of 2 kHz and the resulting N samples are recorded for further processing. What is N? The record of N samples is then divided into K contiguous segments, each of length M, so that $M = N/K$. The periodograms from each segment are computed and averaged together to obtain an estimate of the power spectrum of the signal. A frequency resolution of $\Delta f = 20$ Hz is required. What is the shortest length M that will guarantee such resolution? (Larger Ms will have better resolution than required but will result in a poorer power spectrum estimate because K will be smaller.) What is K in this case?

1.29 A random signal y_n is generated by sending unit-variance zero-mean white noise ϵ_n through the filters defined by the following difference equations:

1. $y_n = -0.9y_{n-1} + \epsilon_n$
2. $y_n = 0.9y_{n-1} + \epsilon_n + \epsilon_{n-1}$
3. $y_n = \epsilon_n + 2\epsilon_{n-1} + \epsilon_{n-2}$
4. $y_n = -0.81y_{n-2} + \epsilon_n$
5. $y_n = 0.1y_{n-1} + 0.72y_{n-2} + \epsilon_n - 2\epsilon_{n-1} + \epsilon_{n-2}$

(a) For each case, determine the transfer function $B(z)$ of the filter and draw its canonical implementation form, identify the set of model parameters, and decide whether the model is ARMA, MA, or AR.

(b) Write explicitly the power spectrum $S_{yy}(\omega)$ using Eq. (1.13.6).

(c) Based on the pole/zero pattern of the filter $B(z)$, draw a rough sketch of the power spectrum $S_{yy}(\omega)$ for each case.

1.30 *Computer Experiment.*

Two different realizations of a stationary random signal $y(n)$, $n = 0, 1, \dots, 19$ are given. It is known that this signal has been generated by a model of the form

$$y(n) = ay(n-1) + \epsilon(n)$$

where $\epsilon(n)$ is gaussian zero-mean white noise of variance σ_ϵ^2.

(a) Estimate the model parameters a and σ_ϵ^2 using the maximum likelihood criterion for both realizations. (The exact values were $a = 0.95$ and $\sigma_\epsilon^2 = 1$.)

(b) Repeat using the Yule-Walker method.

This type of problem might, for example, arise in speech processing where $y(n)$ might represent a short segment of sampled unvoiced speech from which the filter parameters (model parameters) are to be extracted and stored for future regeneration of that segment. A realistic speech model would of course require a higher-order filter, typically, of order 10 to 15.

n	$y(n)$	$y(n)$
0	3.848	5.431
1	3.025	5.550
2	5.055	4.873
3	4.976	5.122
4	6.599	5.722
5	6.217	5.860
6	6.572	6.133
7	6.388	5.628
8	6.500	6.479
9	5.564	4.321
10	5.683	5.181
11	5.255	4.279
12	4.523	5.469
13	3.952	5.087
14	3.668	3.819
15	3.668	2.968
16	3.602	2.751
17	1.945	3.306
18	2.420	3.103
19	2.104	3.694

1.31 *Computer Experiment.*

(a) Using the Yule-Walker estimates $\{\hat{a}, \hat{\sigma}_\epsilon^2\}$ of the model parameters extracted from the first realization of $y(n)$ given in Problem 1.30, make a plot of the estimate of the power spectrum following Eq. (1.13.6), that is,

$$\hat{S}_{yy}(\omega) = \frac{\hat{\sigma}_\epsilon^2}{|1 - \hat{a}e^{-j\omega}|^2}$$

versus frequency ω in the interval $0 \le \omega \le \pi$.

(b) Also, plot the true power spectrum

$$S_{yy}(\omega) = \frac{\sigma_\epsilon^2}{|1 - ae^{-j\omega}|^2}$$

defined by the true model parameters $\{a, \sigma_\epsilon^2\} = \{0.95, 1\}$.

(c) Using the given data values $y(n)$ for the first realization, compute and plot the corresponding periodogram spectrum of Eq. (1.11.5). Preferably, plot all three spectra on the same graph. Compute the spectra at 100 or 200 equally spaced frequency points in the interval $[0, \pi]$. Plot all spectra in decibels.

(d) Repeat parts (a) through (c) using the second realization of $y(n)$.

Better agreement between estimated and true spectra can be obtained using Burg's analysis procedure instead of the Yule-Walker method. Burg's method performs remarkably well on the basis of very short data records. The Yule-Walker method also performs well but it requires somewhat longer records. These methods will be compared in Chap. 14.

1.32 In addition to the asymptotic results (1.16.4) for the model parameters, we will show in Chap. 14 that the estimates of filter parameter and the input variance are asymptotically uncorrelated, $E[\Delta a\, \Delta\sigma_\epsilon^2] = 0$. Using this result and Eq. (1.16.4), show that the variance of the spectrum estimate is given asymptotically by

$$E[\Delta S(\omega)\, \Delta S(\omega)] = \frac{2S(\omega)^2}{N}\left[1 + \frac{2(1-a^2)(\cos\omega - a)^2}{(1 - 2a\cos\omega + a^2)^2}\right]$$

where $\Delta S(\omega) = \hat{S}(\omega) - S(\omega)$, with the theoretical and estimated spectra given in terms of the theoretical and estimated model parameters by

$$S(\omega) = \frac{\sigma_\epsilon^2}{|1 - ae^{-j\omega}|^2}, \quad \hat{S}(\omega) = \frac{\hat{\sigma}_\epsilon^2}{|1 - \hat{a}e^{-j\omega}|^2}$$

1.33 For any positive semi-definite matrix B show the inequality $\mathrm{tr}(B - I - \ln B) \ge 0$ with equality achieved for $B = I$. Using this property, show the inequality $f(R) \ge f(\hat{R})$, where $f(R) = \mathrm{tr}(\ln R + R^{-1}\hat{R})$. This implies the maximum likelihood property of $\hat{R}$, discussed in Sec. 1.18.

1.34 Show the following three matrix properties used in Sec. 1.18:

$$\ln(\det R) = \mathrm{tr}(\ln R), \quad d\,\mathrm{tr}(\ln R) = \mathrm{tr}(R^{-1}dR), \quad dR^{-1} = -R^{-1}\,dR\,R^{-1}$$

(*Hints:* for the first two, use the eigenvalue decomposition of R; for the third, start with $R^{-1}R = I$.)

1.35 Let $x(n)$ be a zero-mean white-noise sequence of unit variance. For each of the following filters compute the output autocorrelation $R_{yy}(k)$ for all k, using z-transforms:

1. $y(n) = x(n) - x(n-1)$
2. $y(n) = x(n) - 2x(n-1) + x(n-2)$
3. $y(n) = -0.5y(n-1) + x(n)$
4. $y(n) = 0.25y(n-2) + x(n)$

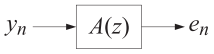
Also, sketch the output power spectrum $S_{yy}(\omega)$ versus frequency ω.

1.36 Let y_n be the output of a (stable and causal) filter $H(z)$ driven by the signal x_n, and let w_n be another unrelated signal. Assume all signals are stationary random signals. Show the following relationships between power spectral densities:

(a) $S_{yw}(z) = H(z)S_{xw}(z)$

(b) $S_{wy}(z) = S_{wx}(z)H(z^{-1})$

1.37 A stationary random signal y_n is sent through a finite filter $A(z) = a_0 + a_1 z^{-1} + \cdots + a_M z^{-M}$ to obtain the output signal e_n:



$$e_n = \sum_{m=0}^{M} a_m y_{n-m}$$

Show that the average power of the output e_n can be expressed in the two alternative forms:

$$E[e_n^2] = \int_{-\pi}^{\pi} S_{yy}(\omega)\,|A(\omega)|^2 \frac{d\omega}{2\pi} = \mathbf{a}^T R_{yy}\mathbf{a}$$

where $\mathbf{a} = [a_0, a_1, \dots, a_M]^T$ and R_{yy} is the $(M+1)\times(M+1)$ autocorrelation matrix of y_n having matrix elements $R_{yy}(i,j) = E[y_i y_j] = R_{yy}(i-j)$.

1.38 Consider the two autoregressive random signals y_n and y'_n generated by the two signal models:

$$A(z) = 1 + a_1 z^{-1} + \cdots + a_M z^{-M} \quad \text{and} \quad A'(z) = 1 + a'_1 z^{-1} + \cdots + a'_M z^{-M}$$

$$\epsilon_n \longrightarrow \boxed{1/A(z)} \longrightarrow y_n \qquad\qquad \epsilon'_n \longrightarrow \boxed{1/A'(z)} \longrightarrow y'_n$$

(a) Suppose y_n is filtered through the analysis filter $A'(z)$ of y'_n producing the output signal e_n; that is,

$$y_n \longrightarrow \boxed{A'(z)} \longrightarrow e_n \qquad e_n = \sum_{m=0}^{M} a'_m y_{n-m}$$

If y_n were to be filtered through its own analysis filter $A(z)$, it would produce the innovations sequence ϵ_n. Show that the average power of e_n compared to the average power of ϵ_n is given by

$$\frac{\sigma_e^2}{\sigma_\epsilon^2} = \frac{\mathbf{a}'^T R_{yy}\mathbf{a}'}{\mathbf{a}^T R_{yy}\mathbf{a}} = \int_{-\pi}^{\pi} \left| \frac{A'(\omega)}{A(\omega)} \right|^2 \frac{d\omega}{2\pi} = \left\| \frac{A'}{A} \right\|^2$$

where $\mathbf{a}, \mathbf{a}'$ and R_{yy} have the same meaning as in Problem 1.37. This ratio can be taken as a measure of *similarity* between the two signal models. The log of this ratio is Itakura's *LPC distance measure* used in speech recognition.

(b) Alternatively, show that if y'_n were to be filtered through y_n's analysis filter $A(z)$ resulting in $e'_n = \sum_{m=0}^{M} a_m y'_{n-m}$, then

$$\frac{\sigma_{e'}^2}{\sigma_{\epsilon'}^2} = \frac{\mathbf{a}^T R'_{yy}\mathbf{a}}{\mathbf{a}'^T R'_{yy}\mathbf{a}'} = \int_{-\pi}^{\pi} \left| \frac{A(\omega)}{A'(\omega)} \right|^2 \frac{d\omega}{2\pi} = \left\| \frac{A}{A'} \right\|^2$$

1.39 The autocorrelation function of a complex-valued signal is defined by

$$R_{yy}(k) = E[y_{n+k} y_n^*]$$

(a) Show that stationarity implies $R_{yy}(-k) = R_{yy}(k)^*$.

(b) If y_n is filtered through a (possibly complex-valued) filter $A(z) = a_0 + a_1 z^{-1} + \cdots + a_M z^{-M}$, show that the average power of the output signal e_n can be expressed as

$$E[e_n^* e_n] = \mathbf{a}^\dagger R_{yy}\mathbf{a}$$

where $\mathbf{a}^\dagger$ denotes the hermitian conjugate of $\mathbf{a}$ and R_{yy} has matrix elements

$$R_{yy}(i,j) = R_{yy}(i-j)$$

1.40 (a) Let $y_n = A_1 \exp\big[j(\omega_1 n + \phi_1)\big]$ be a complex sinusoid of amplitude A_1 and frequency ω_1. The randomness of y_n arises only from the phase ϕ_1 which is assumed to be a random variable uniformly distributed over the interval $0 \le \phi_1 \le 2\pi$. Show that the autocorrelation function of y_n is

$$R_{yy}(k) = |A_1|^2 \exp(j\omega_1 k)$$

(b) Let y_n be the sum of two sinusoids

$$y_n = A_1 \exp\big[j(\omega_1 n + \phi_1)\big] + A_2 \exp\big[j(\omega_2 n + \phi_2)\big]$$

with uniformly distributed random phases ϕ_1 and ϕ_2 which are also assumed to be independent of each other. Show that the autocorrelation function of y_n is

$$R_{yy}(k) = |A_1|^2 \exp(j\omega_1 k) + |A_2|^2 \exp(j\omega_2 k)$$

1.41 *Sinusoids in Noise.* Suppose y_n is the sum of L complex sinusoids with random phases, in the presence of uncorrelated noise:

$$y_n = v_n + \sum_{i=1}^{L} A_i \exp\big[j(\omega_i n + \phi_i)\big]$$

where ϕ_i, $i = 1, 2, \dots, L$ are uniformly distributed random phases which are assumed to be mutually independent, and v_n is zero-mean white noise of variance σ_v^2. Also, assume that v_n is independent of ϕ_i.

(a) Show that $E[e^{j\phi_i} e^{-j\phi_k}] = \delta_{ik}$, for $i, k = 1, 2, \dots, L$.

(b) Show that the autocorrelation of y_n is

$$R_{yy}(k) = \sigma_v^2 \delta(k) + \sum_{i=1}^{L} |A_i|^2 \exp(j\omega_i k)$$

(c) Suppose y_n is filtered through a filter $A(z) = a_0 + a_1 z^{-1} + \cdots + a_M z^{-M}$ of order M, producing the output signal e_n. Show that the average output power is expressible as

$$\mathcal{E} = E[e_n^* e_n] = \mathbf{a}^\dagger R_{yy}\mathbf{a} = \sigma_v^2\, \mathbf{a}^\dagger \mathbf{a} + \sum_{i=1}^{L} |A_i|^2 \,|A(\omega_i)|^2$$

where $\mathbf{a}, \mathbf{a}^\dagger, R_{yy}$ have the same meaning as in Problem 1.39, and $A(\omega_i)$ is the frequency response of the filter evaluated at the sinusoid frequency ω_i, that is,

$$A(\omega_i) = \sum_{m=0}^{M} a_m e^{-j\omega_i m}, \quad i = 1, 2, \dots, M$$

(d) If the noise v_n is correlated with autocorrelation $Q(k)$, so that $E[v_{n+k} v_n^*] = Q(k)$, show that in this case

$$\mathcal{E} = E[e_n^* e_n] = \mathbf{a}^\dagger R_{yy}\mathbf{a} = \mathbf{a}^\dagger Q \mathbf{a} + \sum_{i=1}^{L} |A_i|^2 \,|A(\omega_i)|^2$$

where Q is the noise covariance matrix, $Q(i,j) = Q(i-j)$.

1.42 A filter is defined by $y(n) = -0.64y(n-2) + 0.36x(n)$.

(a) Suppose the input is zero-mean, unit-variance, white noise. Compute the output spectral density $S_{yy}(z)$ and power spectrum $S_{yy}(\omega)$ and plot it roughly versus frequency.

(b) Compute the output autocorrelation $R_{yy}(k)$ for all lags k.

(c) Compute the noise reduction ratio of this filter.

(d) What signal $s(n)$ can pass through this filter and remain entirely unaffected (at least in the steady-state regime)?

(e) How can the filter coefficients be changed so that (i) the noise reduction capability of the filter is improved, while at the same time (ii) the above signal $s(n)$ still goes through unchanged? Explain any tradeoffs.

1.43 *Computer Experiment.* (a) Generate 1000 samples of a zero-mean, unit-variance, white gaussian noise sequence $x(n)$, $n = 0, 1, \dots, 999$, and filter them through the filter defined by the difference equation:

$$y(n) = ay(n-1) + (1-a)x(n)$$

with $a = 0.95$. To avoid the transient effects introduced by the filter, discard the first 900 output samples and save the last 100 samples of $y(n)$. Compute the sample autocorrelation of $y(n)$ from this length-100 block of samples.

(b) Determine the theoretical autocorrelation $R_{yy}(k)$, and on the same graph, plot the theoretical and sample autocorrelations versus k. Do they agree?

1.44 Prove Eq. (1.19.6).

1.45 Using Eq. (1.19.10), show Eqs. (1.19.12) and (1.19.13).

1.46 A random signal y_n has autocorrelation function

$$R_{yy}(k) = (0.5)^{|k|}\,, \quad \text{for all } k$$

Find a random signal generator model for y_n.

1.47 Repeat Problem 1.46 when

$$R_{yy}(k) = (0.5)^{|k|} + (-0.5)^{|k|}\,, \quad \text{for all } k$$

1.48 The autocorrelation function of a stationary random signal $y(n)$ is

$$R_{yy}(k) = \frac{1-R^2}{1+R^2}R^{|k|}\cos(\pi k/2)\,, \quad \text{for all } k\,, \quad \text{where } 0 < R < 1$$

(a) Compute the power spectrum $S_{yy}(\omega)$ of $y(n)$ and sketch it versus frequency for various values of R.

(b) Find the signal generator filter for $y(n)$ and determine its difference equation and its poles and zeros.

1.49 A stationary random signal y_n has a rational power spectral density given by

$$S_{yy}(z) = \frac{2.18 - 0.6(z + z^{-1})}{1.25 - 0.5(z + z^{-1})}$$

Determine the signal model filter $B(z)$ and the parameter σ_ϵ^2. Write the difference equation generating y_n.

1.50 Let $y_n = cx_n + v_n$. It is given that

$$S_{xx}(z) = \frac{Q}{(1 - az^{-1})(1 - az)}\,, \quad S_{vv}(z) = R\,, \quad S_{xv}(z) = 0$$

where a, c, Q, R are known constants (assume $|a| < 1$) for the stability of x_n.)

(a) Show that the filter model for y_n is of the form

$$B(z) = \frac{1 - fz^{-1}}{1 - az^{-1}}$$

where f has magnitude less than one and is the solution of the algebraic quadratic equation

$$aR(1 + f^2) = \big[c^2Q + R(1 + a^2)\big]f$$

and show that the other solution has magnitude greater than one.

(b) Show that f can alternatively be expressed as

$$f = \frac{Ra}{R + c^2P}$$

where P is the *positive* solution of the quadratic equation

$$Q = P - \frac{PRa^2}{R + c^2P}$$

known as the *algebraic Riccati* equation. Show that the other solution is negative. Show that the positivity of P is essential to guarantee that f has magnitude less than one.

(c) Show that the scale factor σ_ϵ^2 that appears in the spectral factorization (1.20.1) can also be expressed in terms of P as

$$\sigma_\epsilon^2 = R + c^2P$$

The above method of solution of the spectral factorization problem by reducing it to the solution of an algebraic Riccati equation is quite general and can be extended to the multichannel case.

1.51 Consider a stable (but not necessarily causal) sequence b_n, $-\infty < n < \infty$ with a z-transform $B(z)$

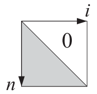
$$B(z) = \sum_{n=-\infty}^{\infty} b_n z^{-n}$$

Define an infinite Toeplitz matrix B by

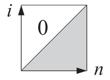
$$B_{ni} = b_{n-i}\,, \quad \text{for } -\infty < n, i < \infty$$

This establishes a correspondence between stable z-transforms or stable sequences and infinite Toeplitz matrices.

(a) Show that if the sequence b_n is causal, then B is lower triangular, as shown here



In the literature of integral operators and kernels, such matrices are rotated by 90° degrees as shown:



so that the n axis is the horizontal axis. For this reason, in that context they are called “right Volterra kernel,” or “causal kernels.”

(b) Show that the transposed B^T corresponds to the reflected (about the origin) sequence b_{-n} and to the z-transform $B(z^{-1})$.

(c) Show that the convolution of two sequences a_n and b_n

$$c_n = a_n * b_n \quad \text{or} \quad C(z) = A(z)B(z)$$

corresponds to the commutative matrix product

$$C = AB = BA$$

1.52 Prove Eq. (1.21.2) for any M.

2

Signal Extraction Basics

2.1 Introduction

One of the most common tasks in signal processing is to extract a desired signal, say x_n, from an observed signal:

$$y_n = x_n + v_n \tag{2.1.1}$$

where v_n is an undesired component. The nature of v_n depends on the application. For example, it could be a white noise signal, which is typical of the background noise picked up during the measurement process, or it could be any other signal—not necessarily measurement noise—that must be separated from x_n.

The desired signal x_n often represents a smooth *trend* that conveys useful information about the underlying dynamics of the evolving time series. Trend extraction is carried out routinely on financial, business, census, climatic, and other applications.

An estimate, $\hat{x}_n$, of the desired signal x_n is obtained by processing the observed signal y_n through a processor designed according to some optimization criterion. There exist a large variety of signal extraction methods, most of them based on a least-squares minimization criterion, falling into two basic classes: (a) model-based parametric methods, such as those based on Wiener and Kalman filtering, and (b) non-parametric methods based on a variety of approaches, such as local polynomial modeling, exponential smoothing, splines, regularization filters, wavelets, and SVD-based methods. Some of the non-parametric methods (exponential smoothing, splines, regularization filters) can also be cast in a state-space Kalman filtering form.

We discuss the Wiener and Kalman approaches in chapters 11 and 13, and the SVD-based methods in chapter 15. In this chapter, we concentrate primarily on non-parametric methods.

We consider also the problem of “de-seasonalizing” a time series, that is, estimating and removing a periodic component. Many physical and financial time series have a natural periodicity built into them, such as daily, monthly, quarterly, yearly. The observed signal can be decomposed into three components: a periodic (or nearly periodic) seasonal part s_n, a smooth trend t_n, and a residual irregular part v_n that typically represents noise,

$$y_n = s_n + t_n + v_n \tag{2.1.2}$$

In such cases, the signal processing task is to determine both the trend and the seasonal components, t_n and s_n. Often, economic data are available only after they have been de-seasonalized, that is, after the seasonal part s_n has been removed. Further processing of the de-seasonalized trend, t_n, can provide additional information such as identifying business cycles. Moreover, modeling of the trend can be used for forecasting purposes.

The particular methods of smoothing, trend extraction, and seasonal decomposition that we consider in this and the next few chapters are:

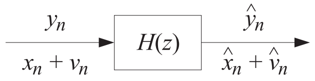
- local polynomial smoothing filters (Savitzky-Golay filters) — Chap. 3
- minimum-roughness filters (Henderson filters) — Chap. 4
- local polynomial modeling and LOESS — Chap. 5
- exponential smoothing — Chap. 6
- smoothing splines — Chap. 7
- regularization filters (Whittaker-Henderson, Hodrick-Prescott) — Chap. 8
- wavelet denoising — Chap. 10
- seasonal decomposition (classical, moving average, census X-11) — Chap. 9
- bandpass and other filters in business and finance — Chap. 8

2.2 Noise Reduction and Signal Enhancement

A standard method of extracting the desired signal x_n from y_n is to design an appropriate filter $H(z)$ that removes the noise component v_n and at the same time lets x_n go through unchanged. It is useful to view the design specifications and operation of such filter both in the time and frequency domains. Using linearity, we can express the output signal due to the input of Eq. (2.1.1) in the form:

$$\hat{y}_n = \hat{x}_n + \hat{v}_n \tag{2.2.1}$$

where $\hat{x}_n$ is the output due to x_n and $\hat{v}_n$ the output due to v_n. The two design conditions for the filter are that $\hat{x}_n$ be as similar to x_n as possible and that $\hat{v}_n$ be as small as possible; that is, ideally we require:[†]



$$\boxed{\begin{aligned}\hat{x}_n &= x_n\\ \hat{v}_n &= 0\end{aligned}} \tag{2.2.2}$$

In general, these conditions cannot be satisfied simultaneously. To determine when they can be satisfied, we may express them in the frequency domain in terms of the corresponding frequency spectra as follows: $\hat{X}(\omega)= X(\omega)$ and $\hat{V}(\omega)= 0$.

Applying the filtering equation $\hat{Y}(\omega)= H(\omega)Y(\omega)$ separately to the signal and noise components, we have the conditions:

$$\begin{aligned}\hat{X}(\omega) &= H(\omega)X(\omega)= X(\omega)\\ \hat{V}(\omega) &= H(\omega)V(\omega)= 0\end{aligned} \tag{2.2.3}$$

[†] An overall delay in the recovered signal is often acceptable, that is, $\hat{x}_n = x_{n-D}$.

The first requires that $H(\omega)=1$ at all ω at which the signal spectrum is nonzero, $X(\omega)\neq 0$. The second requires that $H(\omega)=0$ at all ω for which the noise spectrum is nonzero, $V(\omega)\neq 0$.

These two conditions can be met simultaneously only if the signal and noise spectra do not overlap, as shown in Fig. 2.2.1.[‡] In such cases, the filter $H(\omega)$ must have a *passband* that coincides with the signal band, and a *stopband* that coincides with the noise band. The filter removes the noise spectrum and leaves the signal spectrum unchanged.

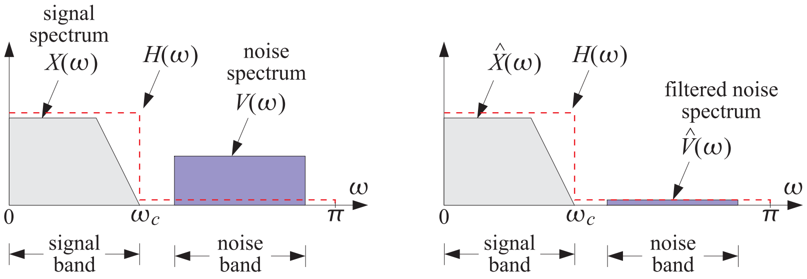


Fig. 2.2.1 Signal and noise spectra before and after filtering.

If the signal and noise spectra overlap, as is the typical case in practice, the above conditions cannot be satisfied simultaneously. In such cases, we must compromise between the two design conditions and trade off one for the other. Depending on the application, we may decide to design the filter to remove as much noise as possible, but at the expense of distorting the desired signal. Alternatively, we may decide to leave the desired signal as undistorted as possible, but at the expense of having some noise in the output.

The latter alternative is depicted in Fig. 2.2.2 where a low-frequency signal x_n exists in the presence of a broadband noise component, such as white noise, having a flat spectrum extending over the entire[1] Nyquist interval, $-\pi\leq\omega\leq\pi$.

The filter $H(\omega)$ is chosen to be an ideal lowpass filter with passband covering the signal bandwidth, say $0\leq\omega\leq\omega_c$. The noise energy in the filter's stopband $\omega_c\leq\omega\leq\pi$ is removed completely by the filter, thus reducing the strength (i.e., the rms value) of the noise. The spectrum of the desired signal is not affected by the filter, but neither is the portion of the noise spectrum that falls within the signal band. Thus, some noise will survive the filtering process.

A measure of the amount of noise reduction achieved by a filter is given by the *noise gain*, or *noise reduction ratio* (NRR) of the filter, defined in Eq. (1.12.16), which is valid for white noise input signals. Denoting the input and output mean-square noise values by $\sigma^2=E[v_n^2]$ and $\hat{\sigma}^2=E[\hat{v}_n^2]$, we have:

$$\boxed{\mathcal{R}=\frac{\hat{\sigma}^2}{\sigma^2}=\frac{1}{2\pi}\int_{-\pi}^{\pi}|H(\omega)|^2\,d\omega=\sum_n h_n^2}\tag{2.2.4}$$

[‡] Here, ω is in units of radians per sample, i.e., $\omega=2\pi f/f_s$, with f in Hz, and f_s is the sampling rate.

[1] For discrete-time signals, the spectra are periodic in ω with period 2π, or in f with period f_s.

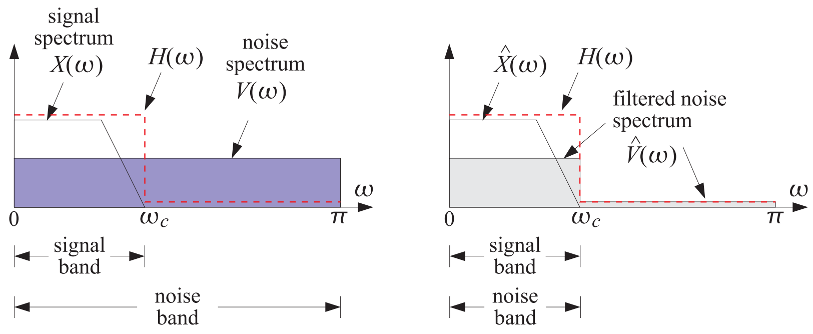


Fig. 2.2.2 Signal enhancement filter with partial noise reduction.

For the case of an ideal lowpass filter, with frequency and impulse responses given by [29],

$$H(\omega)=\begin{cases}1, & \text{if } |\omega|\leq\omega_c\\ 0, & \text{if } \omega_c\leq|\omega|\leq\pi\end{cases}\qquad\text{and}\qquad h_n=\frac{\sin(\omega_c n)}{\pi n}\,,\quad -\infty<n<\infty\tag{2.2.5}$$

the integration range in Eq. (2.2.4) collapses to the filter's passband, that is, $-\omega_c\leq\omega\leq\omega_c$, and over this range the value of $H(\omega)$ is unity, giving:

$$\mathcal{R}=\frac{\hat{\sigma}^2}{\sigma^2}=\frac{1}{2\pi}\int_{-\omega_c}^{\omega_c}1\cdot d\omega=\frac{2\omega_c}{2\pi}=\frac{\omega_c}{\pi}\tag{2.2.6}$$

Thus, the NRR is the proportion of the signal bandwidth with respect to the Nyquist interval. The same conclusion also holds when the desired signal is a high-frequency or a mid-frequency signal. For example, if the signal spectrum extends only over the mid-frequency band $\omega_a\leq|\omega|\leq\omega_b$, then $H(\omega)$ can be designed to be unity over this band and zero otherwise. A similar calculation yields in this case:

$$\mathcal{R}=\frac{\hat{\sigma}^2}{\sigma^2}=\frac{\omega_b-\omega_a}{\pi}\tag{2.2.7}$$

The noise reduction/signal enhancement capability of a filter can also be expressed in terms of the signal-to-noise ratio. The SNRs at the input and output of the filter are defined in terms of the mean-square values as:

$$SNR_{\text{in}}=\frac{E[x_n^2]}{E[v_n^2]}\,,\qquad SNR_{\text{out}}=\frac{E[\hat{x}_n^2]}{E[\hat{v}_n^2]}$$

Therefore, the relative *improvement* in the SNR introduced by the filter will be:

$$\frac{SNR_{\text{out}}}{SNR_{\text{in}}}=\frac{E[\hat{x}_n^2]}{E[\hat{v}_n^2]}\cdot\frac{E[v_n^2]}{E[x_n^2]}=\frac{1}{\mathcal{R}}\cdot\frac{E[\hat{x}_n^2]}{E[x_n^2]}$$

If the desired signal is not changed by the filter, $\hat{x}_n=x_n$, then

$$\boxed{\frac{SNR_{\text{out}}}{SNR_{\text{in}}}=\frac{1}{\mathcal{R}}}\tag{2.2.8}$$

Thus, minimizing the noise reduction ratio is equivalent to maximizing the signal-to-noise ratio at the filter's output.

The NRRs computed in Eqs. (2.2.6) or (2.2.7) give the *maximum* noise reductions achievable with *ideal* lowpass or bandpass filters that do not distort the desired signal. Such ideal filters are not realizable because they have double-sided impulse responses with infinite anticausal tails. Thus, in practice, we must use *realizable* approximations to the ideal filters, such as FIR filters, or causal IIR filters. The realizable filters may meet the two design goals approximately, for example, by minimizing the NRR subject to certain constraints that help sustain the signal passband. Examples of this approach are discussed in Sections 2.3, 2.4, and generalized in Sections 3.1 and 4.2.

The use of realizable filters introduces two further design issues that must be dealt with in practice: One is the *transient response* of the filter and the other, the amount of *delay* introduced into the output. The more closely a filter approximates the sharp transition characteristics of an ideal response, the closer to the unit circle its poles get, and the longer its transient response becomes. Stated differently, maximum noise reduction, approaching the ideal limit (2.2.6), can be achieved only at the expense of introducing long transients in the output.

The issue of the delay introduced into the output has to do with the steady-state response of the filter. After steady-state has set in, different frequency components of an input signal suffer different amounts of delay, as determined by the phase delay $d(\omega)= -\text{Arg}\,H(\omega)/\omega$ of the filter [29].

In particular, if the filter has *linear phase*, then it causes an overall delay in the output. Indeed, assuming that the filter has nearly unity magnitude, $|H(\omega)|\simeq 1$, over its passband (i.e., the signal band) and is zero over the stopband, and assuming a constant phase delay $d(\omega)= D$, we have for the frequency response

$$H(\omega)= |H(\omega)|e^{-j\omega d(\omega)} \simeq e^{-j\omega D}$$

over the passband, and we find for the filtered version of the desired signal:

$$\begin{aligned}\hat{x}_n &= \frac{1}{2\pi}\int_{-\pi}^{\pi}\hat{X}(\omega)e^{j\omega n}\,d\omega = \frac{1}{2\pi}\int_{-\pi}^{\pi}H(\omega)X(\omega)e^{j\omega n}\,d\omega\\ &= \frac{1}{2\pi}\int_{-\omega_c}^{\omega_c}X(\omega)e^{j\omega(n-D)}\,d\omega = x(n-D)\end{aligned}$$

the last equation following from the inverse DTFT of the desired signal:

$$x_n = \frac{1}{2\pi}\int_{-\omega_c}^{\omega_c}X(\omega)e^{j\omega n}\,d\omega$$

Many smoothing filters used in practice (e.g., see Chapters 3 and 4) are double-sided filters, h_n, $-M\le n\le M$, with a symmetric impulse response, $h_n = h_{-n}$, and therefore, they introduce no delay in the output ($D = 0$). On the other hand, if such filters are made causal by a delay ($D = M$), then they will introduce a delay in the output. Such delays are of concern in some applications such as monitoring and filtering real-time data in the financial markets.

Next, we consider some noise reduction examples based on simple filters, calculate the corresponding noise reduction ratios, discuss the tradeoff between transient response and noise reduction, and present some simulation examples.

2.3 *First-Order Exponential Smoother*

It is desired to extract a constant signal $x_n = s$ from the noisy measured signal

$$y_n = x_n + v_n = s + v_n$$

where v_n is zero-mean white Gaussian noise of variance σ_v^2. To this end, the following IIR lowpass filter may be used, where $b = 1-a$,

$$H(z)= \frac{b}{1-az^{-1}}\,,\quad H(\omega)= \frac{b}{1-ae^{-j\omega}}\,,\quad |H(\omega)|^2 = \frac{b^2}{1-2a\cos\omega + a^2} \qquad (2.3.1)$$

where the parameter a is restricted to the range $0<a<1$. Because the desired signal x_n is constant in time, the signal band will be just the DC frequency $\omega = 0$. We require therefore that the filter have unity gain at DC. This is guaranteed by the above choice of the parameter b, that is, we have at $\omega = 0$, or equivalently at $z = 1$,

$$H(z)\big|_{z=1} = \frac{b}{1-a} = 1$$

The NRR can be calculated from Eq. (2.2.4) by summing the impulse response squared. Here, $h_n = ba^n u_n$, therefore, using the geometric series, we find

$$\mathcal{R} = \frac{\hat{\sigma}^2}{\sigma^2} = \sum_n h_n^2 = b^2\sum_{n=0}^{\infty}a^{2n} = \frac{b^2}{1-a^2} = \frac{(1-a)^2}{1-a^2} = \frac{1-a}{1+a} \qquad (2.3.2)$$

The filter's magnitude response, pole-zero pattern, and the corresponding input and output noise spectra are shown in Fig. 2.3.1. The shaded area under the $|H(\omega)|^2$ curve (including its negative-frequency portion) is equal as the NRR computed above.

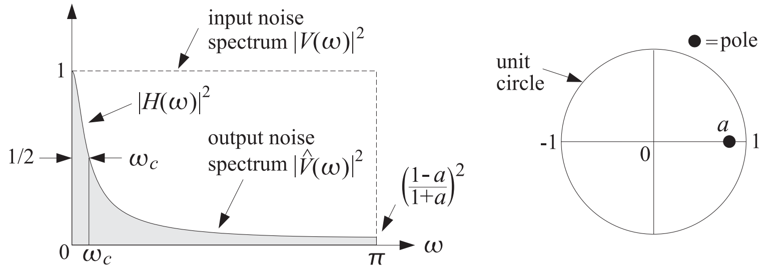


Fig. 2.3.1 Lowpass exponential smoothing filter.

The NRR is always less than unity because a is restricted to $0<a<1$. To achieve high noise reduction, a must be chosen near one. But, then the filter's effective time constant will become large:[†]

$$n_{\text{eff}} = \frac{\ln\epsilon}{\ln a} \to \infty \qquad \text{as} \qquad a\to 1$$

[†] The values $\epsilon = 0.01$ and $\epsilon = 0.001$ correspond to the so-called 40-dB and 60-dB time constants [30].

The filter's 3-dB cutoff frequency ω_c can be calculated by requiring that $|H(\omega_c)|^2$ drops by $1/2$, that is,

$$|H(\omega_c)|^2 = \frac{b^2}{1 - 2a\cos\omega_c + a^2} = \frac{1}{2}$$

which can be solved to give $\cos\omega_c = 1 - (1-a)^2/2a$. If a is near one, $a \lesssim 1$, we can use the approximation $\cos x \simeq 1 - x^2/2$ and solve for ω_c approximately:[†]

$$\omega_c \simeq 1 - a$$

This shows that as $a \to 1$, the filter becomes a narrower lowpass filter, removing more noise from the input, but at the expense of increasing the time constant.

The tradeoff between noise reduction and speed of response is illustrated in Fig. 2.3.2, where 200 samples of a simulated noisy signal y_n were filtered using the difference equation of the filter, that is, replacing $b = 1 - a$

$$y_n = s + v_n\,, \quad \hat{x}_n = a\hat{x}_{n-1} + (1-a)y_n \tag{2.3.3}$$

and initialized at $\hat{x}_{-1} = 0$. The value of the constant was $s = 5$, and the input noise variance, $\sigma_v^2 = 1$. The random signal v_n was generated by the built-in MATLAB function `randn`. The figure on the left corresponds to $a = 0.90$, which has a 40-dB time constant, NRR, and SNR improvement in dB:

$$n_{\text{eff}} = \frac{\ln(0.01)}{\ln(0.90)} = 44\,, \qquad \mathcal{R} = \frac{1-0.90}{1+0.90} = \frac{1}{19}\,, \qquad 10\log_{10}\left(\frac{1}{\mathcal{R}}\right) = 12.8\text{ dB}$$

The right figure has $a = 0.98$, with a longer time constant of $n_{\text{eff}} = 228$, a smaller $\mathcal{R} = 1/99$, and bigger SNR improvement, $10\log_{10}(1/\mathcal{R}) = 20$ dB.

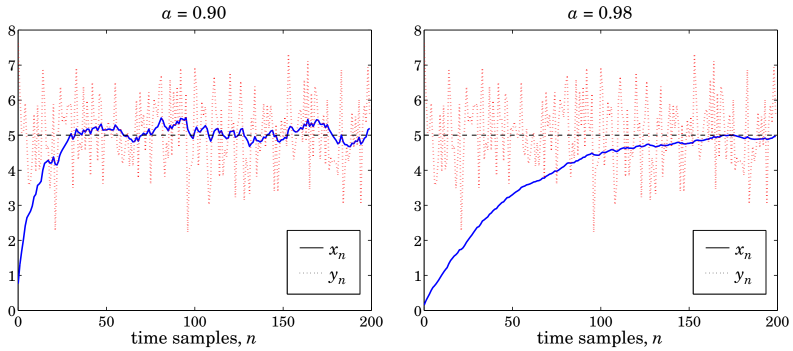


Fig. 2.3.2 Noisy input and smoothed output.

[†] The full 3-dB width of the interval $[-\omega_c, \omega_c]$ is $2\omega_c = 2(1-a)$. This is a special case of a more general result [30] that the 3-dB width due to a filter pole with radius r near the unit circle, $r \lesssim 1$, is given by $\Delta\omega = 2(1-r)$.

To understand how this filter works in the time domain and manages to reduce the noise, we rewrite the difference equation (2.3.3) in its convolutional form:

$$\hat{x}_n = b\sum_{m=0}^{n} a^m y_{n-m} = b\big(y_n + ay_{n-1} + a^2 y_{n-2} + \cdots + a^n y_0\big)$$

The sum represents a weighted average of all the past samples up to the present time instant. As a result, the rapid fluctuations of the noise component v_n are averaged out. The closer a is to 1, the more equal weighting the terms get, and the more effective the averaging of the noise. The exponential weighting de-emphasizes the older samples and causes the sum to behave as though it had effectively a finite number of terms, thus, safeguarding the mean-square value of $\hat{x}_n$ from diverging (see, for example, Sec. 1.15.) Because of the exponential weighting, this filter is also called an *exponential smoother*.

This filter can be applied to the smoothing of *any* low-frequency signal, not just constants. One must make sure that the bandwidth of the desired signal x_n is *narrower* than the filter's lowpass width ω_c, so that the filter will not remove any of the higher frequencies present in x_n.

The exponential smoother is a standard tool in many applications requiring the smoothing of data in signal processing, statistics, economics, physics, and chemistry. It is also widely used in forecasting applications, for example in inventory control, where the quantity $\hat{x}_n$ is interpreted as the one-step ahead forecast. More precisely, the *forecasting filter* and its I/O difference equation are given by:

$$H_f(z) = z^{-1}H(z) = \frac{bz^{-1}}{1 - az^{-1}}\,, \quad F_{n+1} = aF_n + (1-a)y_n \tag{2.3.4}$$

where F_{n+1} is the predicted value of x_{n+1} based on the available data y_n up to time n.

We discuss the exponential smoother further in Sec. 6.1, where we rederive it from an optimization criterion and generalize it to higher orders.

A slight variation of Eq. (2.3.1) which improves the NRR without affecting the speed of response can be derived by adding a zero in the transfer function at $z = -1$ or equivalently, at $\omega = \pi$. The resulting first-order filter will be:

$$H(z) = \frac{b(1+z^{-1})}{1 - az^{-1}} \quad \Rightarrow \quad |H(\omega)|^2 = \frac{2b^2(1+\cos\omega)}{1 - 2a\cos\omega + a^2} \tag{2.3.5}$$

where b is fixed by requiring unity gain at DC:

$$H(z)\big|_{z=1} = \frac{2b}{1-a} = 1 \quad \Rightarrow \quad b = \frac{1-a}{2}$$

The zero at $\omega = \pi$ suppresses the high-frequency portion of the input noise spectrum even more than the filter of Eq. (2.3.1), thus, resulting in smaller NRR for the same value of a. The impulse response of this filter can be computed using partial fractions:

$$H(z) = \frac{b(1+z^{-1})}{1 - az^{-1}} = A_0 + \frac{A_1}{1 - az^{-1}}\,, \quad \text{where} \quad A_0 = -\frac{b}{a}, \quad A_1 = \frac{b(1+a)}{a}$$

Therefore, its (causal) impulse response will be:

$$h_n = A_0\delta(n) + A_1 a^n u(n)$$

Note, in particular, that $h_0 = A_0 + A_1 = b$. It follows that

$$\mathcal{R} = \sum_{n=0}^{\infty} h_n^2 = h_0^2 + \sum_{n=1}^{\infty} h_n^2 = b^2 + A_1^2 \frac{a^2}{1-a^2} = \frac{1-a}{2}$$

This is slightly smaller than that of Eq. (2.3.2), because of the inequality:

$$\frac{1-a}{2} < \frac{1-a}{1+a}$$

The 3-dB cutoff frequency can be calculated easily in this example. We have

$$|H(\omega_c)|^2 = \frac{2b^2(1+\cos\omega_c)}{1-2a\cos\omega_c + a^2} = \frac{1}{2}$$

which can be solved for ω_c in terms of a:

$$\cos\omega_c = \frac{2a}{1+a^2} \quad \Leftrightarrow \quad \tan\left(\frac{\omega_c}{2}\right) = \frac{1-a}{1+a} \tag{2.3.6}$$

Conversely, we can solve for a in terms of ω_c:

$$a = \frac{1-\sin\omega_c}{\cos\omega_c} = \frac{1-\tan(\omega_c/2)}{1+\tan(\omega_c/2)} \tag{2.3.7}$$

It is easily checked that the condition $0 < a < 1$ requires that $\omega_c < \pi/2$. We note also that the substitution $z \to -z$ changes the filter into a highpass one.

Such simple first-order lowpass or highpass filters with easily controllable widths are useful in many applications, such as the low- and high-frequency shelving filters of audio equalizers [30].

2.4 FIR Averaging Filters

The problem of extracting a constant or a low-frequency signal x_n from the noisy signal $y_n = x_n + v_n$ can also be approached with FIR filters. Consider, for example, the third-order filter:

$$H(z) = h_0 + h_1 z^{-1} + h_2 z^{-2} + h_3 z^{-3}$$

The condition that the constant signal x_n go through the filter unchanged is the condition that the filter have unity gain at DC, which gives the constraint among the filter weights:

$$H(z)\big|_{z=1} = h_0 + h_1 + h_2 + h_3 = 1 \tag{2.4.1}$$

The NRR of this filter will be simply:

$$\mathcal{R} = \sum_n h_n^2 = h_0^2 + h_1^2 + h_2^2 + h_3^2 \tag{2.4.2}$$

The *optimum* third-order FIR filter will be the one that *minimizes* this NRR, subject to the lowpass constraint (2.4.1). To solve this minimization problem, we introduce a Lagrange multiplier λ and incorporate the constraint (2.4.1) into the performance index:

$$\mathcal{J} = \mathcal{R} + \lambda\left(1 - \sum_{n=0}^{3} h_n\right) = \sum_{n=0}^{3} h_n^2 + \lambda\left(1 - \sum_{n=0}^{3} h_n\right) \tag{2.4.3}$$

The minimization can be carried out easily by setting the partial derivatives of $\mathcal{J}$ to zero and solving for the h's:

$$\frac{\partial \mathcal{J}}{\partial h_n} = 2h_n - \lambda = 0 \quad \Rightarrow \quad h_n = \frac{\lambda}{2}, \quad n = 0, 1, 2, 3$$

Thus, all four h's are equal, $h_0 = h_1 = h_2 = h_3 = \lambda/2$. The constraint (2.4.1) then fixes the value of λ to be $1/2$ and we find the optimum weights:

$$h_0 = h_1 = h_2 = h_3 = \frac{1}{4}$$

and the minimized NRR becomes:

$$\mathcal{R}_{\min} = \left(\frac{1}{4}\right)^2 + \left(\frac{1}{4}\right)^2 + \left(\frac{1}{4}\right)^2 + \left(\frac{1}{4}\right)^2 = 4\left(\frac{1}{4}\right)^2 = \frac{1}{4}$$

The I/O equation for this optimum smoothing filter becomes:

$$\hat{x}_n = \frac{1}{4}(y_n + y_{n-1} + y_{n-3} + y_{n-3})$$

More generally, the optimum length-N FIR filter with unity DC gain and minimum NRR is the filter with equal weights:

$$H(z) = \frac{1}{N}\left[1 + z^{-1} + z^{-2} + \cdots + z^{-(N-1)}\right] \tag{2.4.4}$$

and I/O equation:

$$\hat{x}_n = \frac{1}{N}(y_n + y_{n-1} + \cdots + y_{n-N+1}) \tag{2.4.5}$$

with minimized NRR:

$$\mathcal{R} = h_0^2 + h_1^2 + \cdots + h_{N-1}^2 = N \cdot \left(\frac{1}{N}\right)^2 = \frac{1}{N} \tag{2.4.6}$$

Thus, by choosing N large enough, the NRR can be made as small as desired. Again, as the NRR decreases, the filter's time constant ($n_{\text{eff}} = N$) increases.

How does the FIR smoother compare with the IIR smoother of Eq. (2.3.1)? First, we note the IIR smoother is very simple computationally, requiring only 2 MACs[†] per output sample, whereas the FIR requires N MACs.

Second, the FIR smoother typically performs better in terms of both the NRR and the transient response, in the sense that for the same NRR value, the FIR smoother has shorter time constant, and for the same time constant, it has a smaller NRR. We illustrate these remarks below.

Given a time constant $n_{\text{eff}} = \ln\epsilon/\ln a$ for an IIR smoother, the "equivalent" FIR smoother should be chosen to have the same length $N = n_{\text{eff}}$, thus,

$$\boxed{N = \frac{\ln\epsilon}{\ln a}, \quad a = \epsilon^{1/N}} \tag{2.4.7}$$

[†] multiplication-accumulations

For example, if $a = 0.90$ and $\epsilon = 0.01$, then $N = n_{\text{eff}} = 44$. But then, the NRR of the FIR smoother will be $\mathcal{R} = 1/N = 1/44$, which is better than that of the IIR filter, $\mathcal{R} = (1-a)/(1+a) = 1/19$. This case is illustrated in the left graph of Fig. 2.4.1, where the FIR output was computed by Eq. (2.4.5) with $N = 44$ for the same noisy input of Fig. 2.3.2. The IIR output is the same as in that figure.

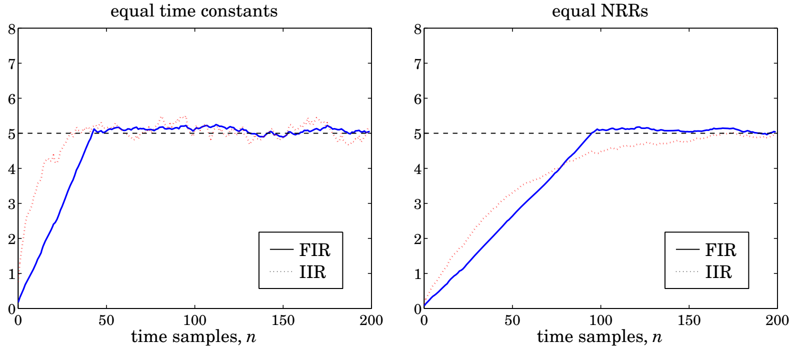


Fig. 2.4.1 Comparison of FIR and IIR smoothing filters.

Similarly, if an IIR smoother achieves a certain NRR value, the "equivalent" FIR filter with the same NRR should have length N such that:

$$\mathcal{R} = \frac{1-a}{1+a} = \frac{1}{N} \quad \Rightarrow \quad \boxed{N = \frac{1+a}{1-a}\,, \quad a = \frac{N-1}{N+1}} \tag{2.4.8}$$

For example, if $a = 0.98$, then we get $N = 99$, which is much shorter than the IIR time constant $n_{\text{eff}} = 228$ computed with $\epsilon = 0.01$. The right graph of Fig. 2.4.1 illustrates this case, where the FIR output was computed by Eq. (2.4.5) with $N = 99$.

An approximate relationship between the IIR time constant n_{eff} and N can be derived in this case as follows. Using the small-x approximation $\ln\big((1+x)/(1-x)\big) \simeq 2x$, we have for large N:

$$\ln(1/a) = \ln\left(\frac{1+(1/N)}{1-(1/N)}\right) \simeq \frac{2}{N}$$

It follows that

$$n_{\text{eff}} = \frac{\ln(1/\epsilon)}{\ln(1/a)} \simeq N\,\frac{1}{2}\ln\big(\frac{1}{\epsilon}\big)$$

Typically, the factor $(\ln(1/\epsilon)/2)$ is greater than one, resulting in a longer IIR time constant n_{eff} than N. For example, we have:

$$\begin{array}{lll}
n_{\text{eff}} = 1.15\,N\,, & \text{if } \epsilon = 10^{-1} & \text{(10\% time constant)} \\
n_{\text{eff}} = 1.50\,N\,, & \text{if } \epsilon = 5\cdot 10^{-2} & \text{(5\% time constant)} \\
n_{\text{eff}} = 2.30\,N\,, & \text{if } \epsilon = 10^{-2} & \text{(1\% or 40-dB time constant)} \\
n_{\text{eff}} = 3.45\,N\,, & \text{if } \epsilon = 10^{-3} & \text{(0.1\% or 60-dB time constant)}
\end{array}$$

Finally, we note that a further advantage of the FIR smoother is that it is a *linear phase* filter. Indeed, using the finite geometric series formula, we can write the transfer function of Eq. (2.4.5) in the form:

$$H(z) = \frac{1}{N}\big(1 + z^{-1} + z^{-2} + \cdots + z^{-(N-1)}\big) = \frac{1}{N}\frac{1-z^{-N}}{1-z^{-1}} \tag{2.4.9}$$

Setting $z = e^{j\omega}$, we obtain the frequency response:

$$\boxed{H(\omega) = \frac{1}{N}\frac{1-e^{-jN\omega}}{1-e^{-j\omega}} = \frac{1}{N}\frac{\sin(N\omega/2)}{\sin(\omega/2)}\,e^{-j\omega(N-1)/2}} \tag{2.4.10}$$

which has a linear phase response. The transfer function (2.4.9) has zeros at the Nth roots of unity, except at $z = 1$, that is,

$$z_k = e^{j\omega_k}\,, \quad \omega_k = \frac{2\pi k}{N}\,, \quad k = 1, 2, \dots, N-1$$

The zeros are distributed equally around the unit circle and tend to suppress the noise spectrum along the Nyquist interval, except at $z = 1$ where there is a pole/zero cancellation and we have $H(z) = 1$.

Fig. 2.4.2 shows the magnitude and phase response of $H(\omega)$ for $N = 16$. Note that the phase response is *piece-wise linear* with slope $(N-1)/2$. It exhibits 180° jumps at $\omega = \omega_k$, where the factor $\sin(N\omega/2)/\sin(\omega/2)$ changes algebraic sign.

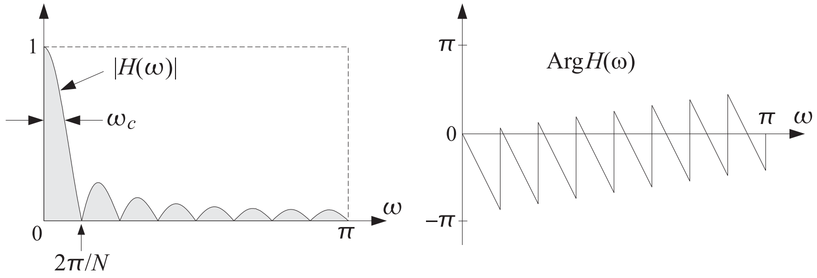


Fig. 2.4.2 Magnitude and phase responses of FIR smoother, for $N = 16$.

The 3-dB cutoff frequency of the filter is somewhat less than half the base of the mainlobe, that is,

$$\boxed{\omega_c = \frac{0.886\pi}{N}} \tag{2.4.11}$$

It corresponds to a drop of the magnitude response squared by a factor of 1/2. Indeed, setting $\omega/2 = \omega_c/2 = 0.443\pi/N$ in (2.4.10), we have

$$\left|\frac{1}{N}\frac{\sin(N\,0.443\pi/N)}{\sin(0.443\pi/N)}\right|^2 \simeq \left|\frac{1}{N}\frac{\sin(0.443\pi)}{(0.443\pi/N)}\right|^2 = \left|\frac{\sin(0.443\pi)}{0.443\pi}\right|^2 \simeq \frac{1}{2}$$

where we used the approximation $\sin(\pi/2N) \simeq \pi/2N$, for large N. In decibels, we have $-20\log_{10}\big(\sin(0.443\pi)/0.443\pi\big) = 3.01$ dB, hence, the name "3-dB frequency."

Like its IIR counterpart of Eq. (2.3.1), the FIR averaging filter (2.4.5) can be applied to any low-frequency signal x_n—not just a constant signal. The averaging of the N successive samples in Eq. (2.4.5) tends to smooth out the highly fluctuating noise component v_n, while it leaves the slowly varying component x_n almost unchanged.

However, if x_n is not so slowly varying, the filter will also tend to average out these variations, especially when the averaging operation (2.4.5) reaches across many time samples when N is large. In the frequency domain, the same conclusion follows by noting that as N increases, the filter's cutoff frequency ω_c decreases, thus removing more and more of the higher frequencies that might be present in the desired signal.

Thus, there is a limit to the applicability of this type of smoothing filter: Its length must be chosen to be large enough to reduce the noise, but not so large as to start distorting the desired signal by smoothing it too much.

A rough quantitative criterion for the selection of the length N is as follows. If it is known that the desired signal x_n contains significant frequencies up to a maximum frequency, say $\omega_{\max}$, then we may choose N such that $\omega_{\max} \le \omega_c = 0.886\pi/N$, which gives $N \le 0.886\pi/\omega_{\max}$.

The FIR averaging filter can also be implemented in a *recursive form* based on the summed version of the transfer function (2.4.9). For example, the direct-form realization of $H(z)$ is described by the I/O difference equation:

$$\hat{x}_n = \hat{x}_{n-1} + \frac{1}{N}\big[y_n - y_{n-N}\big] \qquad (2.4.12)$$

Because of the pole-zero cancellation implicit in (2.4.12) such implementation is prone to roundoff accumulation errors and instabilities, and therefore, not recommended for continuous real-time processing even though it is efficient computationally.

The FIR smoothing filter will be considered in further detail in Sec. 3.1, generalized to local polynomial smoothing filters that minimize the NRR subject to additional linear constraints on the filter weights. In Sec. 4.2, it is generalized to minimum-roughness filters that minimize a filtered version of the NRR subject to similar constraints.

Like the IIR smoother, the FIR smoother and its generalizations are widely used in many data analysis applications. It is also useful in de-seasonalizing applications, where if N is chosen to be the seasonal period, the filter's Nth root of unity zeros coincide with the harmonics of the seasonal component so that the filter will extract the smooth trend while eliminating the seasonal part.

2.5 Problems

2.1 Show that the z-domain transformation, $z \to -z$, maps a lowpass filter into a highpass one. Show that under this transformation, the impulse response of the lowpass filter h_n gets mapped into $(-1)^n h_n$.

2.2 Given the real-valued impulse response h_n of a lowpass filter, show that the filter with the complex-valued impulse response $e^{j\omega_0 n}h_n$ defines a bandpass filter centered at ω_0. What sort of filter is defined by the real-valued impulse response $\cos(\omega_0 n)h_n$? Explain how the previous problem is a special case of this problem.

2.3 *Highpass Signal Extraction.* Design a first-order IIR filter to extract the high-frequency $x_n = (-1)^n s$ from the noisy signal

$$y_n = x_n + v_n = (-1)^n s + v_n$$

where s is a constant amplitude and v_n is zero-mean, white Gaussian noise with variance σ_v^2. Start by converting the two lowpass filters given in Sec. 2.3 into highpass filters. For each of the resulting filters, plot the corresponding magnitude response and calculate the NRR in terms of the pole parameter a.

For the values of the parameters $s = 2$ and $a = 0.99$, compute 200 samples of the signal y_n and process it through your filters and plot the output. Discuss the transient effect vs. the signal extraction ability of the filters.

2.4 *Bandpass Signal Extraction.* A noisy sinusoid of frequency $f_0 = 500$ Hz is sampled at a rate of $f_s = 10$ kHz:

$$y_n = x_n + v_n = \cos(\omega_0 n) + v_n$$

where $\omega_0 = 2\pi f_0/f_s$ and v_n is a zero-mean, unit-variance, white Gaussian noise signal. The sinusoid can be extracted by a bandpass resonator-like filter of the form:

$$H(z) = \frac{G}{(1 - Re^{j\omega_0}z^{-1})(1 - Re^{-j\omega_0}z^{-1})} = \frac{G}{1 - 2R\cos\omega_0\, z^{-1} + R^2 z^{-2}}$$

Its poles are at $z = Re^{\pm j\omega_0}$ with $0 < R < 1$. For R near unity, the 3-dB width of this filter is given approximately by $\Delta\omega = 2(1-R)$.

Fix the gain factor G by requiring that the filter have unity gain at ω_0, that is, $|H(\omega_0)| = 1$. Then, show that the NRR of this filter is given by:

$$\mathcal{R} = \sum_{n=0}^{\infty} h_n^2 = \frac{(1-R)(1+R^2)(1-2R\cos(2\omega_0)+R^2)}{(1+R)(1-2R^2\cos(2\omega_0)+R^4)}$$

For the values of the parameters $R = 0.99$ and $\omega_0 = 0.1\pi$, plot the magnitude response of this filter and indicate on the graph its 3-dB width. Calculate the corresponding NRR.

Then, calculate and plot 300 samples of the noisy signal y_n, and process it through the filter. On a separate graph, plot the resulting estimate $\hat{x}_n$ together with the desired signal x_n.

Discuss the signal extraction capability of this filter vs. the transient effects vs. the delay shift introduced by the filter's phase delay $d(\omega) = -\text{Arg}\,H(\omega)/\omega$, and calculate the amount of delay $d(\omega_0)$ at ω_0 and indicate it on the graph.

3

Local Polynomial Filters

3.1 Introduction

We mentioned in Sec. 2.4 that there are limits to the applicability of the plain FIR averager filter—in order to achieve a high degree of noise reduction, its length N may be required to be so large that the filter's passband becomes smaller than the signal bandwidth, causing the removal of useful high frequencies from the desired signal.

In other words, in its attempt to smooth out the noise v_n, the filter begins to smooth out the desired signal x_n to an unacceptable degree. For example, if x_n contains some short-duration peaks, corresponding to the higher frequencies present in x_n, and the filter's length N is longer than the duration of the peaks, the filter will tend to smooth the peaks too much, broadening them and reducing their height.

Local polynomial smoothing filters [36–99] are generalizations of the FIR averager filter that can preserve better the higher frequency content of the desired signal, at the expense of not removing as much noise as the averager. They can be characterized in three equivalent ways:

1. They are the optimal lowpass filters that minimize the NRR, subject to additional constraints than the DC unity-gain condition (2.4.1)—the constraints being equivalent to the requirement that polynomial input signals go through the filter unchanged.
2. They are the optimal filters that minimize the NRR whose frequency response $H(\omega)$ satisfies certain *flatness* constraints at DC.
3. They are the filters that optimally fit, in a least-squares sense, a set of data points to polynomials of different degrees.

Local polynomial smoothing (LPSM) filters have a long history and have been rediscovered repeatedly in different contexts. They were originally derived in 1866 by the Italian astronomer Schiaparelli [36] who formulated the problem as the minimization of the NRR subject to polynomial-preserving constraints and derived the filters in complete generality, discussing also the case of even-length filters. They were rederived in 1871 by De Forest [65] who generalized them further to include the case of "minimum-roughness" or minimum-R_s filters. Subsequently, they were rediscovered many times and used extensively in actuarial applications, for example, by Gram, Hardy, Sheppard, Henderson, and others. See Refs. [68–75] for the development and history of these filters. In the actuarial context, smoothing is referred to as the process of "graduation." They were revived again in the 1960s by Savitzky and Golay [42] and have been applied widely in chemistry and spectroscopy [42–53] known in that context as *Savitzky-Golay filters*. They, and their minimum-R_s versions [65–99] known typically as *Henderson filters*, are used routinely for trend extraction in financial, business, and census applications.

Some recent incarnations also include predictive FIR interpolation, differentiation, fractional-delay, and maximally-flat filters [152–187], and applications to the representation of speech and images in terms of orthogonal-polynomial moments [137–150].

The least-squares polynomial fitting approach also has a long history. Chebyshev [104] derived in 1864 the discrete Chebyshev orthogonal polynomials,[‡] also known as Gram polynomials, which provide convenient and computationally efficient bases for the solution of the least-squares problem and the design of local polynomial filters. Several applications and reviews of the discrete Chebyshev orthogonal polynomials may be found in [104–151]. The minimum-R_s Henderson filters also admit similar efficient representations in terms of the Hahn orthogonal polynomials, a special case of which are the discrete Chebyshev polynomials. We discuss Henderson filters in Sec. 4.2 and orthogonal polynomial bases in Sec. 4.3.

3.2 Local Polynomial Fitting

We begin with the least-squares polynomial fitting approach. We assume that the signal model for the observations is:

$$y_n = x_n + v_n$$

where v_n is white noise and x_n is a smooth signal to be estimated. Fig. 3.2.1 shows five noisy signal samples $[y_{-2}, y_{-1}, y_0, y_1, y_2]$ positioned symmetrically about the origin. Later on, we will shift them to an arbitrary position along the time axis. Polynomial smoothing of the five samples is equivalent to replacing them by the values that lie on smooth polynomial curves drawn between the noisy samples. In Fig. 3.2.1, we consider fitting the five data to a constant signal, a linear signal, and a quadratic signal.

The corresponding smoothed values are given by the 0th, 1st, and 2nd degree polynomials defined for $m = -2, -1, 0, 1, 2$:

$$\begin{aligned} \hat{y}_m &= c_0 && \text{(constant)} \\ \hat{y}_m &= c_0 + c_1 m && \text{(linear)} \\ \hat{y}_m &= c_0 + c_1 m + c_2 m^2 && \text{(quadratic)} \end{aligned} \tag{3.2.1}$$

For each choice of the polynomial order, the coefficients c_i must be determined optimally such that the corresponding polynomial curve best fits the given data. This can be accomplished by a *least-squares fit*, which chooses the c_i that minimize the total mean-square error. For example, in the quadratic case, we have the performance index:

$$\mathcal{J} = \sum_{m=-2}^{2} e_m^2 = \sum_{m=-2}^{2} \left(y_m - (c_0 + c_1 m + c_2 m^2)\right)^2 = \min \tag{3.2.2}$$

[‡] not to be confused with the ordinary Chebyshev polynomials.

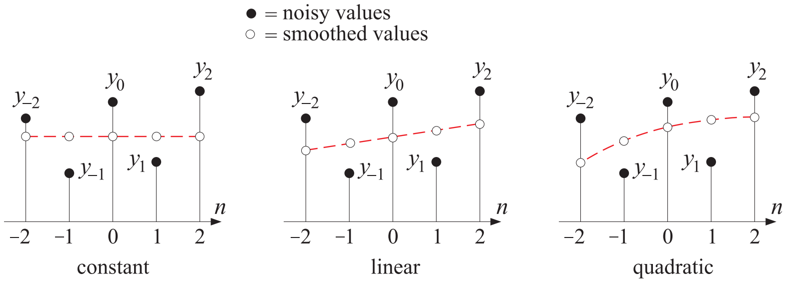


Fig. 3.2.1 Data smoothing with polynomials of degrees $d = 0, 1, 2$.

where the fitting errors are defined as

$$e_m = y_m - \hat{y}_m = y_m - (c_0 + c_1 m + c_2 m^2), \qquad m = -2, -1, 0, 1, 2$$

It proves convenient to express Eqs. (3.2.1) and (3.2.2) in a vectorial form, which generalizes to higher polynomial orders and to more than five data points. We define the five-dimensional vectors of data, estimates, and errors:

$$\mathbf{y} = \begin{bmatrix} y_{-2} \\ y_{-1} \\ y_0 \\ y_1 \\ y_2 \end{bmatrix}, \qquad \hat{\mathbf{y}} = \begin{bmatrix} \hat{y}_{-2} \\ \hat{y}_{-1} \\ \hat{y}_0 \\ \hat{y}_1 \\ \hat{y}_2 \end{bmatrix}, \qquad \mathbf{e} = \begin{bmatrix} e_{-2} \\ e_{-1} \\ e_0 \\ e_1 \\ e_2 \end{bmatrix} = \mathbf{y} - \hat{\mathbf{y}}$$

Similarly, we define the five-dimensional polynomial *basis vectors* $\mathbf{s}_0, \mathbf{s}_1, \mathbf{s}_2$, whose components are:

$$s_0(m) = 1, \qquad s_1(m) = m, \qquad s_2(m) = m^2, \qquad -2 \le m \le 2$$

Vectorially, we have:

$$\mathbf{s}_0 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}, \qquad \mathbf{s}_1 = \begin{bmatrix} -2 \\ -1 \\ 0 \\ 1 \\ 2 \end{bmatrix}, \qquad \mathbf{s}_2 = \begin{bmatrix} 4 \\ 1 \\ 0 \\ 1 \\ 4 \end{bmatrix} \tag{3.2.3}$$

In this notation, we may write the third of Eq. (3.2.1) vectorially:

$$\hat{\mathbf{y}} = c_0 \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} + c_1 \begin{bmatrix} -2 \\ -1 \\ 0 \\ 1 \\ 2 \end{bmatrix} + c_2 \begin{bmatrix} 4 \\ 1 \\ 0 \\ 1 \\ 4 \end{bmatrix} = c_0\mathbf{s}_0 + c_1\mathbf{s}_1 + c_2\mathbf{s}_2$$

Therefore,

$$\hat{\mathbf{y}} = c_0\mathbf{s}_0 + c_1\mathbf{s}_1 + c_2\mathbf{s}_2 = [\mathbf{s}_0, \mathbf{s}_1, \mathbf{s}_2] \begin{bmatrix} c_0 \\ c_1 \\ c_2 \end{bmatrix} \equiv S\mathbf{c} \tag{3.2.4}$$

The 5×3 basis matrix S has as columns the three basis vectors $\mathbf{s}_0, \mathbf{s}_1, \mathbf{s}_2$. It is given explicitly as follows:

$$S = [\mathbf{s}_0, \mathbf{s}_1, \mathbf{s}_2] = \begin{bmatrix} 1 & -2 & 4 \\ 1 & -1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \end{bmatrix} \tag{3.2.5}$$

Writing $\mathbf{e} = \mathbf{y} - \hat{\mathbf{y}} = \mathbf{y} - S\mathbf{c}$, we can express the performance index (3.2.2) as the dot product:

$$\mathcal{J} = \mathbf{e}^T\mathbf{e} = (\mathbf{y} - S\mathbf{c})^T(\mathbf{y} - S\mathbf{c}) = \min \tag{3.2.6}$$

To minimize this expression with respect to $\mathbf{c}$, we set the gradient $\partial\mathcal{J}/\partial\mathbf{c}$ to zero:

$$\frac{\partial\mathcal{J}}{\partial\mathbf{c}} = -2S^T\mathbf{e} = -2S^T(\mathbf{y} - S\mathbf{c}) = -2(S^T\mathbf{y} - S^TS\mathbf{c}) = 0 \tag{3.2.7}$$

Therefore, the minimization condition gives the so-called *orthogonality equations* and the equivalent *normal equations*:

$$\frac{\partial\mathcal{J}}{\partial\mathbf{c}} = 0 \quad \Leftrightarrow \quad S^T\mathbf{e} = 0 \quad \Leftrightarrow \quad S^TS\mathbf{c} = S^T\mathbf{y} \tag{3.2.8}$$

with optimal solution:

$$\mathbf{c} = (S^TS)^{-1}S^T\mathbf{y} \equiv G^T\mathbf{y} \tag{3.2.9}$$

where we defined the 5×3 matrix G by

$$G = S(S^TS)^{-1} \tag{3.2.10}$$

We note that the solution (3.2.9) is none other than the unique least-squares solution of the full-rank overdetermined linear system $S\mathbf{c} = \mathbf{y}$, as given for example by Eq. (15.4.10), $\mathbf{c} = S^+\mathbf{y}$, where $S^+ = (S^TS)^{-1}S^T$ is the corresponding pseudoinverse. Inserting the optimal coefficients $\mathbf{c}$ into Eq. (3.2.4), we find the smoothed values:[†]

$$\hat{\mathbf{y}} = S\mathbf{c} = SG^T\mathbf{y} = S(S^TS)^{-1}S^T\mathbf{y} \equiv B^T\mathbf{y} \tag{3.2.11}$$

where we defined the 5×5 matrix B by

$$B = B^T = SG^T = GS^T = S(S^TS)^{-1}S^T \tag{3.2.12}$$

The symmetric 3×3 matrix $F = S^TS$, which appears in the expressions for G and B, has matrix elements that are the *dot products* of the basis vectors, that is, the ijth matrix element is $F_{ij} = (S^TS)_{ij} = \mathbf{s}_i^T\mathbf{s}_j$. Indeed, using Eq. (3.2.5), we find:

$$F = S^TS = \begin{bmatrix} \mathbf{s}_0^T \\ \mathbf{s}_1^T \\ \mathbf{s}_2^T \end{bmatrix} [\mathbf{s}_0, \mathbf{s}_1, \mathbf{s}_2] = \begin{bmatrix} \mathbf{s}_0^T\mathbf{s}_0 & \mathbf{s}_0^T\mathbf{s}_1 & \mathbf{s}_0^T\mathbf{s}_2 \\ \mathbf{s}_1^T\mathbf{s}_0 & \mathbf{s}_1^T\mathbf{s}_1 & \mathbf{s}_1^T\mathbf{s}_2 \\ \mathbf{s}_2^T\mathbf{s}_0 & \mathbf{s}_2^T\mathbf{s}_1 & \mathbf{s}_2^T\mathbf{s}_2 \end{bmatrix} \tag{3.2.13}$$

[†] although B is symmetric, we prefer to write $\hat{\mathbf{y}} = B^T\mathbf{y}$, which generalizes to the non-symmetric case of minimum-roughness filters of Sec. 4.2.

Using Eq. (3.2.5), we calculate F and its inverse F^{-1}:

$$F = \begin{bmatrix} 5 & 0 & 10 \\ 0 & 10 & 0 \\ 10 & 0 & 34 \end{bmatrix}, \qquad F^{-1} = \frac{1}{35}\begin{bmatrix} 17 & 0 & -5 \\ 0 & 3.5 & 0 \\ -5 & 0 & 2.5 \end{bmatrix} \tag{3.2.14}$$

Then, we calculate the 5×3 matrix $G = S(S^TS)^{-1} = SF^{-1}$:

$$G = SF^{-1} = \frac{1}{35}\begin{bmatrix} 1 & -2 & 4 \\ 1 & -1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \end{bmatrix}\begin{bmatrix} 17 & 0 & -5 \\ 0 & 3.5 & 0 \\ -5 & 0 & 2.5 \end{bmatrix} \quad \text{or,}$$

$$G = \frac{1}{35}\begin{bmatrix} -3 & -7 & 5 \\ 12 & -3.5 & -2.5 \\ 17 & 0 & -5 \\ 12 & 3.5 & -2.5 \\ -3 & 7 & 5 \end{bmatrix} \equiv [\mathbf{g}_0, \mathbf{g}_1, \mathbf{g}_2] \tag{3.2.15}$$

As we see below, the three columns of G have useful interpretations as differentiation filters. Next, using Eq. (3.2.12), we calculate the 5×5 matrix B:

$$B = GS^T = \frac{1}{35}\begin{bmatrix} -3 & -7 & 5 \\ 12 & -3.5 & -2.5 \\ 17 & 0 & -5 \\ 12 & 3.5 & -2.5 \\ -3 & 7 & 5 \end{bmatrix}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ -2 & -1 & 0 & 1 & 2 \\ 4 & 1 & 0 & 1 & 4 \end{bmatrix} \quad \text{or,}$$

$$B = \frac{1}{35}\begin{bmatrix} 31 & 9 & -3 & -5 & 3 \\ 9 & 13 & 12 & 6 & -5 \\ -3 & 12 & 17 & 12 & -3 \\ -5 & 6 & 12 & 13 & 9 \\ 3 & -5 & -3 & 9 & 31 \end{bmatrix} \equiv [\mathbf{b}_{-2}, \mathbf{b}_{-1}, \mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2] \tag{3.2.16}$$

Because B is symmetric, its rows are the same as its columns. Thus, we can write it either in column-wise or row-wise form:

$$B = [\mathbf{b}_{-2}, \mathbf{b}_{-1}, \mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2] = \begin{bmatrix} \mathbf{b}_{-2}^T \\ \mathbf{b}_{-1}^T \\ \mathbf{b}_0^T \\ \mathbf{b}_1^T \\ \mathbf{b}_2^T \end{bmatrix} = B^T$$

The five columns or rows of B are the LPSM filters of length 5 and polynomial order 2. The corresponding smoothed values $\hat{\mathbf{y}}$ can be expressed component-wise in terms of these filters, as follows:

$$\begin{bmatrix} \hat{y}_{-2} \\ \hat{y}_{-1} \\ \hat{y}_0 \\ \hat{y}_1 \\ \hat{y}_2 \end{bmatrix} = \hat{\mathbf{y}} = B^T\mathbf{y} = \begin{bmatrix} \mathbf{b}_{-2}^T \\ \mathbf{b}_{-1}^T \\ \mathbf{b}_0^T \\ \mathbf{b}_1^T \\ \mathbf{b}_2^T \end{bmatrix}\mathbf{y} = \begin{bmatrix} \mathbf{b}_{-2}^T\mathbf{y} \\ \mathbf{b}_{-1}^T\mathbf{y} \\ \mathbf{b}_0^T\mathbf{y} \\ \mathbf{b}_1^T\mathbf{y} \\ \mathbf{b}_2^T\mathbf{y} \end{bmatrix}$$

or, for $m = -2, -1, 0, 1, 2$:

$$\hat{y}_m = \mathbf{b}_m^T\mathbf{y} \tag{3.2.17}$$

and more explicitly,

$$\begin{bmatrix} \hat{y}_{-2} \\ \hat{y}_{-1} \\ \hat{y}_0 \\ \hat{y}_1 \\ \hat{y}_2 \end{bmatrix} = \frac{1}{35}\begin{bmatrix} 31 & 9 & -3 & -5 & 3 \\ 9 & 13 & 12 & 6 & -5 \\ -3 & 12 & 17 & 12 & -3 \\ -5 & 6 & 12 & 13 & 9 \\ 3 & -5 & -3 & 9 & 31 \end{bmatrix}\begin{bmatrix} y_{-2} \\ y_{-1} \\ y_0 \\ y_1 \\ y_2 \end{bmatrix} \tag{3.2.18}$$

Thus, the mth filter $\mathbf{b}_m$ dotted into the data vector $\mathbf{y}$ generates the mth smoothed data sample. In a similar fashion, we can express the polynomial coefficients c_i as dot products. Using the solution Eq. (3.2.9), we have

$$\begin{bmatrix} c_0 \\ c_1 \\ c_2 \end{bmatrix} = \mathbf{c} = G^T\mathbf{y} = \begin{bmatrix} \mathbf{g}_0^T \\ \mathbf{g}_1^T \\ \mathbf{g}_2^T \end{bmatrix}\mathbf{y} = \begin{bmatrix} \mathbf{g}_0^T\mathbf{y} \\ \mathbf{g}_1^T\mathbf{y} \\ \mathbf{g}_2^T\mathbf{y} \end{bmatrix}$$

or, explicitly,

$$\begin{bmatrix} c_0 \\ c_1 \\ c_2 \end{bmatrix} = \frac{1}{35}\begin{bmatrix} -3 & 12 & 17 & 12 & -3 \\ -7 & -3.5 & 0 & 3.5 & 7 \\ 5 & -2.5 & -5 & -2.5 & 5 \end{bmatrix}\begin{bmatrix} y_{-2} \\ y_{-1} \\ y_0 \\ y_1 \\ y_2 \end{bmatrix} \tag{3.2.19}$$

Thus, the coefficients c_i can be expressed as the dot products of the columns of G with the data vector $\mathbf{y}$:

$$c_i = \mathbf{g}_i^T\mathbf{y}, \qquad i = 0, 1, 2 \tag{3.2.20}$$

Of the five columns of B, the middle one, $\mathbf{b}_0$, is the most important because it smooths the value y_0, which is symmetrically placed with respect to the other samples in $\mathbf{y}$, as shown in Fig. 3.2.1.

In smoothing a long block of data, the filter $\mathbf{b}_0$ is used during the *steady-state* period, whereas the other columns of B are used only during the input-on and input-off *transients*. We will refer to $\mathbf{b}_0$ and the other columns of B as the *steady-state* and *transient* LPSM filters.

Setting $m = 0$ into Eq. (3.2.1), we note that the middle smoothed value $\hat{y}_0$ is equal to the polynomial coefficient c_0. Using Eqs. (3.2.17) and (3.2.20), we find: $\hat{y}_0 = c_0 = \mathbf{b}_0^T\mathbf{y} = \mathbf{g}_0^T\mathbf{y}$ (the middle column of B and the first column of G are always the same, $\mathbf{b}_0 = \mathbf{g}_0$.)

To express (3.2.18) as a true filtering operation acting on an input sequence y_n, we shift the group of five samples to be centered around the nth time instant, that is, we make the substitution:

$$[y_{-2}, y_{-1}, y_0, y_1, y_2] \longrightarrow [y_{n-2}, y_{n-1}, y_n, y_{n+1}, y_{n+2}]$$

The corresponding five smoothed values will be then:

$$\begin{bmatrix} \hat{y}_{n-2} \\ \hat{y}_{n-1} \\ \hat{y}_n \\ \hat{y}_{n+1} \\ \hat{y}_{n+2} \end{bmatrix} = \frac{1}{35} \begin{bmatrix} 31 & 9 & -3 & -5 & 3 \\ 9 & 13 & 12 & 6 & -5 \\ -3 & 12 & 17 & 12 & -3 \\ -5 & 6 & 12 & 13 & 9 \\ 3 & -5 & -3 & 9 & 31 \end{bmatrix} \begin{bmatrix} y_{n-2} \\ y_{n-1} \\ y_n \\ y_{n+1} \\ y_{n+2} \end{bmatrix} \qquad (3.2.21)$$

In particular, the middle sample y_n is smoothed by the filter $\mathbf{b}_0$:

$$\boxed{\hat{x}_n = \frac{1}{35}\big(-3y_{n-2} + 12y_{n-1} + 17y_n + 12y_{n+1} - 3y_{n+2}\big)} \qquad (3.2.22)$$

where, in accordance with our assumed model of noisy observations $y_n = x_n + v_n$, we denoted $\hat{y}_n$ by $\hat{x}_n$, i.e., the estimated value of x_n.

The other estimated values $\{\hat{y}_{n+m}\,,\; m = \pm1, \pm2\}$, are not used for smoothing, except, as we see later, at the beginning and end of the signal block y_n. They may be used, however, for prediction and interpolation.

The filter (3.2.22) corresponds to fitting every group of five samples $\{y_{n-2}, y_{n-1}, y_n, y_{n+1}, y_{n+2}\}$ to a quadratic polynomial and replacing the middle sample y_n by its smoothed value $\hat{x}_n$. It is a lowpass filter and is normalized to unity gain at DC, because its coefficients add up to one.

Its NRR is the sum of the squared filter coefficients. It can be proved in general that the NRR of any steady-state filter $\mathbf{b}_0$ is equal to the *middle* value of its impulse response, that is, the coefficient $b_0(0)$. Therefore,

$$\mathcal{R} = \mathbf{b}_0^T\mathbf{b}_0 = \sum_{m=-2}^{2} b_0(m)^2 = b_0(0) = \frac{17}{35} = \frac{17/7}{5} = \frac{2.43}{5} = 0.49$$

By comparison, the length-5 FIR averager operating on the same five samples is:

$$\boxed{\hat{x}_n = \frac{1}{5}\big(y_{n-2} + y_{n-1} + y_n + y_{n+1} + y_{n+2}\big)} \qquad (3.2.23)$$

with $\mathcal{R} = 1/N = 1/5$. Thus, the length-5 quadratic-polynomial filter performs 2.43 times worse in reducing noise than the FIR averager. However, the higher-order polynomial filters have other advantages to be discussed later.

We saw that the coefficient c_0 represents the smoothed value of y_0 at $m = 0$. Similarly, the coefficient c_1 represents the slope, the derivative, of y_0 at $m = 0$. Indeed, we have from Eq. (3.2.1) by differentiating and setting $m = 0$:

$$\dot{\hat{y}}_0 = \frac{d\hat{y}_m}{dm}\bigg|_0 = c_1\,, \qquad \ddot{\hat{y}}_0 = \frac{d^2\hat{y}_m}{dm^2}\bigg|_0 = 2c_2$$

Thus, c_1 and $2c_2$ represent the polynomial estimates of the first and second derivatives at $m = 0$. Using Eq. (3.2.20) we can express them in terms of the second and third columns of the matrix G:

$$\begin{aligned} \dot{\hat{y}}_0 &= c_1 = \mathbf{g}_1^T\mathbf{y} \\ \ddot{\hat{y}}_0 &= 2c_2 = 2\mathbf{g}_2^T\mathbf{y} \end{aligned} \qquad (3.2.24)$$

Shifting these to the nth time sample, and denoting them by $\dot{\hat{x}}_n$ and $\ddot{\hat{x}}_n$, we find the length-5 local polynomial filters for estimating the *first and second derivatives* of x_n:

$$\boxed{\begin{aligned} \dot{\hat{x}}_n &= \frac{1}{35}\big(-7y_{n-2} - 3.5y_{n-1} + 3.5y_{n+1} + 7y_{n+2}\big) \\ \ddot{\hat{x}}_n &= \frac{2}{35}\big(5y_{n-2} - 2.5y_{n-1} - 5y_n - 2.5y_{n+1} + 5y_{n+2}\big) \end{aligned}} \qquad (3.2.25)$$

The above designs can be generalized in a straightforward manner to an arbitrary degree d of the fitted polynomial and to an arbitrary length N of the data vector $\mathbf{y}$. We require only that $d \le N - 1$, a restriction to be clarified later. Assuming that N is odd, say, $N = 2M + 1$, the five-dimensional data vector $\mathbf{y} = [y_{-2}, y_{-1}, y_0, y_1, y_2]^T$ is replaced by an N-dimensional one, having M points on either side of y_0:

$$\mathbf{y} = [y_{-M}, \dots, y_{-1}, y_0, y_1, \dots, y_M]^T \qquad (3.2.26)$$

The N data samples in $\mathbf{y}$ are then fitted by a polynomial of degree d:

$$\hat{y}_m = c_0 + c_1 m + \cdots + c_d m^d, \qquad -M \le m \le M \qquad (3.2.27)$$

In this case, there are $d{+}1$ polynomial basis vectors $\mathbf{s}_i$, $i = 0, 1, \dots, d$, defined to have components:

$$s_i(m) = m^i, \qquad -M \le m \le M \qquad (3.2.28)$$

The corresponding $N{\times}(d{+}1)$ basis matrix S is defined to have the $\mathbf{s}_i$ as columns:

$$S = [\mathbf{s}_0, \mathbf{s}_1, \dots, \mathbf{s}_d] \qquad (3.2.29)$$

The smoothed values (3.2.27) can be written in the vector form:

$$\hat{\mathbf{y}} = \sum_{i=0}^{d} c_i\mathbf{s}_i = [\mathbf{s}_0, \mathbf{s}_1, \dots, \mathbf{s}_d] \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_d \end{bmatrix} = S\mathbf{c} \qquad (3.2.30)$$

The design steps for the LPSM filters can be summarized then as follows:

$$\begin{aligned} F &= S^T S \quad \Leftrightarrow \quad F_{ij} = \mathbf{s}_i^T\mathbf{s}_j, \quad i,j = 0, 1, \dots, d \\ G &= SF^{-1} \equiv [\mathbf{g}_0, \mathbf{g}_1, \dots, \mathbf{g}_d] \\ B &= GS^T = SF^{-1}S^T \equiv [\mathbf{b}_{-M}, \dots, \mathbf{b}_0, \dots, \mathbf{b}_M] \end{aligned} \qquad (3.2.31)$$

The corresponding coefficient vector $\mathbf{c}$ and smoothed data vector $\hat{\mathbf{y}}$ will be:

$$\begin{aligned} \mathbf{c} &= G^T\mathbf{y} \quad \Leftrightarrow \quad c_i = \mathbf{g}_i^T\mathbf{y}, \quad i = 0,1,\dots,d \\ \hat{\mathbf{y}} &= B^T\mathbf{y} \quad \Leftrightarrow \quad \hat{y}_m = \mathbf{b}_m^T\mathbf{y}, \quad -M \le m \le M \end{aligned} \tag{3.2.32}$$

The middle smoothed value $\hat{y}_0$ is given in terms of the middle LPSM filter $\mathbf{b}_0$:

$$\hat{y}_0 = \mathbf{b}_0^T\mathbf{y} = \sum_{k=-M}^{M} b_0(k)y_k$$

The N-dimensional vector $\mathbf{y} = [y_{-M},\dots,y_{-1},y_0,y_1,\dots,y_M]^T$ can be shifted to the nth time instant by the replacement:

$$[y_{-M},\,\dots\,,y_{-1},\,y_0,\,y_1,\,\dots\,,y_M] \longrightarrow [y_{n-M},\,\dots\,,y_{n-1},\,y_n,\,y_{n+1},\,\dots\,,y_{n+M}]$$

The resulting length-N, order-d, LPSM filter for smoothing a noisy sequence y_n will be, in its steady-state form (denoting again $\hat{x}_n = \hat{y}_n$):

$$\boxed{\hat{x}_n = \hat{y}_n = \sum_{k=-M}^{M} b_0(k)y_{n+k} = \sum_{k=-M}^{M} b_0(-k)y_{n-k}} \tag{3.2.33}$$

The second equation expresses the output in convolutional form.† Because the filter $\mathbf{b}_0$ is symmetric about its middle, we can replace $b_0(-k) = b_0(k)$. The non-central estimated values are obtained from the $\mathbf{b}_m$ filters:

$$\boxed{\hat{y}_{n+m} = \sum_{k=-M}^{M} b_m(k)y_{n+k} = \sum_{k=-M}^{M} b_m^R(k)y_{n-k}}\,, \quad -M \le m \le M \tag{3.2.34}$$

These filters satisfy the symmetry property $b_m^R(k) = b_m(-k) = b_{-m}(k)$ and can be used for prediction, as we discuss later.

The $d+1$ columns of the $N\times(d+1)$-dimensional matrix G give the LPSM *differentiation filters*, for derivatives of orders $i = 0,1,\dots,d$. It follows by differentiating Eq. (3.2.27) i times and setting $m = 0$:

$$\hat{y}_0^{(i)} = \frac{d^i\hat{y}_m}{dm^i}\bigg|_0 = i!\,c_i = i!\,\mathbf{g}_i^T\mathbf{y}$$

Shifting these to time n, gives the differentiation convolutional filtering equations:

$$\hat{x}_n^{(i)} = i!\sum_{m=-M}^{M} g_i^R(m)y_{n-m}, \qquad i = 0,1,\dots,d \tag{3.2.35}$$

where, $g_i^R(m) = g_i(-m)$ and as in Eq. (3.2.33), we reversed the order of writing the terms, but here the filters $\mathbf{g}_i$ are not necessarily symmetric (actually, they are symmetric for even i, and antisymmetric for odd i.)

†We use the notation $\mathbf{b}^R$ to denote the reverse of a double-sided filter $\mathbf{b}$, that is, $b^R(k) = b(-k)$.

Example 3.2.1: We construct the length-5 LPSM filters for the cases $d = 0$ and $d = 1$. For $d = 0$, corresponding to the constant $\hat{y}_m = c_0$ in Eq. (3.2.1), there is only one basis vector $\mathbf{s}_0$ defined in Eq. (3.2.3). The basis matrix $S = [\mathbf{s}_0]$ will have just one column, and the matrix F will be the scalar

$$F = S^TS = \mathbf{s}_0^T\mathbf{s}_0 = [1,1,1,1,1]\begin{bmatrix} 1\\1\\1\\1\\1 \end{bmatrix} = 5$$

The matrix G will then be

$$G = SF^{-1} = \frac{1}{5}\mathbf{s}_0 = \frac{1}{5}[1,1,1,1,1]^T$$

resulting in the LPSM matrix B:

$$B = GS^T = \frac{1}{5}\mathbf{s}_0\mathbf{s}_0^T = \frac{1}{5}\begin{bmatrix} 1\\1\\1\\1\\1 \end{bmatrix}[1,1,1,1,1] = \frac{1}{5}\begin{bmatrix} 1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1 \end{bmatrix}$$

Thus, the steady-state LPSM filter is the length-5 averager:

$$\mathbf{b}_0 = \frac{1}{5}[1,1,1,1,1]^T$$

For the case $d = 1$, corresponding to the linear fit $\hat{y}_m = c_0 + c_1m$, we have the two basis vectors $\mathbf{s}_0$ and $\mathbf{s}_1$, given in Eq. (3.2.3). We calculate the matrices S, F, and F^{-1}:

$$S = [\mathbf{s}_0,\mathbf{s}_1] = \begin{bmatrix} 1&-2\\1&-1\\1&0\\1&1\\1&2 \end{bmatrix}, \quad F = S^TS = \begin{bmatrix} 5&0\\0&10 \end{bmatrix}, \quad F^{-1} = \frac{1}{5}\begin{bmatrix} 1&0\\0&0.5 \end{bmatrix}$$

This gives for G and B:

$$G = SF^{-1} = \frac{1}{5}\begin{bmatrix} 1&-1\\1&-0.5\\1&0\\1&0.5\\1&1 \end{bmatrix}, \quad B = GS^T = \frac{1}{5}\begin{bmatrix} 3&2&1&0&-1\\2&1.5&1&0.5&0\\1&1&1&1&1\\0&0.5&1&1.5&2\\-1&0&1&2&3 \end{bmatrix}$$

Thus, the steady-state LPSM filter $\mathbf{b}_0$ is still equal to the length-5 FIR averager. It is a general property of LPSM filters, that the filter $\mathbf{b}_0$ is the same for successive polynomial orders, that is, for $d = 0,1$, $d = 2,3$, $d = 4,5$, and so on. However, the transient LPSM filters are different. □

Example 3.2.2: Here, we construct the LPSM filters of length $N = 5$ and order $d = 3$. The smoothed estimates are given by the cubic polynomial:

$$\hat{y}_m = c_0 + c_1 m + c_2 m^2 + c_3 m^3$$

There is an additional basis vector $\mathbf{s}_3$ with components $s_3(m) = m^3$. Therefore, the basis matrix S is:

$$S = [\mathbf{s}_0, \mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3] = \begin{bmatrix} 1 & -2 & 4 & -8 \\ 1 & -1 & 1 & -1 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 \end{bmatrix} \Rightarrow F = S^T S = \begin{bmatrix} 5 & 0 & 10 & 0 \\ 0 & 10 & 0 & 34 \\ 10 & 0 & 34 & 0 \\ 0 & 34 & 0 & 130 \end{bmatrix}$$

Because of the checkerboard pattern of this matrix, its inverse can be obtained from the inverses of the two 2×2 interlaced submatrices:

$$\begin{bmatrix} 5 & 10 \\ 10 & 34 \end{bmatrix}^{-1} = \frac{1}{70}\begin{bmatrix} 34 & -10 \\ -10 & 5 \end{bmatrix}, \quad \begin{bmatrix} 10 & 34 \\ 34 & 130 \end{bmatrix}^{-1} = \frac{1}{144}\begin{bmatrix} 130 & -34 \\ -34 & 10 \end{bmatrix}$$

Interlacing these inverses, we obtain:

$$F^{-1} = \begin{bmatrix} 34/70 & 0 & -10/70 & 0 \\ 0 & 130/144 & 0 & -34/144 \\ -10/70 & 0 & 5/70 & 0 \\ 0 & -34/144 & 0 & 10/144 \end{bmatrix}$$

Then, we compute the derivative filter matrix G:

$$G = SF^{-1} = \frac{1}{35}\begin{bmatrix} -3 & 35/12 & 5 & -35/12 \\ 12 & -70/3 & -2.5 & 35/6 \\ 17 & 0 & -5 & 0 \\ 12 & 70/3 & -2.5 & -35/6 \\ -3 & -35/12 & 5 & 35/12 \end{bmatrix}$$

and the LPSM matrix B:

$$B = SG^T = \frac{1}{35}\begin{bmatrix} 34.5 & 2 & -3 & 2 & -0.5 \\ 2 & 27 & 12 & -8 & 2 \\ -3 & 12 & 17 & 12 & -3 \\ 2 & -8 & 12 & 27 & 2 \\ -0.5 & 2 & -3 & 2 & 34.5 \end{bmatrix}$$

As mentioned above, the steady-state LPSM filter $\mathbf{b}_0$ is the same as that of case $d = 2$. But, the transient and differentiation filters are different. □

3.3 Exact Design Equations

In practice, the most common values of d are $0, 1, 2, 3, 4$. For these ds and arbitrary filter lengths N, the LPSM matrix B can be constructed in closed form; see references [36–99],

as well as the extensive tables in [54]. Denoting the inverse of the $(d+1)\times(d+1)$ matrix $F = S^T S$ by $\Phi = F^{-1}$, we can write

$$B = SF^{-1}S^T = S\Phi S^T = \sum_{i=0}^{d}\sum_{j=0}^{d} \mathbf{s}_i \mathbf{s}_j^T \Phi_{ij} \tag{3.3.1}$$

which gives for the mkth matrix element

$$\boxed{B_{mk} = \sum_{i=0}^{d}\sum_{j=0}^{d} s_i(m)s_j(k)\Phi_{ij} = \sum_{i=0}^{d}\sum_{j=0}^{d} m^i k^j \Phi_{ij}}\,, \qquad -M \le m,k \le M \tag{3.3.2}$$

Because of symmetry, $B_{mk} = B_{km}$, these matrix elements represent the kth component of the LPSM filter $\mathbf{b}_m$ or the mth component of the filter $\mathbf{b}_k$, that is,

$$B_{mk} = B_{km} = b_m(k) = b_k(m) = \sum_{i=0}^{d}\sum_{j=0}^{d} m^i k^j \Phi_{ij} \tag{3.3.3}$$

The matrix Φ can be determined easily for the cases $0 \le d \le 4$. The matrix F is a *Hankel matrix*, that is, having the same entries along each antidiagonal line. Therefore, its matrix elements F_{ij} depend only on the sum $i+j$ of the indices. To see this, we write F_{ij} as the inner product:

$$F_{ij} = (S^T S)_{ij} = \mathbf{s}_i^T \mathbf{s}_j = \sum_{m=-M}^{M} s_i(m)s_j(m) = \sum_{m=-M}^{M} m^{i+j}\,, \qquad \text{or,}$$

$$\boxed{F_{ij} = \sum_{m=-M}^{M} m^{i+j} \equiv F_{i+j}}\,, \qquad 0 \le i,j \le d \tag{3.3.4}$$

Note that because of the symmetric limits of summation, F_{i+j} will be zero whenever $i+j$ is odd. This leads to the checkerboard pattern of alternating zeros in F that we saw in the above examples. Also, because $d \le 4$, the only values of $i+j$ that we need are: $i+j = 0, 2, 4, 6, 8$. For those, the summations over m can be done in closed form:

$$\begin{aligned}
F_0 &= \sum_{m=-M}^{M} m^0 = N = 2M+1 \\
F_2 &= \sum_{m=-M}^{M} m^2 = \frac{1}{3}M(M+1)(2M+1) \\
F_4 &= \sum_{m=-M}^{M} m^4 = \frac{1}{5}(3M^2+3M-1)F_2 \\
F_6 &= \sum_{m=-M}^{M} m^6 = \frac{1}{7}(3M^4+6M^3-3M+1)F_2 \\
F_8 &= \sum_{m=-M}^{M} m^8 = \frac{1}{15}(5M^6+15M^5+5M^4-15M^3-M^2+9M-3)F_2
\end{aligned} \tag{3.3.5}$$

We can express F in terms of these definitions for various values of d. For example, for $d = 0, 1, 2, 3$, the F matrices are:

$$[F_0]\,, \quad \begin{bmatrix} F_0 & 0 \\ 0 & F_2 \end{bmatrix}, \quad \begin{bmatrix} F_0 & 0 & F_2 \\ 0 & F_2 & 0 \\ F_2 & 0 & F_4 \end{bmatrix}, \quad \begin{bmatrix} F_0 & 0 & F_2 & 0 \\ 0 & F_2 & 0 & F_4 \\ F_2 & 0 & F_4 & 0 \\ 0 & F_4 & 0 & F_6 \end{bmatrix}$$

The corresponding inverse matrices $\Phi = F^{-1}$ are obtained by interlacing the inverses of the checkerboard submatrices, as in Example 3.2.2. For $d = 0, 1, 2$, we have for Φ:

$$[1/F_0]\,, \quad \begin{bmatrix} 1/F_0 & 0 \\ 0 & 1/F_2 \end{bmatrix}, \quad \begin{bmatrix} F_4/D_4 & 0 & -F_2/D_4 \\ 0 & 1/F_2 & 0 \\ -F_2/D_4 & 0 & F_0/D_4 \end{bmatrix},$$

and for $d = 3$:

$$\Phi = F^{-1} = \begin{bmatrix} F_4/D_4 & 0 & -F_2/D_4 & 0 \\ 0 & F_6/D_8 & 0 & -F_4/D_8 \\ -F_2/D_4 & 0 & F_0/D_4 & 0 \\ 0 & -F_4/D_8 & 0 & F_2/D_8 \end{bmatrix}$$

where the D_4 and D_8 are determinants of the interlaced submatrices:

$$\begin{aligned} D_4 &= F_0F_4 - F_2^2 = \frac{1}{45}M(M+1)(2M+1)(2M+3)(4M^2-1) \\ D_8 &= F_2F_6 - F_4^2 = \frac{3}{35}M(M+2)(M^2-1)D_4 \end{aligned} \tag{3.3.6}$$

Inserting the above expressions for Φ into Eq. (3.3.3), we determine the corresponding LPSM filters. For $d = 0$, we find for $-M \le m, k \le M$:

$$b_m(k) = B_{mk} = \frac{1}{F_0} = \frac{1}{N} \tag{3.3.7}$$

For $d = 1$:

$$b_m(k) = B_{mk} = \frac{1}{F_0} + \frac{mk}{F_2} \tag{3.3.8}$$

For $d = 2$:

$$b_m(k) = B_{mk} = \frac{F_4}{D_4} + \frac{1}{F_2}mk - \frac{F_2}{D_4}(m^2 + k^2) + \frac{F_0}{D_4}m^2k^2 \tag{3.3.9}$$

For $d = 3$:

$$\begin{aligned} b_m(k) = B_{mk} = &\frac{F_4}{D_4} + \frac{F_6}{D_8}mk - \frac{F_2}{D_4}(m^2 + k^2) + \frac{F_0}{D_4}m^2k^2 \\ &- \frac{F_4}{D_8}(km^3 + mk^3) + \frac{F_2}{D_8}m^3k^3 \end{aligned} \tag{3.3.10}$$

The required ratios are given explicitly as follows:

$$\begin{aligned} \frac{F_4}{D_4} &= \frac{3(3M^2 + 3M - 1)}{(2M+3)(4M^2-1)} \\ \frac{F_2}{D_4} &= \frac{15}{(2M+3)(4M^2-1)} \\ \frac{F_0}{D_4} &= \frac{45}{M(M+1)(2M+3)(4M^2-1)} \\ \frac{F_6}{D_8} &= \frac{25(3M^4 + 6M^3 - 3M + 1)}{M(M+2)(M^2-1)(2M+3)(4M^2-1)} \\ \frac{F_4}{D_8} &= \frac{35(3M^2 + 3M - 1)}{M(M+2)(M^2-1)(2M+3)(4M^2-1)} \\ \frac{F_2}{D_8} &= \frac{175}{M(M+2)(M^2-1)(2M+3)(4M^2-1)} \end{aligned} \tag{3.3.11}$$

In a similar fashion, we also find for the case $d = 4$:

$$\begin{aligned} b_m(k) = B_{mk} = &\frac{D_{12}}{D} + \frac{F_6}{D_8}mk - \frac{D_{10}}{D}(m^2 + k^2) + \frac{E_8}{D}m^2k^2 \\ &- \frac{F_4}{D_8}(km^3 + mk^3) + \frac{F_2}{D_8}m^3k^3 + \frac{D_8}{D}(m^4 + k^4) \\ &- \frac{D_6}{D}(m^2k^4 + k^2m^4) + \frac{D_4}{D}m^4k^4 \end{aligned} \tag{3.3.12}$$

where

$$\begin{aligned} D_6 &= F_0F_6 - F_2F_4 & E_8 &= F_0F_8 - F_4^2 \\ D_{10} &= F_2F_8 - F_4F_6 & D_{12} &= F_4F_8 - F_6^2 \\ D &= F_0D_{12} - F_2D_{10} + F_4D_8 \end{aligned} \tag{3.3.13}$$

These are given explicitly as follows:

$$\begin{aligned} D_6 &= \frac{1}{7}(6M^2 + 6M - 5)D_4 \\ D_{10} &= \frac{1}{21}M(M+2)(M^2-1)(2M^2 + 2M - 3)D_4 \\ E_8 &= \frac{1}{5}(4M^4 + 8M^3 - 4M^2 - 8M + 1)D_4 \\ D_{12} &= \frac{1}{735}M(M+2)(M^2-1)(15M^4 + 30M^3 - 35M^2 - 50M + 12)D_4 \\ D &= \frac{4}{11025}M(M+2)(M^2-1)(2M+5)(4M^2-9)(4M^2-1)D_4 \end{aligned} \tag{3.3.14}$$

and the required ratios are:

$$
\begin{aligned}
\frac{D_{12}}{D} &= \frac{15\,(15M^4+30M^3-35M^2-50M+12)}{4\,(2M+5)\,(4M^2-1)\,(4M^2-9)}\\
\frac{D_{10}}{D} &= \frac{525\,(2M^2+2M-3)}{4\,(2M+5)\,(4M^2-1)\,(4M^2-9)}\\
\frac{E_8}{D} &= \frac{2205\,(4M^4+8M^3-4M^2-8M+5)}{4M\,(M+2)\,(M^2-1)\,(2M+5)\,(4M^2-1)\,(4M^2-9)}\\
\frac{D_8}{D} &= \frac{945}{4\,(2M+5)\,(4M^2-1)\,(4M^2-9)}\\
\frac{D_6}{D} &= \frac{1575\,(6M^2+6M-5)}{4M\,(M+2)\,(M^2-1)\,(2M+5)\,(4M^2-1)\,(4M^2-9)}\\
\frac{D_4}{D} &= \frac{11025}{4M\,(M+2)\,(M^2-1)\,(2M+5)\,(4M^2-1)\,(4M^2-9)}
\end{aligned}
\qquad (3.3.15)
$$

In this case, the matrix F and its two interlaced submatrices are:

$$
F = \begin{bmatrix} F_0 & 0 & F_2 & 0 & F_4 \\ 0 & F_2 & 0 & F_4 & 0 \\ F_2 & 0 & F_4 & 0 & F_6 \\ 0 & F_4 & 0 & F_6 & 0 \\ F_4 & 0 & F_6 & 0 & F_8 \end{bmatrix}, \quad \begin{bmatrix} F_0 & F_2 & F_4 \\ F_2 & F_4 & F_6 \\ F_4 & F_6 & F_8 \end{bmatrix}, \quad \begin{bmatrix} F_2 & F_4 \\ F_4 & F_6 \end{bmatrix}
$$

Its inverse—obtained by interlacing the inverses of these two submatrices—can be expressed in terms of the determinant quantities of Eq. (3.3.13):

$$
\Phi = F^{-1} = \begin{bmatrix} D_{12}/D & 0 & -D_{10}/D & 0 & D_8/D \\ 0 & F_6/D_8 & 0 & -F_4/D_8 & 0 \\ -D_{10}/D & 0 & E_8/D & 0 & -D_6/D \\ 0 & -F_4/D_8 & 0 & F_2/D_8 & 0 \\ D_8/D & 0 & -D_6/D & 0 & D_4/D \end{bmatrix}
$$

Eqs. (3.3.5)–(3.3.15) provide closed-form expressions for the LPSM filters $b_m(k)$ of orders $d = 0,1,2,3,4$. Setting $m = 0$, we obtain the explicit forms of the steady-state filters $b_0(k)$, $-M \le k \le M$. For $d = 0,1$:

$$
b_0(k) = \frac{1}{2M+1} \qquad (3.3.16)
$$

for $d = 2,3$:

$$
b_0(k) = \frac{3\,(3M^2+3M-1-5k^2)}{(2M+3)\,(4M^2-1)} \qquad (3.3.17)
$$

and for $d = 4,5$:

$$
b_0(k) = \frac{15\big(15M^4+30M^3-35M^2-50M+12-35\,(2M^2+2M-3)\,k^2+63k^4\big)}{4\,(2M+5)\,(4M^2-1)\,(4M^2-9)} \qquad (3.3.18)
$$

Example 3.3.1: Determine the quadratic/cubic LPSM filters of lengths $N = 5,7,9$. Using (3.3.17) with $M = 2,3,4$, we find (for $-M \le k \le M$):

$$
\begin{aligned}
b_0(k) &= \frac{17-5k^2}{35} = \frac{1}{35}[-3,12,17,12,-3]\\
b_0(k) &= \frac{7-k^2}{21} = \frac{1}{21}[-2,3,6,7,6,3,-2]\\
b_0(k) &= \frac{59-5k^2}{231} = \frac{1}{231}[-21,14,39,54,59,54,39,14,-21]
\end{aligned}
$$

where the coefficients have been reduced to integers as much as possible. □

Example 3.3.2: Determine the quartic and quintic LPSM filters of length $N = 7,9$. Using Eq. (3.3.18) with $M = 3,4$, we find:

$$
\begin{aligned}
b_0(k) &= \frac{131-61.25k^2+5.25k^4}{231} = \frac{1}{231}[5,-30,75,131,75,-30,5]\\
b_0(k) &= \frac{179-46.25k^2+2.25k^4}{429} = \frac{1}{429}[15,-55,30,135,179,135,30,-55,15]
\end{aligned}
$$

3.4 *Geometric Interpretation*

The LPSM filters admit a nice *geometric* interpretation, which is standard in least-squares problems. Let $\mathbb{Y}$ be the vector space of the N-dimensional real-valued vectors $\mathbf{y}$, that is, the space $\mathbb{R}^N$, and let $\mathbb{S}$ be the $(d+1)$-dimensional *subspace* spanned by all linear combinations of the basis vectors $\mathbf{s}_i$, $i = 0,1,\dots,d$.

Thus, the matrix $S = [\mathbf{s}_0, \mathbf{s}_1, \dots, \mathbf{s}_d]$ is a (non-orthogonal) basis of the subspace $\mathbb{S}$. The smoothed vector $\hat{\mathbf{y}}$, being a linear combination of the $\mathbf{s}_i$, belongs to the subspace $\mathbb{S}$. Moreover, because of the orthogonality equations (3.2.8), $\hat{\mathbf{y}}$ is *orthogonal* to the error vector $\mathbf{e}$:

$$
\hat{\mathbf{y}}^T\mathbf{e} = (S\mathbf{c})^T\mathbf{e} = \mathbf{c}^T S^T\mathbf{e} = 0
$$

Then, the equation $\mathbf{e} = \mathbf{y} - \hat{\mathbf{y}}$ can be rewritten as the *orthogonal decomposition*:

$$
\mathbf{y} = \hat{\mathbf{y}} + \mathbf{e} \qquad (3.4.1)
$$

which expresses $\mathbf{y}$ as a sum of a part that belongs to the subspace $\mathbb{S}$ and a part that belongs to the *orthogonal complement* subspace $\mathbb{S}^\perp$. The decomposition is *unique* and represents the *direct sum* decomposition of the full vector space $\mathbb{Y}$:

$$
\mathbb{Y} = \mathbb{S} \oplus \mathbb{S}^\perp
$$

This geometric interpretation requires that the dimension of the subspace $\mathbb{S}$ not exceed the dimension of the full space $\mathbb{Y}$, that is, $d+1 \le N$. The component $\hat{\mathbf{y}}$ that lies in $\mathbb{S}$ is the *projection* of $\mathbf{y}$ onto $\mathbb{S}$. The matrix B in Eq. (3.2.11) is the corresponding *projection matrix*. As such, it will be symmetric, $B^T = B$, and *idempotent*:

$$
B^2 = B \qquad (3.4.2)
$$

The proof is straightforward:

$$B^2 = (SF^{-1}S^T)(SF^{-1}S^T) = SF^{-1}(S^TS)F^{-1}S^T = SF^{-1}S^T = B$$

The matrix $(I-B)$, where I is the N-dimensional identity matrix, is also a projection matrix, projecting onto the orthogonal subspace $\mathbb{S}^\perp$. Thus, the error vector $\mathbf{e}$ belonging to $\mathbb{S}^\perp$ can be obtained from $\mathbf{y}$ by the projection:

$$\mathbf{e} = \mathbf{y} - \hat{\mathbf{y}} = (I-B)\mathbf{y}$$

Because $(I-B)$ is also idempotent and symmetric, $(I-B)^2 = (I-B)$, we obtain for the *minimized* value of the performance index $\mathcal{J}$ of Eq. (3.2.6):

$$\mathcal{J}_{\min} = \mathbf{e}^T\mathbf{e} = \mathbf{y}^T(I-B)^2\mathbf{y} = \mathbf{y}^T(I-B)\mathbf{y} = \mathbf{y}^T\mathbf{y} - \mathbf{y}^TB\mathbf{y} \tag{3.4.3}$$

3.5 Orthogonal Polynomial Bases

Computationally, the non-orthogonal basis $S = [\mathbf{s}_0, \mathbf{s}_1, \dots, \mathbf{s}_d]$ is not the most convenient one. The Gram-Schmidt orthogonalization process may be applied to the columns of S to obtain an orthogonal basis. This procedure amounts to performing the QR-factorization[†] on S, that is,

$$S = QR \tag{3.5.1}$$

where Q is an $N\times(d+1)$ matrix with orthonormal columns, that is, $Q^TQ = I$, and R is a $(d+1)\times(d+1)$ non-singular *upper-triangular* matrix.

The columns of $Q = [\mathbf{q}_0, \mathbf{q}_1, \dots, \mathbf{q}_d]$, correspond to the (orthonormalized) discrete Chebyshev or Gram polynomials $q_i(n)$, $i = 0, 1, \dots, d$, constructed from the monomial basis $s_i(n) = n^i$ by the Gram-Schmidt process. Noting that $S^TS = R^T(Q^TQ)R = R^TR$, the design of the filter matrices B, G can be formulated more efficiently as follows:

$$\begin{aligned} F &= S^TS = R^TR \\ G &= SF^{-1} = QR^{-T} \\ B &= SF^{-1}S^T = QQ^T \end{aligned} \tag{3.5.2}$$

which lead to the explicit construction of the differentiation and LPSM filters in terms of the Chebyshev polynomials $q_i(n)$:

$$\begin{aligned} \mathbf{g}_i &= \sum_{j=0}^{i}\mathbf{q}_j\,(R^{-1})_{ij} \quad\Rightarrow\quad g_i(n) = \sum_{j=0}^{i} q_j(n)\,(R^{-1})_{ij} \\ B &= \sum_{i=0}^{d}\mathbf{q}_i\mathbf{q}_i^T \quad\Rightarrow\quad b_m(k) = B_{km} = \sum_{i=0}^{d} q_i(k)q_i(m) \end{aligned} \tag{3.5.3}$$

The expression for $b_m(k)$ can be simplified further using the Christoffel-Darboux identity for orthogonal polynomials. We discuss these matters further in Sec. 4.3. The MATLAB function `lpsm` implements (3.5.2). Its inputs are N, d and its outputs B, G:

```
[B,G] = lpsm(N,d);        % local polynomial smoothing and differentiation filter design
```

The function constructs the basis matrix S with the help of the function `lpbasis` and carries out its QR-factorization with the help of the built-in function `qr`. The following code fragment illustrates the computational steps:

```
S = lpbasis(N,d);        % construct polynomial basis
[Q,R] = qr(S, 0);        % economy form, R is (d+1)x(d+1) upper triangular
G = Q/R';                % differentiation filters
B = Q*Q';                % smoothing filters
```

[†] see Sec. 15.20.

3.6 Polynomial Predictive and Interpolation Filters

The case $d+1 = N$ or $d = N-1$ is of special interest, corresponding to ordinary polynomial *Lagrange interpolation*. Indeed, in this case, the basis matrix S becomes a square non-singular $N\times N$ matrix with an ordinary inverse S^{-1}, which implies that B becomes the identity matrix,

$$B = S(S^TS)^{-1}S^T = S(S^{-1}S^{-T})S^T = I$$

or, equivalently, the subspace $\mathbb{S}$ becomes the full space $\mathbb{Y}$. The optimal polynomial of degree $d = N-1$ fits through all the sample points of the N-dimensional vector $\mathbf{y}$, that is, $\mathbf{e} = 0$ or $\hat{\mathbf{y}} = \mathbf{y} = S\mathbf{c}$, with solution $\mathbf{c} = S^{-1}\mathbf{y}$, and interpolates between those samples. This polynomial is defined for any independent variable t by:

$$\hat{y}_t = \sum_{i=0}^{N-1} c_i t^i = \mathbf{c}^T\mathbf{u}_t = \mathbf{y}^TS^{-T}\mathbf{u}_t \equiv \mathbf{y}^T\mathbf{b}_t = \sum_{k=-M}^{M} b_t(k)y_k \tag{3.6.1}$$

where we set,

$$\mathbf{u}_t = \begin{bmatrix} 1 \\ t \\ \vdots \\ t^{N-1} \end{bmatrix}, \quad \mathbf{b}_t = S^{-T}\mathbf{u}_t \quad\Rightarrow\quad b_t(k) = \sum_{i=0}^{N-1}(S^{-1})_{ik}\,t^i \tag{3.6.2}$$

The polynomials $b_t(k)$ of degree $(N-1)$ in t are the ordinary Lagrange interpolation polynomials, interpolating through the points y_k. To see this, we note that at each discrete value of t, say $t = m$ with $-M \le m \le M$, we have:

$$b_m(k) = \sum_{i=0}^{N-1}(S^{-1})_{ik}\,m^i = \sum_{i=0}^{N-1}(S^{-1})_{ik}S_{mi} = (SS^{-1})_{mk} = I_{mk} = \delta(m-k) \tag{3.6.3}$$

so that the polynomial passes through the signal values at the sampling instants:

$$\hat{y}_t\big|_{t=m} = \sum_{k=-M}^{M} b_m(k)y_k = \sum_{k=-M}^{M}\delta(m-k)y_k = y_m$$

It is straightforward to show using the property (3.6.3) that $b_t(k)$ is given by the usual Lagrange interpolation formula:

$$b_t(k) = \prod_{\substack{m=-M \\ m\neq k}}^{M} \left(\frac{t-m}{k-m}\right), \qquad -M \le k \le M \tag{3.6.4}$$

Indeed, Eq. (3.6.4) states that the $(2M)$ roots of $b_t(k)$ are the points $t = m$, for $-M \le m \le M$ and $m \neq k$, which fixes the polynomial up to a constant. That constant is determined by the condition $b_k(k) = 1$.

Example 3.6.1: For $N = 5$ and $d = N - 1 = 4$, the fourth degree Lagrange polynomials, constructed from Eq. (3.6.4), can be expanded in powers of t:

$$\begin{bmatrix} b_t(-2) \\ b_t(-1) \\ b_t(0) \\ b_t(1) \\ b_t(2) \end{bmatrix} = \frac{1}{24}\begin{bmatrix} 0 & 2 & -1 & -2 & 1 \\ 0 & -16 & 16 & 4 & -4 \\ 24 & 0 & -30 & 0 & 6 \\ 0 & 16 & 16 & -4 & -4 \\ 0 & -2 & -1 & 2 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ t^1 \\ t^2 \\ t^3 \\ t^4 \end{bmatrix}$$

The coefficient matrix is recognized as the inverse transposed of the basis matrix S:

$$S = \begin{bmatrix} 1 & -2 & 4 & -8 & 16 \\ 1 & -1 & 1 & -1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 & 16 \end{bmatrix} \quad \Rightarrow \quad S^{-T} = \frac{1}{24}\begin{bmatrix} 0 & 2 & -1 & -2 & 1 \\ 0 & -16 & 16 & 4 & -4 \\ 24 & 0 & -30 & 0 & 6 \\ 0 & 16 & 16 & -4 & -4 \\ 0 & -2 & -1 & 2 & 1 \end{bmatrix}$$

which verifies Eq. (3.6.2). □

We note that $b_t(k)$ can be written in the following analytical form, which shows the relation of the Lagrange interpolation filter to the ideal sinc-interpolation filter:

$$b_t(k) = \frac{\Gamma(M+1+t)\Gamma(M+1-t)}{\Gamma(M+1+k)\Gamma(M+1-k)} \cdot \frac{\sin\big(\pi(t-k)\big)}{\pi(t-k)} \tag{3.6.5}$$

Some alternative expressions are as follows:

$$b_t(k) = (-1)^{M+k} \sum_{m=M+k}^{2M} \binom{M+t}{m}\binom{m}{M+k}(-1)^m \tag{3.6.6}$$

$$b_t(k) = \frac{(-1)^{M+1-k}\Gamma(M+1-t)}{(t-k)\Gamma(-M-t)\Gamma(M+1+k)\Gamma(M+1-k)} \tag{3.6.7}$$

and since the $b_t(k)$ sum up to one, we also have [156]:

$$b_t(k) = \left[\sum_{n=-M}^{M} \frac{b_t(n)}{b_t(k)}\right]^{-1} = \left[\sum_{n=-M}^{M} (-1)^{k-n}\,\frac{(M+k)!\,(M-k)!}{(M+n)!\,(M-n)!}\,\frac{t-k}{t-n}\right]^{-1} \tag{3.6.8}$$

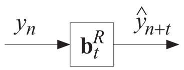
For polynomial orders $d < N-1$, one can still interpolate approximately and smoothly between the samples y_m. In this case, using $\mathbf{c} = G^T\mathbf{y} = (S^TS)^{-1}S^T\mathbf{y}$, we have:

$$\hat{y}_t = \sum_{i=0}^{d} c_i t^i = \mathbf{c}^T\mathbf{u}_t = \mathbf{y}^T G\mathbf{u}_t \equiv \mathbf{y}^T\mathbf{b}_t = \sum_{k=-M}^{M} b_t(k) y_k \tag{3.6.9}$$

where now

$$\mathbf{u}_t = \begin{bmatrix} 1 \\ t^1 \\ t^2 \\ \vdots \\ t^d \end{bmatrix}, \quad \mathbf{b}_t = G\mathbf{u}_t = S(S^TS)^{-1}\mathbf{u}_t \quad \Rightarrow \quad b_t(k) = \sum_{i=0}^{d} G_{ki}\, t^i \tag{3.6.10}$$

and shifting the origin $k = 0$ to the arbitrary time instant n, we obtain the interpolation formula for a shift t relative to the time instant n:



$$\boxed{\hat{y}_{n+t} = \sum_{k=-M}^{M} b_t(k) y_{n+k} = \sum_{k=-M}^{M} b_t^R(k) y_{n-k}} \tag{3.6.11}$$

where $b_t^R(k) = b_t(-k)$. Such formulas can also be used for prediction by choosing $t > M$ so that $n + t > n + M$, that is, it lies beyond the end of the filter range.

We can obtain closed-form expressions for the interpolation filters $b_t(k)$ for $d = 0, 1, 2, 3, 4$ and arbitrary M, by replacing in Eqs. (3.3.7)–(3.3.12) the variable m in $b_m(k)$ by the variable t. For example, for $d = 1, 2, 3, 4$, we have, respectively:

$$\begin{aligned} b_t(k) &= \frac{1}{F_0} + \frac{tk}{F_2} \\ b_t(k) &= \frac{F_4}{D_4} + \frac{1}{F_2}tk - \frac{F_2}{D_4}(t^2+k^2) + \frac{F_0}{D_4}t^2k^2 \\ b_t(k) &= \frac{F_4}{D_4} + \frac{F_6}{D_8}tk - \frac{F_2}{D_4}(t^2+k^2) + \frac{F_0}{D_4}t^2k^2 - \frac{F_4}{D_8}(kt^3+tk^3) + \frac{F_2}{D_8}t^3k^3 \\ b_t(k) &= \frac{D_{12}}{D} + \frac{F_6}{D_8}tk - \frac{D_{10}}{D}(t^2+k^2) + \frac{E_8}{D}t^2k^2 - \frac{F_4}{D_8}(kt^3+tk^3) \\ &\quad + \frac{F_2}{D_8}t^3k^3 + \frac{D_8}{D}(t^4+k^4) - \frac{D_6}{D}(t^2k^4+k^2t^4) + \frac{D_4}{D}t^4k^4 \end{aligned} \tag{3.6.12}$$

where the required coefficient ratios are given by Eqs. (3.3.11) and (3.3.15). The interpolation filter (3.6.10) may be written in terms of the columns of $G = [\mathbf{g}_0, \mathbf{g}_1, \dots, \mathbf{g}_d]$:

$$b_t(k) = \sum_{i=0}^{d} G_{ki}t^i = \sum_{i=0}^{d} g_i(k)\,t^i \quad \Rightarrow \quad \mathbf{b}_t = \sum_{i=0}^{d} \mathbf{g}_i t^i \tag{3.6.13}$$

This representation admits a convenient realization, known as a *Farrow structure*, which allows the changing of the parameter t on the fly without having to redesign the filter. It is essentially a block-diagram realization of Eq. (3.6.13) written in nested form using Hörner's rule. For example, if $d = 3$, we have

$$\mathbf{b}_t = \mathbf{g}_0 + \mathbf{g}_1 t + \mathbf{g}_2 t^2 + \mathbf{g}_3 t^3 = \big((\mathbf{g}_3 t + \mathbf{g}_2)t + \mathbf{g}_1\big)t + \mathbf{g}_0 \tag{3.6.14}$$

Fig. 3.6.1 shows this realization where we replaced $\mathbf{g}_i$ by their reversed versions $\mathbf{g}_i^R$, which appear in the convolutional filtering equations. The parameter t appears only in the lower multipliers and can be independently controlled.

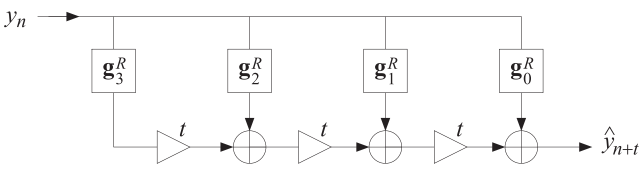


Fig. 3.6.1 Farrow structure for interpolating or predictive FIR filter.

The filtering equation (3.6.11) can also be written in a causal manner by setting $t = M + \tau$ and defining the causal filter, where $N = 2M + 1$:

$$\boxed{h_\tau(k) = b_{M+\tau}(M-k)}\,, \quad k = 0, 1, \dots, N-1 \tag{3.6.15}$$

Replacing $n \to n-M$ and $k \to k-M$, Eq. (3.6.11) is transformed into a *causal* filtering operation that predicts the future sample $y_{n+\tau}$ from the present and past samples y_{n-k}, $k = 0, 1, \dots, N-1$. The mapping of the time indices is explained in Fig. 3.6.2. The resulting filtering operation reads:

$$\boxed{\hat{y}_{n+\tau} = \sum_{k=0}^{N-1} h_\tau(k) y_{n-k}}\,, \quad \tau \geq 0 \tag{3.6.16}$$

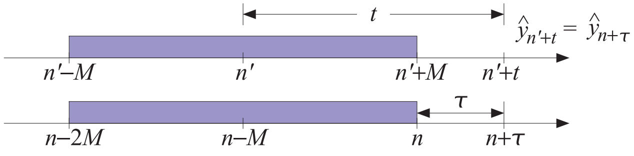


Fig. 3.6.2 Double-sided and causal predictive FIR filters, with $n' = n - M$ and $t = M + \tau$.

Since τ is any real number, the notation $n + \tau$ corresponds to the actual time instant $(n+\tau)T$ in seconds, where T is the sampling time interval. The filter $h_{-\tau}(k)$ may also be used for implementing a *fractional delay* as opposed to prediction, that is,

$$\boxed{\hat{y}_{n-\tau} = \sum_{k=0}^{N-1} h_{-\tau}(k) y_{n-k}} \quad \text{(fractional delay)} \tag{3.6.17}$$

The filters $b_t(k)$ and $h_\tau(k)$ satisfy the following polynomial-preserving moment constraints (being equivalent to $S^T \mathbf{b}_t = \mathbf{u}_t$), where $i = 0, 1, \dots, d$:

$$\sum_{k=-M}^{M} k^i b_t(k) = t^i \quad \Rightarrow \quad \sum_{k=0}^{N-1} k^i h_\tau(k) = (-\tau)^i\,, \quad \sum_{k=0}^{N-1} k^i h_{-\tau}(k) = \tau^i \tag{3.6.18}$$

These constraints imply that Eqs. (3.6.16) and (3.6.17) are exact for polynomials of degree $r \leq d$. For any such polynomial $P(n)$, we have:

$$\sum_{k=0}^{N-1} h_{-\tau}(k) P(n-k) = P(n-\tau) \tag{3.6.19}$$

For example, we have for the monomial $P(n) = n^r$ with $r \leq d$:

$$\begin{aligned}
\sum_{k=0}^{N-1} h_{-\tau}(k)\,(n-k)^r &= \sum_{k=0}^{N-1} h_{-\tau}(k) \sum_{i=0}^{r} \binom{r}{i} n^{r-i} (-1)^i k^i \\
&= \sum_{i=0}^{r} \binom{r}{i} n^{r-i} (-1)^i \sum_{k=0}^{N-1} k^i h_{-\tau}(k) = \sum_{i=0}^{r} \binom{r}{i} n^{r-i} (-1)^i \tau^i = (n-\tau)^r
\end{aligned}$$

It is in the sense of Eq. (3.6.19) that we may think of the transfer function of the filter $h_{-\tau}(k)$ as approximating the ideal fractional delay $z^{-\tau}$:

$$\sum_{k=0}^{N-1} h_{-\tau}(k) z^{-k} \simeq z^{-\tau} \tag{3.6.20}$$

Further insight into the nature of the approximation (3.6.20) can be gained by considering the Lagrange interpolation case, $d = N - 1$. From the definition of $h_{-\tau}(k) = b_{M-\tau}(M-k)$ and Eqs. (3.6.4) and (3.6.6), we obtain, for $k = 0, 1, \dots, N-1$:

$$h_{-\tau}(k) = \prod_{\substack{i=0 \\ i\neq k}}^{N-1} \left(\frac{\tau - i}{k - i}\right) = \sum_{i=N-1-k}^{N-1} \binom{N-1-\tau}{i} \binom{i}{N-1-k} (-1)^{i-(N-1-k)} \tag{3.6.21}$$

The z-transform of $h_{-\tau}(k)$ is then,

$$\begin{aligned}
\sum_{k=0}^{N-1} h_{-\tau}(k) z^{-k} &= \sum_{k=0}^{N-1} \sum_{i=N-1-k}^{N-1} \binom{N-1-\tau}{i} \binom{i}{N-1-k} (-1)^{i-(N-1-k)} z^{-k} \\
&= z^{-(N-1)} \sum_{i=0}^{N-1} \sum_{k=N-1-i}^{N-1} \binom{N-1-\tau}{i} \binom{i}{N-1-k} (-1)^{i-(N-1-k)} z^{N-1-k}
\end{aligned}$$

Changing summation variables and using the binomial expansion of $(z-1)^i$, we obtain,

$$\begin{aligned}
\sum_{k=0}^{N-1} h_{-\tau}(k) z^{-k} &= z^{-(N-1)} \sum_{i=0}^{N-1} \sum_{j=0}^{m} \binom{N-1-\tau}{i} \binom{i}{j} (-1)^{i-j} z^j \\
&= z^{-(N-1)} \sum_{i=0}^{N-1} \binom{N-1-\tau}{i} (z-1)^i
\end{aligned} \tag{3.6.22}$$

Applying the binomial identity,

$$(1+x)^{\alpha}=\sum_{m=0}^{\infty}\binom{\alpha}{i}x^{i} \tag{3.6.23}$$

with $x = z-1$ and $\alpha = N-1-\tau$, we have,

$$z^{N-1-\tau} = (1+z-1)^{N-1-\tau}=\sum_{i=0}^{\infty}\binom{N-1-\tau}{i}(z-1)^{i} \tag{3.6.24}$$

We recognize the sum in Eq. (3.6.22) to be the first N terms of (3.6.24). Thus, taking that sum to approximately represent $z^{N-1-\tau}$, we have,

$$\sum_{k=0}^{N-1}h_{-\tau}(k)z^{-k}\simeq z^{-(N-1)}\,z^{N-1-\tau}=z^{-\tau} \tag{3.6.25}$$

This approximation becomes exact whenever τ is an integer, say $\tau = m$, with $m = 0,1,\dots,N-1$. Indeed in this case, the summation range $0\le i\le N-1$ in Eq. (3.6.22) can be restricted to $0\le i\le N-1-m$ because the binomial coefficient vanishes whenever its (integer) arguments satisfy $N-1-m<i\le N-1$. We then have an ordinary binomial expansion for an integer power:

$$\sum_{k=0}^{N-1}h_{-m}(k)z^{-k}=z^{-(N-1)}\sum_{i=0}^{N-1-m}\binom{N-1-m}{i}(z-1)^{i}=z^{-(N-1)}\,(1+z-1)^{N-1-m}=z^{-m}$$

which implies the expected result $h_{-m}(k)=\delta(k-m)$. Eq. (3.6.22) is equivalent to *Newton's forward interpolation* formula. To see this, let us introduce the forward difference operator $\Delta = z-1$, or, $\Delta f_n = f_{n+1}-f_n$, and apply (3.6.22) in the time domain:

$$\hat{y}_{n-\tau}=\sum_{k=0}^{N-1}h_{-\tau}(k)y_{n-k}=\sum_{i=0}^{N-1}\binom{N-1-\tau}{i}\Delta^{i}y_{n-(N-1)} \tag{3.6.26}$$

This interpolates between the points $[y_{n-(N-1)},\dots,y_{n-1},y_n]$ with τ measured backwards from the end-point y_n. We may measure the interpolation distance forward from the first point $y_{n-(N-1)}$ by defining $x = N-1-\tau$. Then, Eq. (3.6.26) reads,

$$\hat{y}_{n-(N-1)+x}=\sum_{i=0}^{N-1}\binom{x}{i}\Delta^{i}y_{n-(N-1)} \tag{3.6.27}$$

and setting $n = N-1$ so that the data range is $[y_0,y_1,\dots,y_{N-1}]$, we obtain the usual way of writing Newton's polynomial interpolation formula:

$$\hat{y}_{x}=\sum_{i=0}^{N-1}\binom{x}{i}\Delta^{i}y_{0}=\sum_{i=0}^{N-1}\frac{x(x-1)\cdots(x-i+1)}{i!}\,\Delta^{i}y_{0} \tag{3.6.28}$$

We note also that Eq. (3.6.21) is valid for either even or odd values of N. For $N = 2,3,4$, we obtain for the corresponding filter coefficients:

$$\begin{bmatrix}h_{-\tau}(0)\\ h_{-\tau}(1)\end{bmatrix}=\begin{bmatrix}1-\tau\\ \tau\end{bmatrix},\quad \begin{bmatrix}h_{-\tau}(0)\\ h_{-\tau}(1)\\ h_{-\tau}(2)\end{bmatrix}=\frac{1}{2}\begin{bmatrix}(\tau-1)(\tau-2)\\ -2\tau(\tau-2)\\ \tau(\tau-1)\end{bmatrix}$$

$$\begin{bmatrix}h_{-\tau}(0)\\ h_{-\tau}(1)\\ h_{-\tau}(2)\\ h_{-\tau}(3)\end{bmatrix}=\frac{1}{6}\begin{bmatrix}-(\tau-1)(\tau-2)(\tau-3)\\ 3\tau(\tau-2)(\tau-3)\\ -3\tau(\tau-1)(\tau-3)\\ \tau(\tau-1)(\tau-2)\end{bmatrix} \tag{3.6.29}$$

and the corresponding interpolation formulas:

$$\begin{aligned}\hat{y}_{n-\tau}&=(1-\tau)y_n+\tau y_{n-1}\\ \hat{y}_{n-\tau}&=\frac{1}{2}(\tau-1)(\tau-2)y_n-\tau(\tau-2)y_{n-1}+\frac{1}{2}\tau(\tau-1)y_{n-2}\\ \hat{y}_{n-\tau}&=-\frac{1}{6}(\tau-1)(\tau-2)(\tau-3)y_n+\frac{1}{2}\tau(\tau-2)(\tau-3)y_{n-1}\\ &\quad-\frac{1}{2}\tau(\tau-1)(\tau-3)y_{n-2}+\frac{1}{6}\tau(\tau-1)(\tau-2)y_{n-3}\end{aligned} \tag{3.6.30}$$

Example 3.6.2: Fig. 3.6.3 shows in the top row an example of a Lagrange fractional-delay filter with $N = 3$ and polynomial order $d = N-1 = 2$ for the delay values $\tau = m/10$, $m = 1,2,\dots,10$.

The bottom row is the case $N = 5$ and $d = N-1 = 4$ with delays τ extending over the interval $0\le\tau\le 2$. This filter interpolates between the samples $[y_{n-4},y_{n-3},y_{n-2},y_{n-1},y_n]$. The chosen range of τ's spans the gaps between $[y_{n-2},y_{n-1},y_n]$. For the subrange $0\le\tau\le 1$ which spans $[y_{n-1},y_n]$, the magnitude response is greater than one, while it is less than one for the more central range $1\le\tau\le 2$ which spans $[y_{n-2},y_{n-1}]$. The following MATLAB code segment illustrates the generation of the upper two graphs:

```
f = linspace(0,1,1001); w= pi*f;                % frequencies 0 ≤ ω ≤ π
N=3; d=N-1; M = floor(N/2);                     % d = N−1 for Lagrange interpolation

Hmag = []; Hdel = [];
for m=1:10,
   tau = m/10;                                  % desired delays
   h = flip(lpinterp(N,d,M-tau));               % lpinterp is discussed in Sec. 3.8
   H = freqz(h,1,w);
   Hmag = [Hmag; 10*log10(abs(H))];             % magnitude responses in dB
   Delay = -angle(H)./w; Delay(1) = tau;
   Hdel = [Hdel; Delay];                        % phase delays
end

figure; plot(f,Hmag); figure; plot(f,Hdel);
```

The filters were calculated with the function `lpinterp` (from Sec. 3.8) with arguments $d = N-1$, $t = M-\tau$, with reversed output to account for the definition $h_{-\tau}(k)=b_{M-\tau}(M-k)$.

In both cases, we observe that the useful bandwidth of operation, within which both the phase delays have the correct values and the magnitude response is near unity, is fairly narrow extending to about $\omega = 0.2\pi$, or $f = f_s/10$ in units of the sampling rate f_s. □

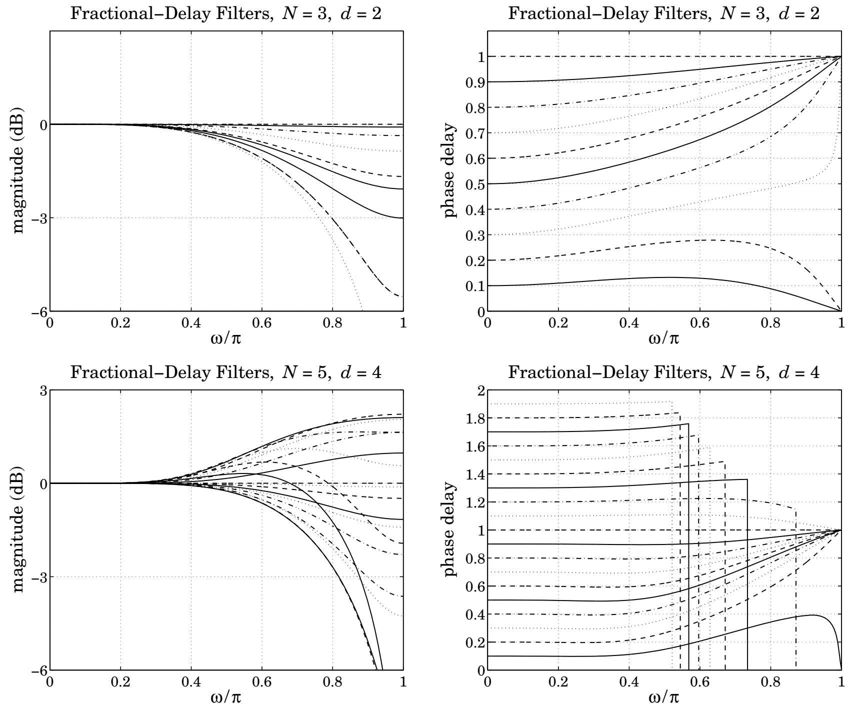


Fig. 3.6.3 Lagrange fractional-delay filters with $N = 3$.

References [152–173] contain further information on predictive FIR and fractional-delay filters. See also [174–187] for alternative implementations of fractional delay using maximally-flat and allpass filters. Ref. [162] provides a nice review of various approaches to the fractional-delay problem.

3.7 Minimum Variance Filters

Next we discuss the *equivalence* of the least-square polynomial fitting approach to the minimization of the NRR subject to linear moment constraints. In the actuarial context, such designs are referred to as "minimum $\mathcal{R}_0$" or "minimum variance" filters, as opposed to the "minimum $\mathcal{R}_s$" or "minimum roughness" filters— the nomenclature being explained in Sec. 4.2.

The projection properties of B may be used to calculate the NRR. For example, the property mentioned previously that the NRR of the filter $\mathbf{b}_0$ is the equal to the middle value $b_0(0)$ follows from Eq. (3.4.2). Using the symmetry of B, we have

$$B^T = B = B^2 = B^T B$$

Taking matrix elements, we have $B_{km} = (B^T)_{mk} = (B^T B)_{mk}$. But, B_{km} is the kth component of the mth column $\mathbf{b}_m$. Using a similar argument as in Eq. (3.2.13), we also have $(B^T B)_{mk} = \mathbf{b}_m^T \mathbf{b}_k$. Therefore,

$$\mathbf{b}_m^T \mathbf{b}_k = b_m(k)$$

For $k = m$, we have the diagonal elements of $B^T B = B$:

$$\mathcal{R} = \mathbf{b}_m^T \mathbf{b}_m = b_m(m) \tag{3.7.1}$$

These are recognized as the NRRs of the filters $\mathbf{b}_m$. In particular, for $m = 0$, we have $\mathcal{R} = \mathbf{b}_0^T \mathbf{b}_0 = b_0(0)$. Setting $k = 0$ in Eqs. (3.3.16)–(3.3.18), we find that the NRRs of the cases $d = 0, 1$, $d = 2, 3$, and $d = 4, 5$ are given by the coefficient ratios $1/F_0$, F_4/D_4, and D_{12}/D. Therefore:

$$\begin{aligned}
&(d = 0, 1) \qquad \mathcal{R} = \frac{1}{N} \\
&(d = 2, 3) \qquad \mathcal{R} = \frac{3(3M^2 + 3M - 1)}{(2M + 3)(4M^2 - 1)} \\
&(d = 4, 5) \qquad \mathcal{R} = \frac{15(15M^4 + 30M^3 - 35M^2 - 50M + 12)}{4(2M + 5)(4M^2 - 1)(4M^2 - 9)}
\end{aligned} \tag{3.7.2}$$

In the limit of large N or M, we have the approximate asymptotic expressions:

$$\begin{aligned}
&(d = 0, 1) \qquad \mathcal{R} = \frac{1}{N} \\
&(d = 2, 3) \qquad \mathcal{R} \simeq \frac{9/4}{N} = \frac{2.25}{N} \\
&(d = 4, 5) \qquad \mathcal{R} \simeq \frac{225/64}{N} = \frac{3.52}{N}
\end{aligned} \tag{3.7.3}$$

Thus, the noise reductions achieved by the quadratic/cubic and quartic/quintic cases are 2.25 and 3.52 times worse than that of the plain FIR averager of the same length N. Another consequence of the projection nature of B is:

$$B^T S = S, \qquad S^T B = S^T \tag{3.7.4}$$

Indeed, $B^T S = BS = S(S^T S)^{-1} S^T S = S$. Column-wise the first equation states that:

$$B^T[\mathbf{s}_0, \mathbf{s}_1, \dots, \mathbf{s}_d] = [\mathbf{s}_0, \mathbf{s}_1, \dots, \mathbf{s}_d] \quad \Rightarrow \quad B^T \mathbf{s}_i = \mathbf{s}_i, \quad i = 0, 1, \dots, d$$

Thus, the basis vectors $\mathbf{s}_i$ remain *invariant* under projection, but that is to be expected because they already lie in $\mathbb{S}$. In fact, any other linear combination of them, such as Eq. (3.2.30), remains invariant under B, that is, $B^T \hat{\mathbf{y}} = \hat{\mathbf{y}}$.

This property answers the question: When are the smoothed values equal to the original ones, $\hat{\mathbf{y}} = \mathbf{y}$, or, equivalently, when is the error zero, $\mathbf{e} = 0$? Because $\mathbf{e} = \mathbf{y} - B^T \mathbf{y}$, the error will be zero if and only if $B^T \mathbf{y} = \mathbf{y}$, which means that $\mathbf{y}$ already *lies* in $\mathbb{S}$, that is, it is a linear combination of $\mathbf{s}_i$. This implies that the samples y_m are already dth order polynomial functions of m, as in Eq. (3.2.27).

The second equation in (3.7.4) implies certain *constraints* on the filters $\mathbf{b}_m$, which can be used to develop an alternative approach to the LPSM filter design problem in terms of minimizing the NRR subject to constraints. To see this, we write the $(d+1)\times N$ transposed matrix S^T column-wise:

$$S^T = [\mathbf{u}_{-M},\dots,\mathbf{u}_{-1},\mathbf{u}_0,\mathbf{u}_1,\dots,\mathbf{u}_M] \tag{3.7.5}$$

For example, in the $N = 5$, $d = 2$ case, we have:

$$S^T = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ -2 & -1 & 0 & 1 & 2 \\ 4 & 1 & 0 & 1 & 4 \end{bmatrix} \equiv [\mathbf{u}_{-2},\, \mathbf{u}_{-1},\, \mathbf{u}_0,\, \mathbf{u}_1,\, \mathbf{u}_2]$$

It is easily verified that the mth column $\mathbf{u}_m$ is simply

$$\mathbf{u}_m = \begin{bmatrix} 1 \\ m \\ m^2 \\ \vdots \\ m^d \end{bmatrix}, \qquad -M \le m \le M \tag{3.7.6}$$

which is the same as $\mathbf{u}_t$ at $t = m$, in terms of the definition (3.6.10). Using $B = GS^T$, we can express the LPSM filters $\mathbf{b}_m$ in terms of $\mathbf{u}_m$, as follows:

$$[\mathbf{b}_{-M},\dots,\mathbf{b}_{-1},\mathbf{b}_0,\mathbf{b}_1,\dots,\mathbf{b}_M] = B = GS^T = G[\mathbf{u}_{-M},\dots,\mathbf{u}_{-1},\mathbf{u}_0,\mathbf{u}_1,\dots,\mathbf{u}_M]$$

which implies:

$$\mathbf{b}_m = G\mathbf{u}_m = SF^{-1}\mathbf{u}_m \tag{3.7.7}$$

Multiplying by S^T, we find $S^T\mathbf{b}_m = S^T SF^{-1}\mathbf{u}_m = \mathbf{u}_m$, or,

$$S^T\mathbf{b}_m = \mathbf{u}_m \qquad \Rightarrow \qquad \begin{bmatrix} \mathbf{s}_0^T\mathbf{b}_m \\ \mathbf{s}_1^T\mathbf{b}_m \\ \vdots \\ \mathbf{s}_d^T\mathbf{b}_m \end{bmatrix} = \begin{bmatrix} 1 \\ m \\ \vdots \\ m^d \end{bmatrix} \tag{3.7.8}$$

These relationships are the column-wise equivalent of $S^TB = S^T$. Thus, each LPSM filter $\mathbf{b}_m$ satisfies $(d+1)$ linear constraints:

$$\mathbf{s}_i^T\mathbf{b}_m = m^i, \qquad i = 0,1,\dots,d \tag{3.7.9}$$

Writing the dot products explicitly, we have equivalently:

$$\boxed{\sum_{n=-M}^{M} n^i b_m(n) = m^i}\,, \qquad i = 0,1,\dots,d \tag{3.7.10}$$

In particular, for the steady-state LPSM filter $\mathbf{b}_0$, we have $\mathbf{u}_0 = [1,0,0,\dots,0]^T$, with ith component $\delta(i)$. Therefore, the constraint $S^T\mathbf{b}_0 = \mathbf{u}_0$ reads component-wise:

$$\boxed{\sum_{n=-M}^{M} n^i b_0(n) = \delta(i), \quad i = 0,1,\dots,d} \tag{3.7.11}$$

For $i = 0$, this is the usual DC constraint:

$$\sum_{n=-M}^{M} b_0(n) = 1 \tag{3.7.12}$$

and for $i = 1,2,\dots,d$:

$$\sum_{n=-M}^{M} n^i b_0(n) = 0 \tag{3.7.13}$$

The quantity in the left-hand side of Eq. (3.7.11) is called the ith *moment* of the impulse response $b_0(n)$. Because of the symmetric limits of summation over n and the symmetry of $b_0(n)$ about its middle, the moments (3.7.13) will be zero for odd i, and therefore are not extra constraints. However, for even i, they are nontrivial constraints.

These moments are related to the *derivatives* of the frequency response at $\omega = 0$. Indeed, defining,

$$B_0(\omega) = \sum_{n=-M}^{M} b_0(n)e^{-j\omega n}$$

and differentiating it i times, we have:

$$j^i B_0^{(i)}(\omega) = j^i \frac{d^i B_0(\omega)}{d\omega^i} = \sum_{n=-M}^{M} n^i b_0(n)e^{-j\omega n}$$

Setting $\omega = 0$, we obtain:

$$j^i B_0^{(i)}(0) = j^i \frac{d^i B_0(\omega)}{d\omega^i}\bigg|_{\omega=0} = \sum_{n=-M}^{M} n^i b_0(n) \tag{3.7.14}$$

Thus, the moment constraints (3.7.12) and (3.7.13) are equivalent to the DC constraint and the *flatness* constraints on the frequency response at $\omega = 0$:

$$\boxed{B_0(0) = 1, \qquad B_0^{(i)}(0) = 0, \quad i = 1,2,\dots,d} \tag{3.7.15}$$

The larger the d, the more derivatives vanish at $\omega = 0$, and the flatter the response $B_0(\omega)$ becomes. This effectively increases the cutoff frequency of the lowpass filter—letting through more noise, but at the same time preserving more of the higher frequencies in the desired signal.

Figure 3.7.1 shows the magnitude response $|B_0(\omega)|$ for the cases $N = 7,15$ and $d = 0,2,4$. The quadratic filters are flatter at DC than the plain FIR averager because of the extra constraint $B_0''(0) = 0$. Similarly, the quartic filters are even flatter because

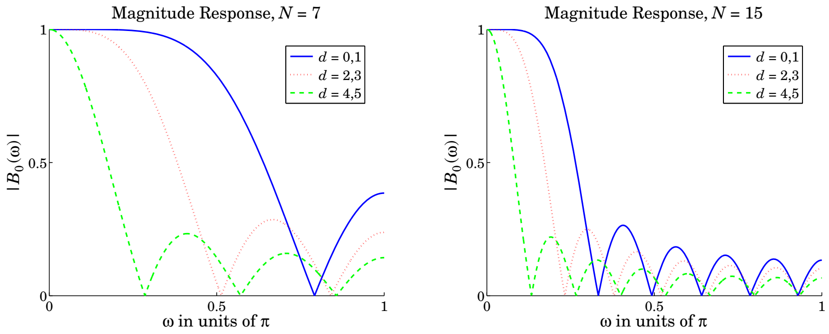


Fig. 3.7.1 LPSM filters of lengths $N = 7, 15$, and orders $d = 0, 2, 4$.

they satisfy two flatness conditions: $B_0''(0) = B_0^{(4)}(0) = 0$. The cutoff frequencies are *approximately* doubled and tripled in the cases $d = 2$ and $d = 4$, as compared to $d = 0$.

A direct consequence of the moment constraints (3.7.11) is that the moments of the input signal $y(n)$ are *preserved* by the filtering operation (3.2.33), that is,

$$\boxed{\sum_n n^i \hat{x}(n) = \sum_n n^i y(n)\,, \qquad i = 0, 1, \dots, d} \tag{3.7.16}$$

This can be proved easily working in the frequency domain. Differentiating the filtering equation $\hat{X}(\omega) = B_0(\omega)Y(\omega)$ i times, and using the product rules of differentiation, we obtain:

$$\hat{X}^{(i)}(\omega) = \sum_{j=0}^{i} \binom{i}{j} B_0^{(j)}(\omega) Y^{(i-j)}(\omega)$$

Setting $\omega = 0$ and using the moment constraints satisfied by the filter, $B_0^{(j)}(0) = \delta(j)$, we observe that only the $j = 0$ term will contribute to the above sum, giving:

$$\hat{X}^{(i)}(0) = B_0(0)Y^{(i)}(0) = Y^{(i)}(0)\,, \qquad i = 0, 1, \dots, d$$

which implies Eq. (3.7.16), by virtue of Eq. (3.7.14) as applied to $x(n)$ and $y(n)$.

The preservation of moments is a useful property in applications, such as spectroscopic analysis or ECG processing, in which the desired signal has one or more sharp peaks, whose widths must be preserved by the smoothing operation. In particular, the second moment corresponding to $i = 2$ in Eq. (3.7.16) is a measure of the square of the width [42–52,56,58,178].

The above moment constraints can be used in a direct way to design the LPSM filters. We consider first the more general problem of designing an optimum length-N filter that *minimizes* the NRR subject to $d + 1$ arbitrary moment constraints. That is, minimize

$$\mathcal{R} = \mathbf{b}^T\mathbf{b} = \sum_{n=-M}^{M} b(n)^2 = \min \tag{3.7.17}$$

subject to the $d + 1$ constraints, with a given $\mathbf{u} = [u_0, u_1, \dots, u_d]^T$:

$$\mathbf{s}_i^T\mathbf{b} = \sum_{n=-M}^{M} n^i b(n) = u_i\,, \quad i = 0, 1, \dots, d \quad \Rightarrow \quad S^T\mathbf{b} = \mathbf{u} \tag{3.7.18}$$

The minimization of Eq. (3.7.17) subject to (3.7.18) can be carried out with the help of Lagrange multipliers, that is, adding the constraint terms to the performance index:

$$\mathcal{J} = \mathbf{b}^T\mathbf{b} + 2\sum_{i=0}^{d} \lambda_i(u_i - \mathbf{s}_i^T\mathbf{b}) = \mathbf{b}^T\mathbf{b} + 2\boldsymbol{\lambda}^T(\mathbf{u} - S^T\mathbf{b}) \tag{3.7.19}$$

The gradient of $\mathcal{J}$ with respect to the unknown filter $\mathbf{b}$ is:

$$\frac{\partial\mathcal{J}}{\partial\mathbf{b}} = 2\mathbf{b} - 2S\boldsymbol{\lambda}$$

Setting the gradient to zero, and solving for $\mathbf{b}$ gives:

$$\mathbf{b} = S\boldsymbol{\lambda} = [\mathbf{s}_0, \mathbf{s}_1, \dots, \mathbf{s}_d]\begin{bmatrix} \lambda_0 \\ \lambda_1 \\ \vdots \\ \lambda_d \end{bmatrix} = \sum_{i=0}^{d} \lambda_i \mathbf{s}_i$$

Component-wise this means that $b(n)$ has the polynomial form:

$$b(n) = \sum_{i=0}^{d} \lambda_i s_i(n) = \sum_{i=0}^{d} \lambda_i n^i\,, \qquad -M \le n \le M$$

The Lagrange multiplier vector $\boldsymbol{\lambda}$ is determined by imposing the desired constraint:

$$\mathbf{u} = S^T\mathbf{b} = S^T S\boldsymbol{\lambda} = F\boldsymbol{\lambda} \qquad \Rightarrow \qquad \boldsymbol{\lambda} = F^{-1}\mathbf{u}$$

resulting in the optimum $\mathbf{b}$:

$$\mathbf{b} = S\boldsymbol{\lambda} = SF^{-1}\mathbf{u} = S(S^T S)^{-1}\mathbf{u} = G\mathbf{u} \tag{3.7.20}$$

Since the solution minimizes the norm $\mathbf{b}^T\mathbf{b}$, it is recognized to be the *minimum-norm* solution of the $(d+1)\times N$ full-rank under-determined linear system $S^T\mathbf{b} = \mathbf{u}$, which can be obtained by the pseudoinverse of S^T, that is, $\mathbf{b} = (S^T)^+\mathbf{u}$, where according to Eq. (15.4.10), $(S^T)^+ = S(S^T S)^{-1}$. In MATLAB, we can simply write $\mathbf{b} = \text{pinv}(S^T)\mathbf{u}$.

Comparing this solution with Eqs. (3.7.7) and (3.7.8), we conclude that the LPSM filters $\mathbf{b}_m$ can be thought of as the optimum filters that have minimum NRR with constraint vectors $\mathbf{u} = \mathbf{u}_m$, that is, the minimization problems,

$$\boxed{\mathcal{R} = \mathbf{b}_m^T\mathbf{b}_m = \min\,, \quad \text{subject to} \;\; S^T\mathbf{b}_m = \mathbf{u}_m} \tag{3.7.21}$$

have solutions,

$$\mathbf{b}_m = SF^{-1}\mathbf{u}_m = G\mathbf{u}_m\,, \quad -M \le m \le M \tag{3.7.22}$$

and putting these together as the columns of B, we obtain Eq. (3.2.31):

$$B = [\dots,\mathbf{b}_m,\dots] = G[\dots,\mathbf{u}_m,\dots] = GS^T = SF^{-1}S^T \tag{3.7.23}$$

In particular, the steady-state LPSM filter $\mathbf{b}_0$ minimizes the NRR with the constraint vector $\mathbf{u} = \mathbf{u}_0 = [1,0,\dots,0]^T$. This was precisely the problem first formulated and solved using Lagrange multipliers by Schiaparelli [36].

Similarly, the interpolating filter $\mathbf{b}_t = G\mathbf{u}_t$ of Eq. (3.6.10) can be thought of as the solution of the constrained minimization problem:

$$\mathcal{R} = \mathbf{b}^T\mathbf{b} = \min\,, \quad \text{subject to } S^T\mathbf{b} = \mathbf{u}_t\,, \quad \text{where } \mathbf{u}_t = [1,t,t^2,\dots,t^d]^T$$

3.8 Predictive Differentiation Filters

Going back to the polynomial fit of Eq. (3.6.9), that is,

$$\hat{y}_t = \sum_{i=0}^{d} c_i t^i = \mathbf{c}^T\mathbf{u}_t = \mathbf{y}^T G\mathbf{u}_t = \mathbf{y}^T\mathbf{b}_t\,, \quad \text{where} \quad \mathbf{b}_t = G\mathbf{u}_t\,, \tag{3.8.1}$$

we recall that the differentiation filters (3.2.24) were derived by differentiating (3.8.1) at $t = 0$, and therefore, they correspond to the center of the data vector $\mathbf{y}$:

$$\begin{aligned} \hat{y}_t\big|_{t=0} &= c_0 = \mathbf{b}_0^T\mathbf{y} = \mathbf{g}_0^T\mathbf{y} \\ \hat{\dot{y}}_t\big|_{t=0} &= c_1 = \mathbf{g}_1^T\mathbf{y} \\ \hat{\ddot{y}}_t\big|_{t=0} &= 2c_2 = \mathbf{g}_2^T\mathbf{y}\,, \quad \text{etc.,} \end{aligned}$$

The first derivative at an arbitrary value of t is given by:

$$\hat{\dot{y}}_t = \mathbf{y}^T\dot{\mathbf{b}}_t\,, \quad \dot{\mathbf{b}}_t = G\dot{\mathbf{u}}_t$$

where the differentiation operation can be expressed as matrix multiplication:

$$\mathbf{u}_t = \begin{bmatrix} 1 \\ t \\ t^2 \\ \vdots \\ t^d \end{bmatrix} \Rightarrow \dot{\mathbf{u}}_t = \begin{bmatrix} 0 \\ 1 \\ 2t \\ \vdots \\ dt^{d-1} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & 0 \\ 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 2 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \cdots & d & 0 \end{bmatrix} \begin{bmatrix} 1 \\ t \\ t^2 \\ \vdots \\ t^{d-1} \\ t^d \end{bmatrix} \equiv \mathcal{D}\mathbf{u}_t \tag{3.8.2}$$

where $\mathcal{D}$ is the $(d{+}1)\times(d{+}1)$ matrix with the sequence of numbers $\{1,2,\dots,d\}$ along its first subdiagonal and zeros everywhere else. Such a matrix can be constructed trivially in MATLAB, for example, by:

```
D = diag(1:d, -1);
```

It follows that the first-order differentiation filter is $\dot{\mathbf{b}}_t = G\mathcal{D}\mathbf{u}_t$. In particular, the differentiation filter at the sample point $t = m$ is $\dot{\mathbf{b}}_m = G\mathcal{D}\mathbf{u}_m$ and the corresponding estimated derivative:

$$\hat{\dot{y}}_m = \dot{\mathbf{b}}_m^T\mathbf{y} = \mathbf{u}_m^T\mathcal{D}^T G^T\mathbf{y}\,, \quad -M \le m \le M \tag{3.8.3}$$

Stacking these together into a column vector, we obtain:

$$\hat{\dot{\mathbf{y}}} = S\mathcal{D}^T G^T\mathbf{y} = \dot{B}^T\mathbf{y}\,, \quad \text{where} \quad \dot{B} = G\mathcal{D}S^T = SF^{-1}\mathcal{D}S^T \tag{3.8.4}$$

so that $\dot{B}$ has the $\dot{\mathbf{b}}_m$ as columns. Higher-order derivatives correspond to higher powers of the matrix $\mathcal{D}$, for example, $\ddot{\mathbf{u}}_t = \mathcal{D}^2\mathbf{u}_t$, and so on, with the highest non-trivial power being $\mathcal{D}^d$, because $\mathcal{D}^{d+1} = 0$, or equivalently, because the elements of $\mathbf{u}_t$ are monomials up to t^d. Therefore, the order-i differentiation matrix will be:

$$\boxed{B^{(i)} = SF^{-1}\mathcal{D}^i S^T}\,, \quad i = 0,1,\dots,d \tag{3.8.5}$$

Centering the data vector $\mathbf{y}$ at time n and denoting the m-th column of $B^{(i)}$ by $\mathbf{b}_m^{(i)}$, we obtain the filtering equation for the i-th estimated derivative:

$$\hat{y}_{n+m}^{(i)} = \sum_{k=-M}^{M} b_m^{(i)}(k)\,y_{n+k} = \sum_{k=-M}^{M} b_m^{(i)}(-k)\,y_{n-k} \tag{3.8.6}$$

We note that at the data-vector center $m = 0$, we have $\mathbf{b}_0^{(i)} = \mathbf{g}_i$. For arbitrary t, we have $\mathbf{b}_t^{(i)} = G\mathcal{D}^i\mathbf{u}_t$ and we obtain the estimated/interpolated derivative:

$$\boxed{\hat{y}_{n+t}^{(i)} = \sum_{k=-M}^{M} b_t^{(i)}(k)\,y_{n+k} = \sum_{k=-M}^{M} b_t^{(i)}(-k)\,y_{n-k}} \tag{3.8.7}$$

As in Eq. (3.6.15), the redefinition $h_\tau^{(i)}(k) = b_{M+\tau}^{(i)}(M-k)$ will result into a causal version of the predictive differentiator filter, with Eq. (3.8.7) transforming into:

$$\boxed{\hat{y}_{n+\tau}^{(i)} = \sum_{k=0}^{N-1} h_\tau^{(i)}(k)\,y_{n-k}} \qquad \text{(causal predictive differentiator)} \tag{3.8.8}$$

One can easily obtain closed-form expressions for the differentiation filters $b_t^{(i)}(k)$ for $d = 0,1,2,3,4$ and arbitrary M, by replacing the variable m in Eqs. (3.3.7)–(3.3.12) by the variable t and differentiating i-times with respect to t. For example, for $d = 1,2,3,4$, we differentiate Eqs. (3.6.12) once to get the first derivative:

$$
\begin{aligned}
\dot{b}_t(k) &= \frac{k}{F_2} \\
\dot{b}_t(k) &= \frac{1}{F_2}k - \frac{F_2}{D_4}(2t) + \frac{F_0}{D_4}(2tk^2) \\
\dot{b}_t(k) &= \frac{F_6}{D_8}k - \frac{F_2}{D_4}(2t) + \frac{F_0}{D_4}(2tk^2) - \frac{F_4}{D_8}(3t^2k + k^3) + \frac{F_2}{D_8}(3t^2k^3) \\
\dot{b}_t(k) &= \frac{F_6}{D_8}k - \frac{D_{10}}{D}(2t) + \frac{E_8}{D}(2tk^2) - \frac{F_4}{D_8}(k3t^2 + k^3) \\
&\quad + \frac{F_2}{D_8}(3t^2k^3) + \frac{D_8}{D}(4t^3) - \frac{D_6}{D}(2tk^4 + k^2 4t^3) + \frac{D_4}{D}(4t^3k^4)
\end{aligned}
\tag{3.8.9}
$$

For the causal versions, we have for $d = 1$:

$$
\begin{aligned}
h_\tau(k) &= \frac{1}{F_0} + \frac{(M+\tau)(M-k)}{F_2} = \frac{M(M+1) + 3(M+\tau)(M-k)}{M(M+1)(2M+1)} \\
\dot{h}_\tau(k) &= \frac{M-k}{F_2} = \frac{3(M-k)}{M(M+1)(2M+1)}
\end{aligned}
\tag{3.8.10}
$$

where $k = 0, 1, \dots, N-1$. We note that $\dot{h}_\tau$ can be obtained by differentiating h_τ with respect to τ. The derivative filter is independent of τ because it corresponds to fitting a first-order polynomial. For $d = 2$, we have similarly,

$$
\begin{aligned}
h_\tau(k) &= \frac{F_4}{D_4} + \frac{1}{F_2}(M+\tau)(M-k) - \frac{F_2}{D_4}\big((M+\tau)^2 + (M-k)^2\big) + \frac{F_0}{D_4}(M+\tau)^2(M-k)^2 \\
\dot{h}_\tau(k) &= \frac{1}{F_2}(M-k) - \frac{F_2}{D_4}2(M+\tau) + \frac{F_0}{D_4}2(M+\tau)(M-k)^2
\end{aligned}
\tag{3.8.11}
$$

where, we recall from Eq. (3.3.11),

$$
\frac{F_4}{D_4} = \frac{3(3M^2+3M-1)}{(2M+3)(4M^2-1)}, \quad \frac{F_2}{D_4} = \frac{15}{(2M+3)(4M^2-1)}
$$

$$
\frac{F_0}{D_4} = \frac{45}{M(M+1)(2M+3)(4M^2-1)}, \quad \frac{1}{F_2} = \frac{3}{M(M+1)(2M+1)}
$$

Example 3.8.1: For the case $N = 5$, $d = 2$, we had found in Eqs. (3.2.5) and (3.2.16) that:

$$
S = [\mathbf{s}_0, \mathbf{s}_1, \mathbf{s}_2] = \begin{bmatrix} 1 & -2 & 4 \\ 1 & -1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \end{bmatrix}, \quad G = \frac{1}{35}\begin{bmatrix} -3 & -7 & 5 \\ 12 & -3.5 & -2.5 \\ 17 & 0 & -5 \\ 12 & 3.5 & -2.5 \\ -3 & 7 & 5 \end{bmatrix}
$$

The corresponding first- and second-order differentiation matrices will be:

$$
\dot{B} = G\mathcal{D}^1 S^T = \frac{1}{35}\begin{bmatrix} -27 & -17 & -7 & 3 & 13 \\ 6.5 & 1.5 & -3.5 & -8.5 & -13.5 \\ 20 & 10 & 0 & -10 & -20 \\ 13.5 & 8.5 & 3.5 & -1.5 & -6.5 \\ -13 & -3 & 7 & 17 & 27 \end{bmatrix}, \quad \mathcal{D}^1 = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 2 & 0 \end{bmatrix}
$$

$$
\ddot{B} = G\mathcal{D}^2 S^T = \frac{1}{35}\begin{bmatrix} 10 & 10 & 10 & 10 & 10 \\ -5 & -5 & -5 & -5 & -5 \\ -10 & -10 & -10 & -10 & -10 \\ -5 & -5 & -5 & -5 & -5 \\ 10 & 10 & 10 & 10 & 10 \end{bmatrix}, \quad \mathcal{D}^2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 2 & 0 & 0 \end{bmatrix}
$$

The central columns agree with Eq. (3.2.25). The interpolating smoothing and first-order differentiation filters are given by:

$$
\mathbf{b}_t = G\mathbf{u}_t = \frac{1}{35}\begin{bmatrix} -3 & -7 & 5 \\ 12 & -3.5 & -2.5 \\ 17 & 0 & -5 \\ 12 & 3.5 & -2.5 \\ -3 & 7 & 5 \end{bmatrix}\begin{bmatrix} 1 \\ t \\ t^2 \end{bmatrix} = \frac{1}{35}\begin{bmatrix} -3 - 7t + 5t^2 \\ 12 - 3.5t - 2.5t^2 \\ 17 - 5t^2 \\ 12 + 3.5t - 2.5t^2 \\ -3 + 7t + 5t^2 \end{bmatrix}
$$

$$
\dot{\mathbf{b}}_t = G\mathcal{D}\mathbf{u}_t = \frac{1}{35}\begin{bmatrix} -3 & -7 & 5 \\ 12 & -3.5 & -2.5 \\ 17 & 0 & -5 \\ 12 & 3.5 & -2.5 \\ -3 & 7 & 5 \end{bmatrix}\begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 2 & 0 \end{bmatrix}\begin{bmatrix} 1 \\ t \\ t^2 \end{bmatrix} = \frac{1}{35}\begin{bmatrix} -7 + 10t \\ -3.5 - 5t \\ -10t \\ 3.5 - 5t \\ 7 + 10t \end{bmatrix}
$$

where $\dot{\mathbf{b}}_t$ can be obtained either by the indicated matrix multiplication or by simply differentiating $\mathbf{b}_t$ with respect to t. □

The MATLAB function `lpdiff` implements the design of the differentiation matrices:

```
B = lpdiff(N,d,i);          % differentiation filters
```

Like `lpsm`, it carries out a Gram-Schmidt QR-transformation on the monomial basis S and constructs the $B^{(i)}$ by:

$$
\begin{aligned}
&S = QR, \quad Q^TQ = I, \quad R = \text{upper triangular} \\
&G = S(S^TS)^{-1} = QR^{-T} \\
&B^{(i)} = GD^iS^T = Q(R^{-T}D^iR^T)Q^T
\end{aligned}
$$

The predictive/interpolating differentiation filters $\mathbf{b}_t^{(i)}$ are the minimum-norm solution of the under-determined linear system $S^T\mathbf{b} = \mathcal{D}^i\mathbf{u}_t$, or, equivalently the solution of the constrained minimization problem:

$$
\mathcal{R} = \mathbf{b}^T\mathbf{b} = \min, \quad \text{subject to } S^T\mathbf{b} = \mathcal{D}^i\mathbf{u}_t
$$

The MATLAB function `lpinterp` implements the design of predictive and interpolating differentiation filters, essentially carrying out the operation $\mathbf{b} = \text{pinv}(S^T)\mathcal{D}^i\mathbf{u}_t$:

```
b = lpinterp(N,d,t,i);     % local polynomial interpolation and differentiation filters
```

The case $i = 0$ corresponds to the predictive interpolation filters of Sec. 3.6. For the integer values $t = m$, $-M \le m \le M$, the filter $\mathbf{b}$ agrees with the columns of $B^{(i)}$.

Example 3.8.2: Fig. 3.8.1 illustrates the performance of the local polynomial differentiation filters on noiseless and noisy signals.

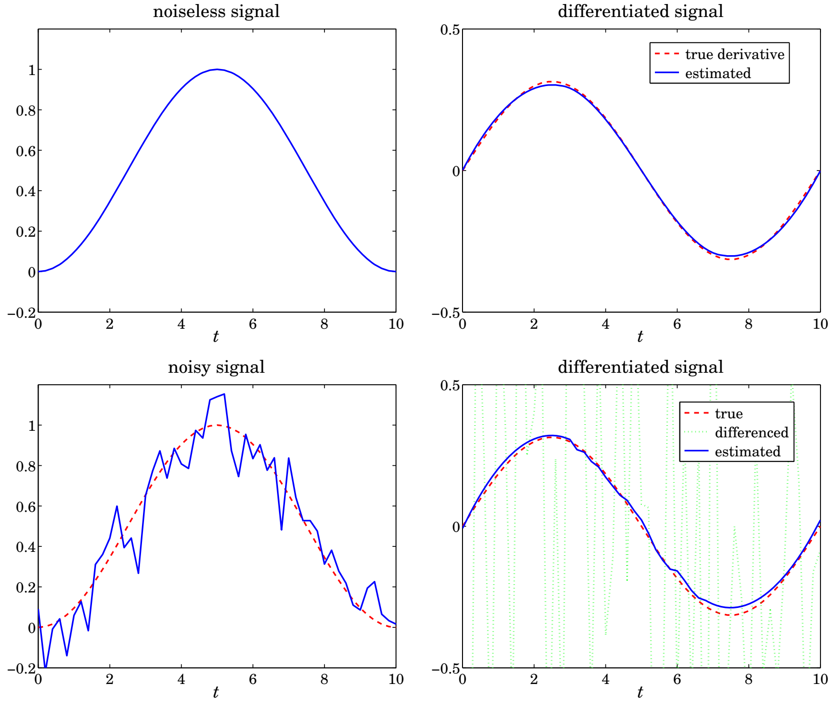


Fig. 3.8.1 Differentiating noisy signals.

The noiseless signal is a raised cosine $s(t) = 0.5 - 0.5\cos(\omega t)$, with $0 \le t \le T$ and $\omega = 2\pi/T$, so that it spans one cycle. Choosing a sampling time interval $\Delta t = T/L$, we can construct a noisy signal sampled at time instants $t_n = n\Delta t = nT/L$, $n = 0, 1, \dots, L$, by adding zero-mean white gaussian noise v_n of variance, say σ^2, so that the noisy observations are:

$$y_n = s(t_n) + v_n\,, \quad n = 0, 1, \dots, L$$

The first derivative of $s(t)$ is $\dot{s}(t) = 0.5\omega\sin(\omega t)$ and its samples, $\dot{s}(t_n) = 0.5\omega\sin(\omega t_n)$. The upper-left graph shows $s(t_n)$ versus t_n, with $T = 10$ and $L = 50$. The upper-right graph shows $\dot{s}(t_n)$ (dashed line) together with the estimated derivative (solid line) of the original signal $s(t_n)$ filtered through an LPSM differentiation filter designed with $N = 31$ and polynomial order $d = 3$. The output of the filter is divided by Δt in order to adjust its dimensions.

The bottom-left graph shows the noisy signal y_n. In the bottom-right graph, the output (solid line) of the same differentiation filter applied to the noisy signal y_n is compared with the true noiseless differentiated signal $\dot{s}_n$, as well as to the differenced signal $\text{diff}(y)/\Delta t$. The following MATLAB code illustrates the generation of the bottom-right graph:

```
T = 10; L = 50; Dt =T/L; w = 2*pi/T; sigma = 0.1;
t = 0:Dt:T;
s = 0.5 - 0.5*cos(w*t);                  % noiseless signal

seed=100; randn('state',seed);
y = s + sigma * randn(1,length(s));      % noisy signal

N = 31; d = 3; B1 = lpdiff(N,d,1);       % first-order differentiation filter

sd = 0.5*w*sin(w*t);                     % derivative of s(t)
xd = lpfilt(B1,s)/Dt;                    % estimated derivative of s(t)
x1 = lpfilt(B1,y)/Dt;                    % estimated derivative from the noisy signal
yd = diff(y)/Dt;   td = t(2:end);        % differenced signal estimates the derivative

plot(t,sd,'--', td,yd,':', t,x1,'-');
```

The differencing operation amplifies the noise and renders the estimated derivative useless, whereas the local-polynomial derivative is fairly accurate. The filtering operation is carried out by the function `lpfilt`, which is explained in the next section. □

3.9 Filtering Implementations

In smoothing a length-L signal block y_n, $n = 0, 1, \dots, L-1$, with a double-sided filter h_m, $-M \le m \le M$, the output signal $\hat{x}_n$ is given by the convolutional form:

$$\hat{x}_n = \sum_{m=\max(-M,n-L+1)}^{\min(n,M)} h_m y_{n-m}\,, \qquad -M \le n \le L + M - 1 \tag{3.9.1}$$

The length of $\hat{x}_n$ is $L + 2M$, and the first $2M$ and last $2M$ output samples correspond to the input-on and input-off transients, while the central $L - 2M$ points, $M \le n \le L - M - 1$, correspond to the steady-state output computed from the steady-state version of Eq. (3.9.1):

$$\hat{x}_n = \sum_{m=-M}^{M} h_m y_{n-m}\,, \qquad M \le n \le L - M - 1 \tag{3.9.2}$$

The range of the output index n and the limits of summation in (3.9.1) are determined from the inequalities $-M \le m \le M$ and $0 \le n - m \le L - 1$ that must be satisfied by the indices of h_m and y_{n-m}. However, only the subrange $\{\hat{x}_n\,,\ 0 \le n \le L - 1\}$ is of interest since these output samples represent the smoothed values of the corresponding input samples $\{y_n\,,\ n = 0, 1, \dots, L - 1\}$. This is illustrated in Fig. 3.9.1.

The first and last M samples in the subrange $0 \le n \le L - 1$ are still parts of the input-on and input-off transients. To clarify these remarks, we consider the case $L = 8$, $M = 2$. The full output (3.9.1) may be represented by the usual convolution matrix of the filter acting on the input signal block:

$$
\begin{bmatrix} \hat{x}_{-2} \\ \hat{x}_{-1} \\ \dots \\ \hat{x}_0 \\ \hat{x}_1 \\ \dots \\ \hat{x}_2 \\ \hat{x}_3 \\ \hat{x}_4 \\ \hat{x}_5 \\ \dots \\ \hat{x}_6 \\ \hat{x}_7 \\ \dots \\ \hat{x}_8 \\ \hat{x}_9 \end{bmatrix}
=
\begin{bmatrix}
h_{-2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
h_{-1} & h_{-2} & 0 & 0 & 0 & 0 & 0 & 0 \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
h_0 & h_{-1} & h_{-2} & 0 & 0 & 0 & 0 & 0 \\
h_1 & h_0 & h_{-1} & h_{-2} & 0 & 0 & 0 & 0 \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
h_2 & h_1 & h_0 & h_{-1} & h_{-2} & 0 & 0 & 0 \\
0 & h_2 & h_1 & h_0 & h_{-1} & h_{-2} & 0 & 0 \\
0 & 0 & h_2 & h_1 & h_0 & h_{-1} & h_{-2} & 0 \\
0 & 0 & 0 & h_2 & h_1 & h_0 & h_{-1} & h_{-2} \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
0 & 0 & 0 & 0 & h_2 & h_1 & h_0 & h_{-1} \\
0 & 0 & 0 & 0 & 0 & h_2 & h_1 & h_0 \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
0 & 0 & 0 & 0 & 0 & 0 & h_2 & h_1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & h_2
\end{bmatrix}
\begin{bmatrix} y_0 \\ y_1 \\ \dots \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ \dots \\ y_6 \\ y_7 \end{bmatrix}
$$

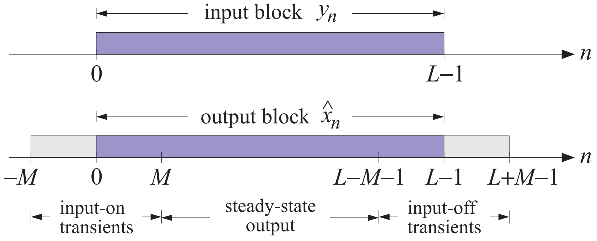


Fig. 3.9.1 Input and output signal blocks from a double-sided filter.

This matrix can be constructed in MATLAB with the built-in function `convmtx`, or with its sparse version `convmat`, or with the function `datamat`, the latter two being part of the OSP toolbox. Defining $\mathbf{h} = [h_{-M}, \dots, h_0, \dots, h_M]^T$, we have the syntax:

```
H = convmtx(h,L);        % built-in convolution matrix
H = convmat(h,L);        % sparse version of convmtx
H = datamat(h,L-1);      % used extensively in Chap. 15
```

Dropping the first and last two outputs, we obtain the outputs in the subrange $0 \le n \le 7$:

$$
\hat{\mathbf{x}} =
\begin{bmatrix} \hat{x}_0 \\ \hat{x}_1 \\ \dots \\ \hat{x}_2 \\ \hat{x}_3 \\ \hat{x}_4 \\ \hat{x}_5 \\ \dots \\ \hat{x}_6 \\ \hat{x}_7 \end{bmatrix}
=
\begin{bmatrix}
h_0 & h_{-1} & h_{-2} & 0 & 0 & 0 & 0 & 0 \\
h_1 & h_0 & h_{-1} & h_{-2} & 0 & 0 & 0 & 0 \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
h_2 & h_1 & h_0 & h_{-1} & h_{-2} & 0 & 0 & 0 \\
0 & h_2 & h_1 & h_0 & h_{-1} & h_{-2} & 0 & 0 \\
0 & 0 & h_2 & h_1 & h_0 & h_{-1} & h_{-2} & 0 \\
0 & 0 & 0 & h_2 & h_1 & h_0 & h_{-1} & h_{-2} \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
0 & 0 & 0 & 0 & h_2 & h_1 & h_0 & h_{-1} \\
0 & 0 & 0 & 0 & 0 & h_2 & h_1 & h_0
\end{bmatrix}
\begin{bmatrix} y_0 \\ y_1 \\ \dots \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ \dots \\ y_6 \\ y_7 \end{bmatrix}
\equiv H\mathbf{y} \tag{3.9.3}
$$

The first two and last two of these outputs are still transient and are being computed with only a subset of the filter coefficients, and therefore, may not adequately represent the corresponding smoothed values. This so-called "end-point problem" has been addressed repeatedly with a number of solutions.

One method that is widely used by the government to process census and business-cycle data (e.g., the X12-ARIMA method) is to backcast and forecast M estimated values at the beginning and end of the length-L input block, so that y_n is now defined over $-M \le n \le L-1+M$, and the desired output samples over the subrange $0 \le n \le L-1$ will be steady-state outputs being computed with the full filter.

Another method is to use different filters for the first M and last M outputs. For example, one can take the outputs $\hat{y}_{n+m}$ of the LPSM filters $b_m(k)$ to estimate the initial and final M transients, while using the central filter $b_0(k)$ for the steady-state outputs. Indeed, the first time index when one can use the steady-state filter $b_0(k)$ is $n = M$:

$$
\hat{x}_M = \hat{y}_M = \sum_{k=-M}^{M} b_0(k) y_{M+k}
$$

Instead of calculating the previous output $\hat{x}_{M-1}$ using the transient version of $b_0(k)$,

$$
\hat{x}_{M-1} = \sum_{k=-(M-1)}^{M} b_0(k) y_{M-1+k}
$$

one could estimate $\hat{x}_{M-1}$ using $\hat{y}_{M+m}$ with $m = -1$, that is, using $b_{-1}(k)$, and using $b_{-2}(k), b_{-3}(k), \dots, b_{-M}(k)$ for the other initial M outputs:

$$
\begin{aligned}
\hat{x}_{M-1} &= \hat{y}_{M-1} = \sum_{k=-M}^{M} b_{-1}(k) y_{M+k} \\
\hat{x}_{M-2} &= \hat{y}_{M-2} = \sum_{k=-M}^{M} b_{-2}(k) y_{M+k} \\
&\vdots \\
\hat{x}_0 &= \hat{y}_{M-M} = \sum_{k=-M}^{M} b_{-M}(k) y_{M+k}
\end{aligned}
\tag{3.9.4}
$$

Similarly, one can use the filters $b_m(k)$ for $m = 1, 2, \dots, M$ to calculate the last M smoothed outputs, starting with the last steady-state output at $n = L - 1 - M$ and proceeding to the end $n = L - 1$:

$$\begin{aligned}
\hat{x}_{L-M} &= \hat{y}_{L-1-M+1} = \sum_{k=-M}^{M} b_1(k) y_{L-1-M+k} \\
\hat{x}_{L-M+1} &= \hat{y}_{L-1-M+2} = \sum_{k=-M}^{M} b_2(k) y_{L-1-M+k} \\
&\vdots \\
\hat{x}_{L-1} &= \hat{y}_{L-1-M+M} = \sum_{k=-M}^{M} b_M(k) y_{L-1-M+k}
\end{aligned} \tag{3.9.5}$$

The following example illustrates the computational steps for the input-on, steady, and input-off output samples, where we denoted $b_{m,k} = b_m(k)$ for simplicity:

$$\hat{\mathbf{x}} = \begin{bmatrix} \hat{x}_0 \\ \hat{x}_1 \\ \cdots \\ \hat{x}_2 \\ \hat{x}_3 \\ \hat{x}_4 \\ \hat{x}_5 \\ \cdots \\ \hat{x}_6 \\ \hat{x}_7 \end{bmatrix} = \begin{bmatrix} b_{-2,-2} & b_{-2,-1} & b_{-2,0} & b_{-2,1} & b_{-2,2} & 0 & 0 & 0 \\ b_{-1,-2} & b_{-1,-1} & b_{-1,0} & b_{-1,1} & b_{-1,2} & 0 & 0 & 0 \\ \hdotsfor{8} \\ b_{0,-2} & b_{0,-1} & b_{0,0} & b_{0,1} & b_{0,2} & 0 & 0 & 0 \\ 0 & b_{0,-2} & b_{0,-1} & b_{0,0} & b_{0,1} & b_{0,2} & 0 & 0 \\ 0 & 0 & b_{0,-2} & b_{0,-1} & b_{0,0} & b_{0,1} & b_{0,2} & 0 \\ 0 & 0 & 0 & b_{0,-2} & b_{0,-1} & b_{0,0} & b_{0,1} & b_{0,2} \\ \hdotsfor{8} \\ 0 & 0 & 0 & b_{1,-2} & b_{1,-1} & b_{1,0} & b_{1,1} & b_{1,2} \\ 0 & 0 & 0 & b_{2,-2} & b_{2,-1} & b_{2,0} & b_{2,1} & b_{2,2} \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ \cdots \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ \cdots \\ y_6 \\ y_7 \end{bmatrix} = H\mathbf{y}$$

In particular, for $N = 5$ and $d = 2$, the convolutional filtering matrix will be:

$$\hat{\mathbf{x}} = \begin{bmatrix} \hat{x}_0 \\ \hat{x}_1 \\ \cdots \\ \hat{x}_2 \\ \hat{x}_3 \\ \hat{x}_4 \\ \hat{x}_5 \\ \cdots \\ \hat{x}_6 \\ \hat{x}_7 \end{bmatrix} = \frac{1}{35} \begin{bmatrix} 31 & 9 & -3 & -5 & 3 & 0 & 0 & 0 \\ 9 & 13 & 12 & 6 & -5 & 0 & 0 & 0 \\ \hdotsfor{8} \\ -3 & 12 & 17 & 12 & -3 & 0 & 0 & 0 \\ 0 & -3 & 12 & 17 & 12 & -3 & 0 & 0 \\ 0 & 0 & -3 & 12 & 17 & 12 & -3 & 0 \\ 0 & 0 & 0 & -3 & 12 & 17 & 12 & -3 \\ \hdotsfor{8} \\ 0 & 0 & 0 & -5 & 6 & 12 & 13 & 9 \\ 0 & 0 & 0 & 3 & -5 & -3 & 9 & 31 \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ \cdots \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ \cdots \\ y_6 \\ y_7 \end{bmatrix} = H\mathbf{y}$$

with entries obtained from the matrix B of Eq. (3.2.16):

$$B = \frac{1}{35} \begin{bmatrix} 31 & 9 & -3 & -5 & 3 \\ 9 & 13 & 12 & 6 & -5 \\ -3 & 12 & 17 & 12 & -3 \\ -5 & 6 & 12 & 13 & 9 \\ 3 & -5 & -3 & 9 & 31 \end{bmatrix} = [\mathbf{b}_{-2}, \mathbf{b}_{-1}, \mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2]$$

More generally, given any smoothing (or differentiation) matrix B whose central column contains the (reversed) steady-state filter, and its other columns, the (reversed) filters to be used for the initial and final transients, one can uniquely construct the corresponding $L\times L$ convolutional matrix H for filtering a length-L block of data $\mathbf{y}$.

The procedure is straightforward. First construct the ordinary full $(L+2M)\times L$ convolution matrix for the central filter, then delete its first M and last M rows, and finally, replace the first M and last M rows of the result by the transient filters.

The following MATLAB code segment illustrates the procedure, where the matrix B is assumed to have size $N\times N$, with $N = 2M + 1$, with the central column being the reversed steady-state filter and the other columns, the reversed transient filters:

```
H = convmat(flip(B(:,M+1)), L);          % ordinary (L+2M)×L convolution matrix
H = H(M+1:L+M,:);                        % extract the L×L convolution submatrix
H(1:M, 1:N) = B(:,1:M)';                 % redefine upper-left M×L corner
H(L-M+1:L, L-N+1:L) = B(:,M+2:N)';       % redefine lower-right M×L corner
```

The function `flip` reverses the central column of B because `convmat` expects as input the actual filter, not its reverse. The above steps have been incorporated into the function `lpmat` with syntax:

```
H = lpmat(B,L);         % local polynomial filter matrix of size L×L
```

Once the $L\times L$ matrix H is constructed, the actual filtering of a length-L input block $\mathbf{y}$ is straightforward, that is, $\hat{\mathbf{x}} = H\mathbf{y}$, and efficient because H is defined as sparse.

An alternative way to structure the filtering operation is to directly use Eqs. (3.9.4) and (3.9.5) for the transient parts and the following equation for the steady part:

$$\hat{x}_n = \sum_{k=-M}^{M} b_0(k) y_{n+k}\,, \quad M \le n \le L - 1 - M \tag{3.9.6}$$

The following MATLAB code illustrates this approach:

```
y = B(:,1:M)' * x(1:N);                  % first M transient outputs
for n = M+1:L-M,                         % middle L−2M steady-state outputs
   y = [y; B(:,M+1)' * x(n-M:n+M)];      % filtered by central column of B
end
y = [y; B(:,M+2:N)' * x(L-N+1:L)];       % last M transient outputs
```

These steps are implemented in the MATLAB function `lpfilt2`. A faster version is the function `lpfilt`, which uses MATLAB's built-in filtering functions. Thus, three possible ways of computing the filtered output $\hat{\mathbf{x}}$ given a smoothing matrix B are as follows (assuming that $\mathbf{y}$ is a length-L column vector):

```
x_hat = lpmat(B,L)*y;     % use L×L convolution matrix constructed from B
x_hat = lpfilt2(B,y);     % use directly the filtering equations (3.9.4)–(3.9.6)
x_hat = lpfilt(B,y);      % fast version using the function filtdbl
```

The function `lpfilt` internally calls the function `filtdbl`, which uses the built-in function `conv` to implement the FIR filtering by the steady-state double-sided central filter. The following code segment shows the essential part of `lpfilt`:

```
x_hat = filtdbl(flip(B(:,M+1)), y);                % filter with the central column of B
x_hat(1:M) = B(:,1:M)' * y(1:N);                   % correct the first M transient outputs
x_hat(end-M+1:end) = B(:,M+2:N)' * y(end-N+1:end); % correct the last M transient outputs
```

where the function `filtdbl` has usage:

```
y = filtdbl(h,x);          % filtering by double-sided FIR filter
```

The function `filtdbl` is essentially the ordinary convolution of the length-$(2M+1)$ filter $\mathbf{h}$ and the length-L signal $\mathbf{x}$, with the first M and last M output points discarded. The result is equivalent to that obtained using the convolution submatrix, as for example, in Eq. (3.9.3). We note, in particular, that the B matrix that gives rise to (3.9.3) is:

$$B = \begin{bmatrix} h_0 & h_1 & h_2 & 0 & 0 \\ h_{-1} & h_0 & h_1 & h_2 & 0 \\ h_{-2} & h_{-1} & h_0 & h_1 & h_2 \\ 0 & h_{-2} & h_{-1} & h_0 & h_1 \\ 0 & 0 & h_{-2} & h_{-1} & h_0 \end{bmatrix}$$

and contains the reversed filter $\mathbf{h}$ in the central column and the transient subfilters in the other columns.

There are other methods of handling the end-point problem, most notably Musgrave's *minimum-revision* method that uses end-point asymmetric filters constructed from a given central filter $\mathbf{h}$. We will discuss it in detail in Sec. 9.8. Here, we note that the output of this method is a B matrix, which can be passed directly into the filtering function `lpfilt`. The MATLAB function `minrev` implements Musgrave's method:

```
B = minrev(h,R);          % Musgrave's minimum revision asymmetric filters
```

where R is a scalar parameter to be explained in Sec. 9.8. The method is widely used in the X-11 method of seasonal adjustment and trend extraction.

Example 3.9.1: Schiaparelli was the first one to systematically pose and solve the minimum-NRR filtering problem. He gave the solution to many specific cases, such as filter lengths $N = 5$–13, and polynomial orders $d = 3, 4$.

Here, we reproduce the example from Schiaparelli's paper on smoothing lunar observations, the signal y_n being a measure of the moon's influence on atmospheric effects. Fig. 3.9.2 shows 30 noisy observations (one for each lunar day) and their smoothed versions produced with an LPSM filter of length $N = 13$ and polynomial order $d = 3$ on the left, and $d = 4$ on the right (Schiaparelli's case).

The central filters for the $d = 3$ and $d = 4$ cases are:

$$\mathbf{b}_0 = \frac{1}{143}[-11,\ 0,\ 9,\ 16,\ 21,\ 24,\ 25,\ 24,\ 21,\ 16,\ 9,\ 0,\ -11]$$

$$\mathbf{b}_0 = \frac{1}{2431}[110,\ -198,\ -135,\ 110,\ 390,\ 600,\ 677,\ 600,\ 390,\ 110,\ -135,\ -198,\ 110]$$

The following program segment illustrates the computations:

```
Y = loadfile('schiaparelli.dat');        % data file available in the OSP toolbox
n = Y(:,1); y = Y(:,2);                  % extract n and y_n from the columns of Y

N=13; d=3; M=floor(N/2);                 % filter length and polynomial order
B = lpsm(N,d);                           % construct LPSM matrix B
x = lpfilt(B,y);                         % filter noisy observations
b0 = B(:,M+1);                           % middle column of B
x0 = filtdbl(b0,y);                      % filter with b0 only
plot(n,y,'.', n,x,'-', n,x0,'--');
```

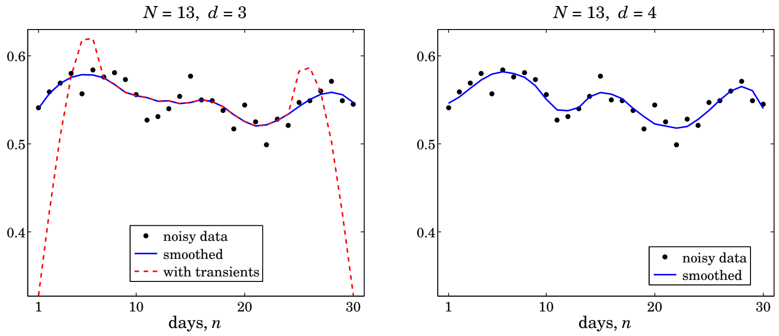


Fig. 3.9.2 Schiaparelli's smoothing example.

where the function `loadfile` extracts only the numerical data from the data file. In the left graph, we have also added the result of filtering with the steady-state filter $\mathbf{b}_0$, which illustrates the end-point problem. The two filtered curves differ only in their first 6 and last 6 points. □

Example 3.9.2: *Global Warming Trends.* Fig. 3.9.3 shows the annual average temperature anomalies (i.e., the differences with respect to the average of the period 1961-90) over the period 1856–2005 in the northern hemisphere. The data are available from the web site: `https://crudata.uea.ac.uk/cru/data/crutem2/`.

Five trend extraction methods are compared. In the upper left, a local polynomial smoothing filter was used of length $N = 65$ and polynomial order $d = 3$. The following MATLAB code illustrates the generation of that graph:

```
Y = loadfile('tavenh2v.dat');     % data file available in the OSP toolbox
n = Y(:,1); y = Y(:,14);          % extract n and y_n from Y

N = 65; d = 3; B = lpsm(N,d);     % design the LPSM matrix B
x = lpfilt(B,y);                  % smooth the data vector y

figure; plot(n,y,':', n,x,'-');
```

In the upper-right graph, a minimum-roughness, or minimum-R_s, Henderson filter was used with length $N = 65$, polynomial order $d = 3$, and smoothing order $s = 2$. Such filters are discussed in Sec. 4.2. The resulting trend is noticeably smoother than that of the LPSM filter on the upper-left.

The middle-left graph uses the SVD signal enhancement method, described in Chap. 15, with embedding order $M = 10$ and rank $r = 2$, with $K = 40$ iterations. The middle-right graph uses the Whittaker-Henderson smoothing method, discussed in Sec. 8.1, with smoothing order $s = 2$ and smoothing parameter $\lambda = 10^4$.

The lower left and right graphs use the Whittaker-Henderson method with the L_1 criterion with differentiation orders $s = 2$ and $s = 3$ and smoothing parameter $\lambda = 10$, implemented with the CVX package.[†] The $s = 2$ case represents the smoothed signal in piece-wise linear form, and the $s = 3$ case, in piece-wise parabolic form. This is further discussed in Sec. 8.7.

[†] `http://cvxr.com/cvx/`

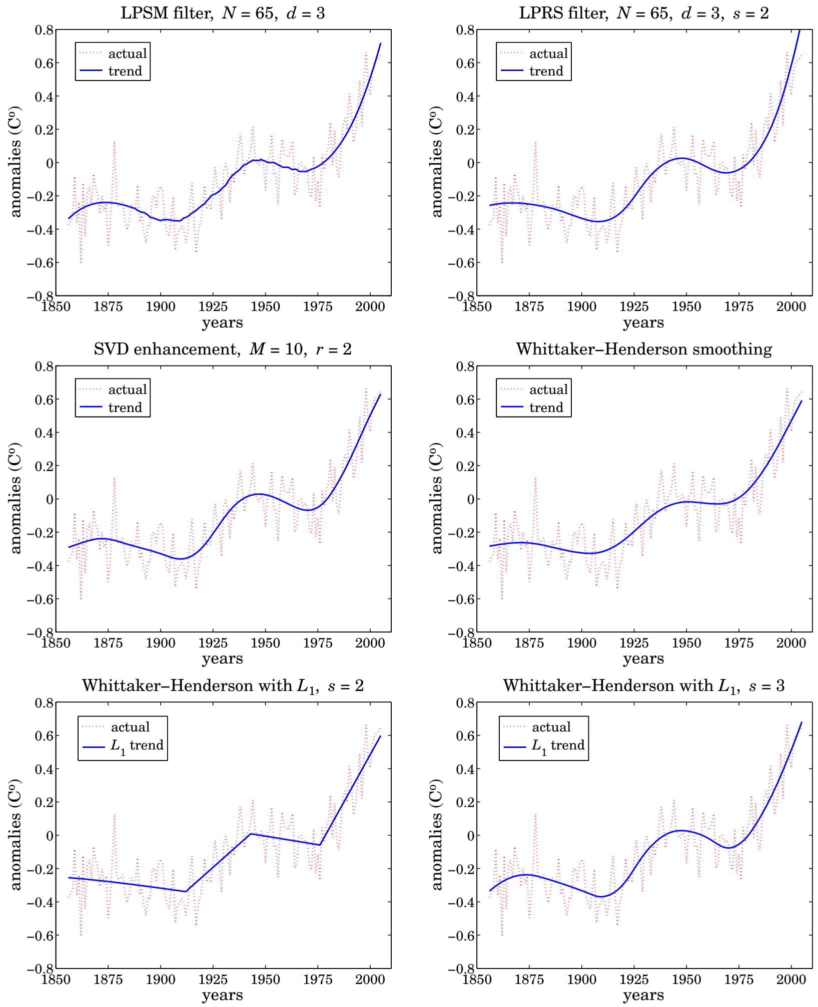


Fig. 3.9.3 Temperature trends determined by five methods.

The following MATLAB code segment illustrates the computation of the corresponding smoothed signals for these four methods:

```
N=65; d=3; s=2; x = lpfilt(lprs(N,d,s), y);   % minimum-R_s Henderson filter
M=10; r=2; K=40; x = svdenh(y,M,r,K);         % SVD enhancement method
la = 10000; s=2; x = whsm(y,la,s);            % Whittaker-Henderson smoothing

s = 2; la = 10; N = length(y);                % Whittaker-Henderson with L_1
D = diff(eye(N),s);                           % s-fold differentiation matrix
```

```
cvx_begin                                     % use CVX package
    variable x(N)
    minimize( sum_square(y-x) + la * norm(D*x,1) )
cvx_end
```

All methods adequately handle the end-point problem. Repeating the same filtering operation several times results in even smoother trend signals. For example, Fig. 3.9.4 shows the result of repeating the filtering operation two additional times. The following MATLAB code illustrates the generation of the left graph:

```
N = 65; d=3; B = lpsm(N,d); x = y;
for i=1:3, x = lpfilt(B,x); end
figure;  plot(n,y,':', n,x,'-');
```

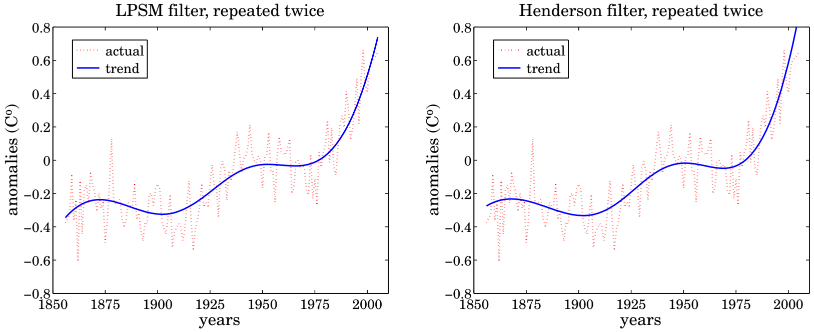


Fig. 3.9.4 Filtering repeated two additional times.

For the steady-state filters $B_0(\omega)$, filtering a total of K times is equivalent to an overall filter $[B_0(\omega)]^K$, an operation which makes a flat passband even flatter and a small stopband even smaller. The properties of iterated smoothing by local polynomial filters has been studied by De Forest, Schoenberg, and Greville [67,83,86].

Fig. 3.9.5 shows the estimated derivatives (solid line) of the temperature signal obtained by filtering it with the LPSM derivative filters, and compares them with the ordinary differencing operation, diff(y), in MATLAB notation. Clearly, differencing is simply too noisy to give any usable results.

The upper two graphs compute the first derivative of the input by $\dot{\hat{x}} = \text{lpfilt}(B_1, y)$ with the differentiator matrix obtained from $B_1 = \text{lpdiff}(N, d, i)$ with $N = 65$ and $i = 1$, and with $d = 1$ in the upper-left, and $d = 2$ in the upper-right graph. During the two periods of almost linear growth from 1910–1940 and 1970-2005, the derivative signal becomes an almost flat positive constant (i.e., the slope). During the other periods, the temperature signal has a very slow upward or downward trend and the derivative signal is almost zero.

We note the flat end-points in the case $d = 1$, which are due to the fact that the asymmetric derivative filters are the same at the end-points ranges as shown in the first equation of (3.8.9). The case $d = 2$ estimates the end-point derivatives better and possibly indicates a faster than linear growth in recent years.

The lower-left graph uses a minimum-R_s derivative filter with $N = 65$, $d = 2$, and smoothness order $s = 3$, resulting in a noticeably smoother estimated derivative than the LPSM case (the W input in `lpdiff` is discussed in the next section.) Finally, the lower-right graph shows the second derivative computed with the filter $B_2 = \text{lpdiff}(N, d, i)$ with $i = 2$, and compares it with the second difference signal, $\text{diff}\big(\text{diff}(y)\big)$, which is even more noisy than the first difference. The following MATLAB code illustrates the computations:

```
d=1; i=1; B1 = lpdiff(N,d,i);                       % LPSM differentiation filters
plot(n, lpfilt(B1,y), n(2:end), diff(y),':');      % upper-left graph

d=2; i=1; B1 = lpdiff(N,d,i);
plot(n, lpfilt(B1,y), n(2:end), diff(y),':');      % upper-right graph

s=3; W = diag(hend(N,s));                           % Henderson weighting matrix
d=2; i=1; B1 = lpdiff(N,d,i,W);                     % LPRS differentiation filters
plot(n, lpfilt(B1,y), n(2:end), diff(y),':');      % lower-left graph

d=2; i=2; B2 = lpdiff(N,d,i);                       % second derivative filters
plot(n, lpfilt(B2,y), n(3:end), diff(y,2),':');    % lower-right graph
```

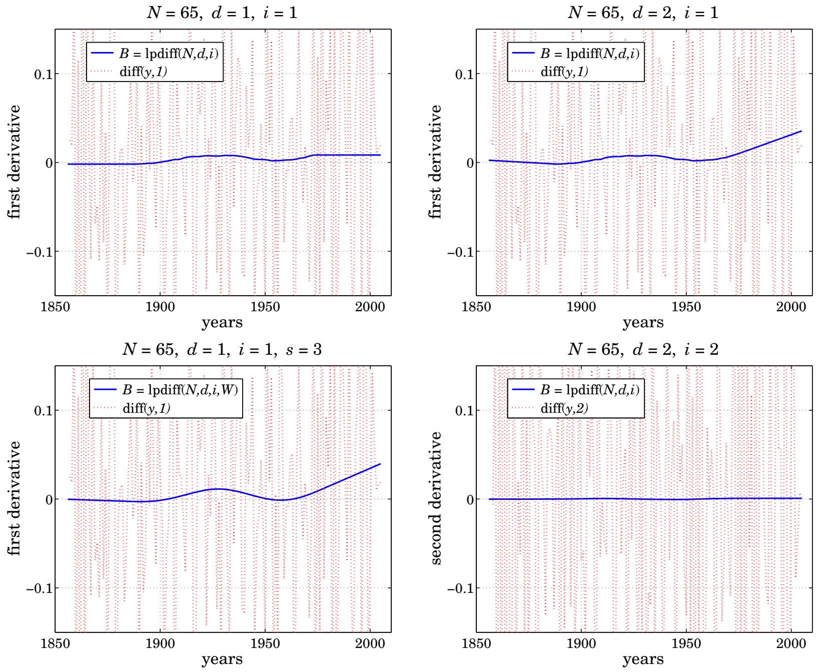


Fig. 3.9.5 Differentiated temperature signal.

The second derivative is essentially zero, being consistent with piecewise linear trends. Derivative signals can also be estimated for the SVD and Whittaker-Henderson methods. Since the outputs $\hat{x}_n$ of these methods are smooth signals, the corresponding derivatives

can be simply computed as the difference signals, $\text{diff}(\hat{x}_n)$, with comparable results as the local polynomial methods. □

4

Minimum Roughness Filters

4.1 Weighted Local Polynomial Filters

The design of the LPSM filters was based on a least-squares criterion, such as (3.2.2), where all error terms were equally weighted within the filter's window:

$$\mathcal{J}=\sum_{m=-M}^{M} e_m^2=\sum_{m=-M}^{M}(y_m-\hat{y}_m)^2=\sum_{m=-M}^{M}\left(y_m-\sum_{i=0}^{d}c_i m^i\right)^2=\min$$

This can be generalized by using unequal positive weights, w_m, $-M\le m\le M$:

$$\mathcal{J}=\sum_{m=-M}^{M} w_m e_m^2=\sum_{m=-M}^{M} w_m\left(y_m-\sum_{i=0}^{d}c_i m^i\right)^2=\min \tag{4.1.1}$$

Introducing the diagonal matrix $W=\mathrm{diag}([w_{-M},\dots,w_0,\dots,w_M])$, we may write Eq. (4.1.1) compactly as:

$$\mathcal{J}=\mathbf{e}^T W\mathbf{e}=(\mathbf{y}-S\mathbf{c})^T W(\mathbf{y}-S\mathbf{c})=\min \tag{4.1.2}$$

where $\mathbf{y}, S, \mathbf{c}$ have the same meaning as in Eqs. (3.2.26)–(3.2.30). Differentiating with respect to $\mathbf{c}$ gives the orthogonality and normal equations:

$$S^T W\mathbf{e}=S^T W(\mathbf{y}-S\mathbf{c})=0 \quad\Leftrightarrow\quad (S^T WS)\mathbf{c}=S^T W\mathbf{y} \tag{4.1.3}$$

with solution for $\mathbf{c}$ and the estimate $\hat{\mathbf{y}}=S\mathbf{c}$:

$$\begin{aligned}\mathbf{c}&=(S^T WS)^{-1}S^T W\mathbf{y}=G^T\mathbf{y}\\ \hat{\mathbf{y}}&=S\mathbf{c}=S(S^T WS)^{-1}S^T W\mathbf{y}=B^T\mathbf{y}\end{aligned} \tag{4.1.4}$$

where we defined

$$\boxed{\begin{aligned}G&=WS(S^T WS)^{-1}\\ B&=GS^T=WS(S^T WS)^{-1}S^T\end{aligned}} \tag{4.1.5}$$

The matrix B satisfies the following properties:

$$\begin{aligned}S^T B&=S^T\\ B^T&=W^{-1}BW\\ BWB^T&=BW=WB^T\end{aligned} \tag{4.1.6}$$

The first implies the usual polynomial-preserving moment constraints $S^T\mathbf{b}_m=\mathbf{u}_m$, for $-M\le m\le M$, where $\mathbf{b}_m$ is the mth column of B. The second shows that B is no longer symmetric, and the third may be used to simplify the minimized value of the performance index. Indeed, using the orthogonality property, we obtain:

$$\mathcal{J}_{\min}=\mathbf{e}^T W\mathbf{e}=\mathbf{y}^T W\mathbf{y}-\mathbf{y}^T BW\mathbf{y}-\mathbf{y}^T WB^T\mathbf{y}+\mathbf{y}^T BWB^T\mathbf{y}=\mathbf{y}^T W\mathbf{y}-\mathbf{y}^T BW\mathbf{y}$$

A fourth property follows if we assume that the weights w_m are symmetric about their middle, $w_m=w_{-m}$, or more generally if W is assumed to be positive-definite, symmetric, and centro-symmetric, which implies that it remains invariant under reversal of its rows and its columns. The centro-symmetric property can be stated concisely as $JW=WJ$, where J is the column-reversing matrix consisting of ones along its anti-diagonal, that is, the reverse of a column vector is $\mathbf{b}^R=J\mathbf{b}$. Under this assumption on W, it can be shown that B is also centro-symmetric:

$$JB=BJ \quad\Rightarrow\quad \mathbf{b}_m^R=\mathbf{b}_{-m}\,,\quad -M\le m\le M \tag{4.1.7}$$

This can be derived by noting that reversing the basis vector $\mathbf{s}_i$ simply multiplies it by the phase factor $(-1)^i$, so that $JS=S\Omega$, where Ω is the diagonal matrix of phase factors $(-1)^i$, $i=0,1,\dots,d$. This then implies Eq. (4.1.7). Similarly one can show that $JG=G\Omega$, so that the reverse of each differentiation filter is $\mathbf{g}_i^R=(-1)^i\mathbf{g}_i$.

The filtering equations (3.2.33) and (3.2.34) retain their form. Among the possible weighting matrices W, we are interested in those such that the polynomial fitting problem (4.1.2) has an equivalent characterization as the minimization of the NRR subject to the polynomial-preserving constraints $S^T\mathbf{b}_m=\mathbf{u}_m$. To this end, we consider the constrained minimization of a generalized or "prefiltered" NRR:

$$\boxed{\mathcal{R}=\mathbf{b}^T V\mathbf{b}=\min\,,\quad \text{subject to}\quad S^T\mathbf{b}=\mathbf{u}} \tag{4.1.8}$$

for a given $(d+1)$-dimensional vector $\mathbf{u}$. The $N\times N$ matrix V, where $N=2M+1$, is assumed to be strictly positive-definite, symmetric, and Toeplitz. We may write component-wise:

$$\mathcal{R}=\sum_{n,m=-M}^{M} b(n)V_{n-m}b(m)=\frac{1}{2\pi}\int_{-\pi}^{\pi}|B(\omega)|^2V(\omega)\,d\omega \tag{4.1.9}$$

where we set $V_{nm}=V_{n-m}$ because of the Toeplitz property, and introduced the corresponding DTFTs:

$$B(\omega)=\sum_{n=-M}^{M} b(n)e^{-j\omega n}\,,\quad V(\omega)=\sum_{k=-\infty}^{\infty} V_k e^{-j\omega k} \tag{4.1.10}$$

One way to guarantee a positive-definite V is to take $V(\omega)$ to be the power spectrum of a given filter, say, $D(\omega)$, that is, choose $V(\omega)=|D(\omega)|^2$, so that $\mathcal{R}$ will be the ordinary NRR of the cascaded filter $F(\omega)=D(\omega)B(\omega)$ or $F(z)=D(z)B(z)$:

$$\mathcal{R}=\frac{1}{2\pi}\int_{-\pi}^{\pi}|B(\omega)|^2V(\omega)d\omega=\frac{1}{2\pi}\int_{-\pi}^{\pi}|B(\omega)D(\omega)|^2d\omega \tag{4.1.11}$$

The minimum-R_s or minimum-roughness filters discussed in Sec. 4.2 correspond to the choice $D(z)=(1-z^{-1})^s$, for some integer s. For a general V and $\mathbf{u}$, the solution of the problem (4.1.8) is obtained by introducing a Lagrange multiplier vector $\boldsymbol{\lambda}$:

$$\mathcal{J}=\mathbf{b}^TV\mathbf{b}+2\boldsymbol{\lambda}^T(\mathbf{u}-S^T\mathbf{b})=\min$$

leading to the solution:

$$\begin{aligned}\boldsymbol{\lambda}&=(S^TV^{-1}S)^{-1}\mathbf{u}\\ \mathbf{b}&=V^{-1}S\boldsymbol{\lambda}=V^{-1}S(S^TV^{-1}S)^{-1}\mathbf{u}\end{aligned} \tag{4.1.12}$$

If we choose $\mathbf{u}_m=[1,m,m^2,\dots,m^d]^T$ as the constraint vectors and put together the resulting solutions as the columns of a matrix B, then,

$$B=[\cdots\mathbf{b}_m\cdots]=V^{-1}S(S^TV^{-1}S)^{-1}[\cdots\mathbf{u}_m\cdots]$$

or, because $S^T=[\cdots\mathbf{u}_m\cdots]$,

$$\boxed{B=V^{-1}S(S^TV^{-1}S)^{-1}S^T} \tag{4.1.13}$$

This solution appears to be different from the solution (4.1.5) of the least-squares problem, $B=WS(S^TWS)^{-1}S^T$. Can the two solutions be the same? The trivial choice $V=W=I$ corresponds to the LPSM filters. The choice $V=W^{-1}$ is not acceptable because with V assumed Toeplitz, and W assumed diagonal, it would imply that all the weights are equal, which is again the LPSM case. A condition that guarantees the equivalence is the following [123,99]:

$$VWS=SC\quad\Rightarrow\quad WS=V^{-1}SC \tag{4.1.14}$$

where C is an invertible $(d+1)\times(d+1)$ matrix. Indeed, then $S^TWS=S^TV^{-1}SC$, and,

$$G=WS(S^TWS)^{-1}=V^{-1}S(S^TV^{-1}S)^{-1} \tag{4.1.15}$$

so that

$$B=WS(S^TWS)^{-1}S^T=V^{-1}S(S^TV^{-1}S)^{-1}S^T \tag{4.1.16}$$

For the minimum-R_s filters, the particular choices for W,V do indeed satisfy condition (4.1.14) with an *upper-triangular* matrix C. With the equivalence of the polynomial-fitting and minimum-NRR approaches at hand, we can also derive the corresponding predictive/interpolating differentiation filters. Choosing $\mathbf{u}=\mathcal{D}^i\mathbf{u}_t$ as the constraint vector in (4.1.12), we obtain,

$$\mathbf{b}_t^{(i)}=V^{-1}S(S^TV^{-1}S)^{-1}\mathcal{D}^i\mathbf{u}_t=WS(S^TWS)^{-1}\mathcal{D}^i\mathbf{u}_t \tag{4.1.17}$$

and at the sample values $t=m$, $-M\le m\le M$, or, at $\mathbf{u}_t=\mathbf{u}_m$, we obtain the differentiation matrix having the $\mathbf{b}_m^{(i)}$ as columns, $B^{(i)}=[\cdots\mathbf{b}_m^{(i)}\cdots]$:

$$B^{(i)}=WS(S^TWS)^{-1}\mathcal{D}^iS^T=V^{-1}S(S^TV^{-1}S)^{-1}\mathcal{D}^iS^T \tag{4.1.18}$$

Computationally, it is best to orthogonalize the basis S. Let $W=U^TU$ be the Cholesky factorization of the positive-definite symmetric matrix W, where U is an $N\times N$ upper-triangular factor. Then, performing the QR-factorization on the $N\times(d+1)$ matrix US, the above computations become:

$$\begin{aligned}W&=U^TU\\ US&=Q_0R_0\,,\quad\text{with }\ Q_0^TQ_0=I\,,\quad R_0=(d+1)\times(d+1)\ \text{upper-triangular}\\ B&=U^TQ_0Q_0^TU^{-T}\\ B^{(i)}&=U^TQ_0(R_0^{-T}\mathcal{D}^iR_0^T)Q_0^TU^{-T}\\ \mathbf{b}_t^{(i)}&=U^TQ_0R_0^{-T}\mathcal{D}^i\mathbf{u}_t\end{aligned} \tag{4.1.19}$$

The MATLAB functions `lpsm`, `lpdiff`, `lpinterp` have the weighting matrix W as an additional input, which if omitted defaults to $W=I$. They implement Eqs. (4.1.19) and their full usage is:

```
[B,G] = lpsm(N,d,W);
B = lpdiff(N,d,i,W);
b = lpinterp(N,d,t,i,W);
```

The factorizations in Eq. (4.1.19) lead naturally to a related implementation in terms of discrete polynomials that are orthogonal with respect to the weighted inner product:

$$\mathbf{a}^TW\mathbf{b}=\sum_{m=-M}^{M}w_m\,a(m)b(m) \tag{4.1.20}$$

Such polynomials may be constructed from the monomials $s_i(m)=m^i$, $i=0,1,\dots,d$ via Gram-Schmidt orthogonalization applied with respect to the above inner product. The result of orthogonalizing the basis $S=[\mathbf{s}_0,\mathbf{s}_1,\dots,\mathbf{s}_d]$ is $Q=[\mathbf{q}_0,\mathbf{q}_1,\dots,\mathbf{q}_d]$ whose columns $q_i(m)$ are polynomials of order i in the variable m that are mutually orthogonal, that is, up to an overall normalization:

$$\mathbf{q}_i^TW\mathbf{q}_j=\delta_{ij}D_i\,,\quad i,j=0,1,\dots,d\quad\Rightarrow\quad Q^TWQ=D \tag{4.1.21}$$

where $D=\text{diag}([D_0,D_1,\dots,D_d])$ is the diagonal matrix of the (positive) normalization factors D_i. These factors can be selected to be unity if so desired. For the minimum-roughness filters, these polynomials are special cases of the Hahn orthogonal polynomials, whose properties are discussed in Sec. 4.3. For unity weights $w_m=1$, the polynomials reduce to the discrete Chebyshev/Gram polynomials.

Numerically, these polynomials can be constructed from the factorization (4.1.19). Since D is positive-definite, we may define $D^{1/2}=\text{diag}([D_0^{1/2},D_1^{1/2},\dots,D_d^{1/2}])$ to be its square root. Then we construct Q,R in terms of the factors U,Q_0,R_0:

$$Q=U^{-1}Q_0D^{1/2}\,,\quad R=D^{-1/2}R_0 \tag{4.1.22}$$

where R is still upper-triangular. Then, we have $Q^T W Q = D$ and

$$QR = U^{-1}Q_0 D^{1/2} D^{-1/2} R_0 = U^{-1} Q_0 R_0 = U^{-1} U S = S$$

which is equivalent to the Gram-Schmidt orthogonalization of the basis S, and leads to the following equivalent representation of Eq. (4.1.19):

$$\begin{aligned} S &= QR\,, \quad \text{with } Q^T W Q = D\,, \quad R = (d+1)\times(d+1) \text{ upper-triangular} \\ B &= WQD^{-1}Q^T \\ B^{(i)} &= WQD^{-1}(R^{-T}\mathcal{D}^i R^T)Q^T \\ \mathbf{b}_t^{(i)} &= WQD^{-1}R^{-T}\mathcal{D}^i \mathbf{u}_t \end{aligned} \tag{4.1.23}$$

Since $Q = [\mathbf{q}_0, \mathbf{q}_1, \dots, \mathbf{q}_d]$, the matrix B can be expressed as,

$$B = WQD^{-1}Q^T = W\sum_{r=0}^{d} D_r^{-1}\mathbf{q}_r\mathbf{q}_r^T \tag{4.1.24}$$

and for diagonal W, we have component-wise:

$$\boxed{b_m(k) = B_{km} = w_k \sum_{r=0}^{d} \frac{q_r(k)q_r(m)}{D_r}} \quad -M \le m, k \le M \tag{4.1.25}$$

The sum in (4.1.25) can be simplified further using the Christoffel-Darboux identity discussed in Sec. 4.3. The polynomial predictive interpolation filters $\mathbf{b}_t^{(i)}$ can also be expressed in a similar summation form:

$$\boxed{b_t^{(i)}(k) = w_k \sum_{r=0}^{d} \frac{q_r(k)q_r^{(i)}(t)}{D_r}} \tag{4.1.26}$$

where $q_r^{(i)}(t)$ is the ith derivative of the polynomial $q_r(t)$ obtained from $q_r(m)$ by replacing the discrete variable m by t. This can be justified as follows. The mth rows of the matrices S and Q are the $(d+1)$-dimensional vectors:

$$\begin{aligned} \mathbf{u}_m^T &= [s_0(m), s_1(m), \dots, s_d(m)] = [1, m, \dots, m^d] \\ \mathbf{p}_m^T &= [q_0(m), q_1(m), \dots, q_d(m)] \end{aligned} \tag{4.1.27}$$

and since $S = QR$, they are related by $\mathbf{u}_m^T = \mathbf{p}_m^T R$. Replacing m by t preserves this relationship, so that $\mathbf{u}_t^T = \mathbf{p}_t^T R$, or,

$$\mathbf{u}_t = R^T \mathbf{p}_t\,, \quad \text{where } \mathbf{p}_t = [q_0(t), q_1(t), \dots, q_d(t)]^T \tag{4.1.28}$$

Differentiating i times, we obtain

$$\mathcal{D}^i \mathbf{u}_t = \mathbf{u}_t^{(i)} = R^T \mathbf{p}_t^{(i)} \quad \Rightarrow \quad \mathbf{p}_t^{(i)} = R^{-T}\mathcal{D}^i \mathbf{u}_t \tag{4.1.29}$$

and therefore $\mathbf{b}_t^{(i)}$ from Eq. (4.1.23) can be written in the following form, which implies Eq. (4.1.26):

$$\boxed{\mathbf{b}_t^{(i)} = WQD^{-1}\mathbf{p}_t^{(i)}} \tag{4.1.30}$$

As in the case of the LPSM filters, for the special case $d = N-1$, the interpolation filters correspond to Lagrange interpolation. In this case Q becomes an invertible $N\times N$ matrix satisfying the weighted unitarity property $Q^T W Q = D$, which implies

$$Q^{-1} = D^{-1}Q^T W \tag{4.1.31}$$

from which we obtain the *completeness* property:

$$\boxed{QD^{-1}Q^T = W^{-1}} \tag{4.1.32}$$

which shows that $B = I$. Similarly, using $WQD^{-1} = Q^{-T}$, we obtain from (4.1.23) the usual Lagrange interpolation polynomials:

$$\mathbf{b}_t = WQD^{-1}R^{-T}\mathbf{u}_t = Q^{-T}R^{-T}\mathbf{u}_t = S^{-T}\mathbf{u}_t \tag{4.1.33}$$

With $d = N-1$, the matrix Q is an orthogonal basis for the full space $\mathbb{R}^N$. One of the applications of Eq. (4.1.31) is the representation of signals, such as images or speech in terms of *orthogonal-polynomial moments* [137–150].

Given an N-dimensional signal block $\mathbf{y}$, such as a row in a scanned image, we define the N-dimensional vector of moments with respect to the polynomials Q,

$$\boldsymbol{\mu} = D^{-1}Q^T W\mathbf{y} \quad \Rightarrow \quad \mu_r = \frac{1}{D_r}\sum_{n=-M}^{M} q_r(n)w_n y_n\,, \quad r = 0, 1, \dots, N-1 \tag{4.1.34}$$

Because of Eq. (4.1.31), we have $\boldsymbol{\mu} = Q^{-1}\mathbf{y}$, which allows the reconstruction of $\mathbf{y}$ from its moments:

$$\mathbf{y} = Q\boldsymbol{\mu} \quad \Rightarrow \quad y_n = \sum_{r=0}^{N-1} q_r(n)\mu_r\,, \quad -M \le n \le M \tag{4.1.35}$$

4.2 Henderson Filters

All the results of the previous section find a concrete realization in the minimum-R_s filters that we discuss here. Consider the order-s backward difference filter and its impulse response defined by:

$$D_s(z) = (1 - z^{-1})^s \quad \Leftrightarrow \quad d_s(k) = (-1)^k \binom{s}{k}\,, \quad k = 0, 1, \dots, s \tag{4.2.1}$$

This follows from the binomial expansion:

$$(1 - z^{-1})^s = \sum_{k=0}^{s} (-1)^k \binom{s}{k} z^{-k} \tag{4.2.2}$$

The operation of the filter $D_s(z)$ on a signal f_n, with output g_n, is usually denoted in terms of the backward difference operator $\nabla f_n = f_n - f_{n-1}$ as follows:

$$g_n = \nabla^s f_n = \sum_{k=0}^{s} d_s(k) f_{n-k} = \sum_{k=0}^{s} (-1)^k \binom{s}{k} f_{n-k} \tag{4.2.3}$$

If the signal f_n is restricted over the range $-M \le n \le M$, then because $0 \le k \le s$ and $-M \le n - k \le M$, the above equation can be written in the more precise form:

$$g_n = \nabla^s f_n = \sum_{k=\max(0,n-M)}^{\min(s,n+M)} (-1)^k \binom{s}{k} f_{n-k}\,, \quad -M \le n \le M + s \tag{4.2.4}$$

Eq. (4.2.4) gives the full convolutional output $g_n = (d_s * f)_n$, while (4.2.3) is the corresponding steady-state output, obtained by restricting the output index n to the range $-M + s \le n \le M$. Defining the $(N+s)$-dimensional output vector $\mathbf{g}$ and N-dimensional input vector $\mathbf{f}$, where $N = 2M + 1$,

$$\mathbf{g} = [g_{-M}, \dots, g_M, \dots, g_{M+s}]^T\,, \quad \mathbf{f} = [f_{-M}, \dots, f_M]^T\,,$$

we may write the full filtering equation (4.2.4) in matrix form:

$$\mathbf{g} = D_s \mathbf{f} \tag{4.2.5}$$

where D_s is the full $(N+s)\times N$ convolutional matrix of the filter $d_s(k)$ defined by its matrix elements:

$$(D_s)_{nm} = d_s(n-m)\,, \quad -M \le n \le M + s, \quad -M \le m \le M \tag{4.2.6}$$

and subject to the restriction that only the values $0 \le n - m \le s$ will result in a non-zero matrix element. The MATLAB functions binom and diffmat allow the calculation of the binomial coefficients $d_s(k)$ and the convolution matrix D_s:

```
d = binom(s,k);      % binomial coefficients d_s(k)
D = diffmat(s,N);    % (N+s)×N difference convolution matrix
```

For example, the convolution matrix for $N = 7$ and $s = 3$ is:

$$D_3 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -3 & 1 & 0 & 0 & 0 & 0 & 0 \\ 3 & -3 & 1 & 0 & 0 & 0 & 0 \\ -1 & 3 & -3 & 1 & 0 & 0 & 0 \\ 0 & -1 & 3 & -3 & 1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -3 & 1 & 0 \\ 0 & 0 & 0 & -1 & 3 & -3 & 1 \\ 0 & 0 & 0 & 0 & -1 & 3 & -3 \\ 0 & 0 & 0 & 0 & 0 & -1 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix}$$

The function diffmat is simply a call to convmat:

```
D = convmat(binom(s),N);
```

A minimum-R_s filter $B(z)$ is defined to minimize the NRR of the cascaded filter $F(z) = D_s(z)B(z)$ subject to the $d+1$ linear constraints $S^T\mathbf{b} = \mathbf{u}$, for a given constraint vector $\mathbf{u}$, where $\mathbf{b}$ denotes the impulse response of $B(z)$ assumed to be double-sided, that is, b_n, $-M \le n \le M$.

The actual smoothing of data is carried out by the filter $B(z)$ itself, whereas the filter $F(z)$ is used to design $B(z)$. This is depicted in Fig. 4.2.1 in which the filtered output is $\hat{x}_n$, and the output of $F(z)$ is the differenced signal $\nabla^s \hat{x}_n$ whose mean-square value may be taken as a measure of smoothness to be minimized.

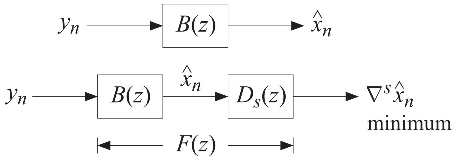


Fig. 4.2.1 Design and smoothing by minimum-R_s filter.

Letting $f_n = \nabla^s b_n$ be the impulse response of the filter $F(z)$, or in matrix form $\mathbf{f} = D_s\mathbf{b}$, the corresponding cascaded NRR will be:

$$\mathcal{R}_s = \mathbf{f}^T\mathbf{f} = \sum_{n=-M}^{M+s} f_n^2 = \sum_{n=-M}^{M+s} (\nabla^s b_n)^2 = \frac{1}{2\pi}\int_{-\pi}^{\pi} |D_s(\omega)B(\omega)|^2\,d\omega$$

Since $\mathbf{f}^T\mathbf{f} = \mathbf{b}^T(D_s^T D_s)\mathbf{b}$, we can state the design condition of the minimum-R_s filters as

$$\boxed{\mathcal{R}_s = \sum_{n=-M}^{M+s} (\nabla^s b_n)^2 = \mathbf{b}^T(D_s^T D_s)\mathbf{b} = \min\,, \quad \text{subject to } S^T\mathbf{b} = \mathbf{u}} \tag{4.2.7}$$

This has exactly the same form as Eq. (4.1.8) with $V = D_s^T D_s$. The minimization of $\mathcal{R}_s$ justifies the name “minimum-R_s ” filters. The minimum-R_0 LPSM filters of Sec. 3.7 correspond to $s = 0$. In the actuarial literature, the following criterion is used instead, which differs from $\mathcal{R}_s$ by a normalization factor:

$$R_s = \frac{\mathbf{b}^T(D_s^T D_s)\mathbf{b}}{\mathbf{d}_s^T\mathbf{d}_s} = \frac{\mathcal{R}_s}{\mathbf{d}_s^T\mathbf{d}_s} = \min \tag{4.2.8}$$

where R_s is referred as the “smoothing coefficient”, $\mathbf{d}_s$ is the impulse response vector of the filter $D_s(z)$, and $\mathbf{d}_s^T\mathbf{d}_s$ is the NRR of $D_s(z)$. Using a binomial identity (a special case of (4.2.13) for $k = 0$), we have,

$$\mathbf{d}_s^T\mathbf{d}_s = \sum_{m=0}^{s} d_s^2(m) = \sum_{m=0}^{s} \binom{s}{m}^2 = \binom{2s}{s} \tag{4.2.9}$$

The criterion (4.2.7) provides a measure of smoothness. To see this, let $\hat{x}_n$ be the result of filtering an arbitrary stationary signal y_n through the filter $B(z)$. If $S_{yy}(\omega)$ is the power spectrum of y_n , then the power spectra of the filtered output $\hat{x}_n$ and of the differenced output $\nabla^s\hat{x}_n$ will be $|B(\omega)|^2S_{yy}(\omega)$ and $|D_s(\omega)B(\omega)|^2S_{yy}(\omega)$, respectively. Therefore, the mean-square value of $\nabla^s\hat{x}_n$ will be:

$$E\big[(\nabla^s\hat{x}_n)^2\big] = \frac{1}{2\pi}\int_{-\pi}^{\pi}|D_s(\omega)B(\omega)|^2S_{yy}(\omega)\,d\omega \tag{4.2.10}$$

If y_n is white noise of variance σ^2, or if we assume that $S_{yy}(\omega)$ is bounded from above by a constant, such as $S_{yy}(\omega)\le\sigma^2$, then we obtain:

$$E\big[(\nabla^s\hat{x}_n)^2\big] \le \frac{1}{2\pi}\int_{-\pi}^{\pi}|D_s(\omega)B(\omega)|^2\sigma^2\,d\omega = \mathcal{R}_s\sigma^2 \tag{4.2.11}$$

For white noise, $S_{yy}(\omega)=\sigma^2$, Eq. (4.2.11) becomes an equality. Thus, minimizing $\mathcal{R}_s$ will minimize $E\big[(\nabla^s\hat{x}_n)^2\big]$ and tend to result in a smoother filtered signal $\hat{x}_n$. This property justifies the term “minimum-roughness” filters.

The choice $s=2$ is preferred in smoothing financial and business-cycle data, and is used also by the related method of the Whittaker-Henderson or Hodrick-Prescott filter. The choice $s=3$ is standard in the actuarial literature. The choice $s=4$ is not common but it was used by De Forest [65–68] who was the first to formulate and solve the minimum-R_s problem in 1871. Others, like Hardy and Henderson have considered the minimum-R_3 problem, while Sheppard [76] solved the minimum-R_s problem in general.

Henderson [79] was the first to show the *equivalence* between the NRR minimization problem (4.2.7) with $V=D_s^TD_s$ and the weighted least-squares polynomial fitting problem (4.1.1) using the so-called Henderson weights w_m. Therefore, the minimum-R_s filters are often referred to as *Henderson filters*. They are used widely in seasonal-adjustment, census, and business-cycle extraction applications. We discuss this equivalence next, following essentially Henderson’s method.

The elements of the $N{\times}N$ matrix $V=D_s^TD_s$ are $(D_s^TD_s)_{nm}=V_{nm}=V_{n-m}$, where V_k is the autocorrelation function of the power spectrum $V(\omega)=|D_s(\omega)|^2$. Working in the z-domain, we have the spectral density:

$$V(z)=D_s(z)D_s(z^{-1})=(1-z^{-1})^s(1-z)^s=(-1)^sz^s(1-z^{-1})^{2s} \tag{4.2.12}$$

which shows that $V(z)$ effectively acts as the $(2s)$-difference operation ∇^{2s}. Taking inverse z-transforms of both sides of (4.2.12), we obtain:

$$V_k=\sum_{m=\max(0,k)}^{\min(s,k+s)}d_s(m)d_s(m-k)=(-1)^sd_{2s}(k+s)\,,\quad -s\le k\le s \tag{4.2.13}$$

or, explicitly in terms of the definition of d_s:

$$V_k=(-1)^k\sum_{m=\max(0,k)}^{\min(s,k+s)}\binom{s}{m}\binom{s}{m-k}=(-1)^k\binom{2s}{s+k},\quad -s\le k\le s \tag{4.2.14}$$

or,

$$V_k=(-1)^k\frac{(2s)!}{(s+k)!\,(s-k)!}\,,\quad -s\le k\le s \tag{4.2.15}$$

The V matrix is a banded Toeplitz matrix with bandwidth $\pm s$, whose central row or central column consist of the numbers V_k, $-s\le k\le s$, with V_0 positioned at the center of the matrix. As an example,

$$V=D_3^TD_3=\begin{bmatrix} 20 & -15 & 6 & -1 & 0 & 0 & 0\\ -15 & 20 & -15 & 6 & -1 & 0 & 0\\ 6 & -15 & 20 & -15 & 6 & -1 & 0\\ -1 & 6 & -15 & 20 & -15 & 6 & -1\\ 0 & -1 & 6 & -15 & 20 & -15 & 6\\ 0 & 0 & -1 & 6 & -15 & 20 & -15\\ 0 & 0 & 0 & -1 & 6 & -15 & 20 \end{bmatrix}$$

with central column or central row:

$$V_k=\{-1,\ 6,\ -15,\ 20,\ -15,\ 6,\ -1\}\quad\text{for}\quad k=\{-2,-1,0,1,2\}$$

To understand the action of V as the difference operator ∇^{2s}, let $\mathbf{f}$ be an N dimensional vector indexed for $-M\le m\le M$, and form the output N-dimensional vector:

$$\mathbf{g}=V\mathbf{f}\quad\Rightarrow\quad g_n=\sum_{m=-M}^{M}V_{n-m}f_m\,,\quad -M\le n\le M \tag{4.2.16}$$

where $n-m$ is further restricted such that $-s\le n-m\le s$. Next, consider an extended version of $\mathbf{f}$ obtained by padding s zeros in front and s zeros at the end, so that the extended vector $\mathbf{f}^{\text{ext}}$ will be indexed over, $-(M+s)\le m\le(M+s)$:

$$\mathbf{f}^{\text{ext}}=[\underbrace{0,\dots,0}_{s},f_{-M},\dots,f_0,\dots,f_M,\underbrace{0,\dots,0}_{s}]^T$$

Then, the summation in Eq. (4.2.16) can be extended as,

$$g_n=\sum_{m=-M-s}^{M+s}V_{n-m}f_m^{\text{ext}}\,,\quad -M\le n\le M \tag{4.2.17}$$

But because of the restriction $-s\le n-m\le s$, the above summation can be restricted to be over $n-s\le m\le n+s$, which is a subrange of the range $-(M+s)\le m\le(M+s)$ because we assumed $-M\le n\le M$. Thus, we may write:

$$g_n=\sum_{m=-n-s}^{n+s}V_{n-m}f_m^{\text{ext}}\,,\quad -M\le n\le M$$

or, changing to $k=n-m$,

$$g_n=\sum_{k=-s}^{s}V_kf_{n-k}^{\text{ext}}=(-1)^s\sum_{k=-s}^{s}d_{2s}(s+k)f_{n-k}^{\text{ext}}=(-1)^s\sum_{i=0}^{2s}d_{2s}(i)f_{n+s-i}^{\text{ext}} \tag{4.2.18}$$

but that is precisely the ∇^{2s} operator:

$$g_n=(-1)^s\nabla^{2s}f_{n+s}^{\text{ext}}\,,\quad -M\le n\le M \tag{4.2.19}$$

If f_m^{ext} is a polynomial of degree $(2s+i)$, then the $(2s)$-differencing operation will result into a polynomial of degree i. Suppose that we start with the weighted monomial:

$$f_m = w_m m^i\,, \quad -M \le m \le M \tag{4.2.20}$$

where the weighting function w_m is itself a polynomial of degree $2s$, then in order for the extended vector f_m^{ext} to vanish over $M < |m| \le M+s$, the function w_m must have zeros at these points, that is,

$$w_m = 0\,, \quad \text{for } \; m = \pm(M+1), \pm(M+2), \dots, \pm(M+s)$$

This condition fixes w_m uniquely, up to a normalization constant:

$$\boxed{w_m = \prod_{i=1}^{s}\big[(M+i)^2 - m^2\big]} \qquad \text{(Henderson weights)} \tag{4.2.21}$$

These are called *Henderson weights.* Because the extended signal f_m^{ext} is a polynomial of degree $(2s+i)$, it follows that the signal g_n will be a polynomial of degree i.

Defining the $N\times N$ diagonal matrix $W = \text{diag}\big([w_{-M}, \dots, w_0, \dots, w_M]\big)$, we can write (4.2.20) vectorially in terms of the monomial basis vector $\mathbf{s}_i$ as $\mathbf{f} = W\mathbf{s}_i$. We showed that the matrix operation $\mathbf{g} = V\mathbf{f} = VW\mathbf{s}_i$ results into a polynomial of degree i, which therefore can be expanded as a linear combination of the monomials $\mathbf{s}_0, \mathbf{s}_1, \dots, \mathbf{s}_i$ up to order i, that is,

$$VW\mathbf{s}_i = \sum_{j=0}^{i} \mathbf{s}_j C_{ji} \tag{4.2.22}$$

for appropriate coefficients C_{ji}, which may thought of as the matrix elements of an upper-triangular matrix. Applying this result to each basis vector of $S = [\mathbf{s}_0, \mathbf{s}_1, \dots, \mathbf{s}_d]$ up to order d, it follows that

$$VWS = SC\,, \quad C = (d+1)\times(d+1) \text{ upper-triangular} \tag{4.2.23}$$

But, this is exactly the condition (4.1.14). Thus, we have shown the equivalence of the NRR minimization problem (4.2.7) with $V = D_s^T D_s$ and the weighted least-squares polynomial fitting problem (4.1.1) with the Henderson weights w_m. The rest of the results of Sec. 4.1 then carry through unchanged.

The MATLAB function `lprs` implements the design. It constructs the W matrix from the Henderson weights and passes it into the function `lpsm`:

```
[B,G] = lprs(N,d,s);      % local polynomial minimum-Rs filters
```

The Henderson weights w_m, $-M \le m \le M$ are calculated by the function `hend`:

```
w = hend(N,s);      % Henderson weights
```

In the next section, we derive closed-form expressions for the Henderson filters using Hahn orthogonal polynomials. Analytical expressions can also be derived working with the non-orthogonal monomial basis S. It follows from $B = WS(S^TWS)^{-1}S^T$ that the kth component of the mth filter will be:

$$b_m(k) = B_{km} = w_k \sum_{i,j=0}^{d} k^i m^j \Phi_{ij} = w_k\,\mathbf{u}_k^T \Phi \mathbf{u}_m \tag{4.2.24}$$

where $\mathbf{u}_k = [1, k, k^2, \dots, k^d]^T$ and Φ is the inverse of the Hankel matrix $F = S^TWS$ whose matrix elements are the weighted inner products:

$$F_{ij} = (S^TWS)_{ij} = \mathbf{s}_i^T W \mathbf{s}_j = \sum_{m=-M}^{M} w_m m^{i+j} \equiv F_{i+j}\,, \quad i,j = 0,1,\dots,d \tag{4.2.25}$$

Except for the factor w_k and the different values of Φ_{ij} the expressions are similar to those of the LPSM filters of Sec. 3.3. The matrix Φ has a similar checkerboard structure. For example, we have for the commonly used case $d = 3$ and $s = 3$:

$$b_m(k) = w_k\,[1,\; k,\; k^2,\; k^3]\begin{bmatrix} \Phi_{00} & 0 & \Phi_{02} & 0 \\ 0 & \Phi_{11} & 0 & \Phi_{13} \\ \Phi_{20} & 0 & \Phi_{22} & 0 \\ 0 & \Phi_{31} & 0 & \Phi_{33} \end{bmatrix}\begin{bmatrix} 1 \\ m \\ m^2 \\ m^3 \end{bmatrix} \tag{4.2.26}$$

where

$$w_k = \big[(M+1)^2 - k^2\big]\big[(M+2)^2 - k^2\big]\big[(M+3)^2 - k^2\big] \tag{4.2.27}$$

and

$$F = \begin{bmatrix} F_0 & 0 & F_2 & 0 \\ 0 & F_2 & 0 & F_4 \\ F_2 & 0 & F_4 & 0 \\ 0 & F_4 & 0 & F_6 \end{bmatrix} \quad\Rightarrow\quad \Phi = F^{-1} = \begin{bmatrix} \Phi_{00} & 0 & \Phi_{02} & 0 \\ 0 & \Phi_{11} & 0 & \Phi_{13} \\ \Phi_{20} & 0 & \Phi_{22} & 0 \\ 0 & \Phi_{31} & 0 & \Phi_{33} \end{bmatrix}$$

where we obtain from the checkerboard submatrices:

$$\begin{bmatrix} \Phi_{00} & \Phi_{02} \\ \Phi_{20} & \Phi_{22} \end{bmatrix} = \begin{bmatrix} F_0 & F_2 \\ F_2 & F_4 \end{bmatrix}^{-1}, \quad \begin{bmatrix} \Phi_{11} & \Phi_{13} \\ \Phi_{31} & \Phi_{33} \end{bmatrix} = \begin{bmatrix} F_2 & F_4 \\ F_4 & F_6 \end{bmatrix}^{-1} \tag{4.2.28}$$

The corresponding F-factors for $s = 3$ are:

$$\begin{aligned} F_0 &= \frac{2}{35}(2M+7)(2M+5)(2M+3)(2M+1)(M+3)(M+2)(M+1) \\ F_2 &= \frac{1}{9}M(M+4)F_0 \\ F_4 &= \frac{1}{11}(3M^2+12M-4)F_2 \\ F_6 &= \frac{1}{143}(15M^4+120M^3+180M^2-240M+68)F_2 \end{aligned}$$

which give rise to the matrix elements of Φ:

$$
\begin{aligned}
\Phi_{00} &= 315\,(3M^2 + 12M - 4)\,/D_1 \\
\Phi_{02} &= -3465/D_1 \\
\Phi_{22} &= 31185/D_1 \\
\Phi_{11} &= 1155\,(15M^4 + 120M^3 + 180M^2 - 240M + 68)\,/D_2 \\
\Phi_{13} &= -15015\,(3M^2 + 12M - 4)\,/D_2 \\
\Phi_{33} &= 165165/D_2
\end{aligned}
$$

with the denominator factors:

$$
\begin{aligned}
D_1 &= 8\,(2M+9)\,(2M+7)\,(2M+5)\,(2M+3)\,(M+3)\,(M+2)\,(M+1)\,(4M^2-1) \\
D_2 &= 8M\,(M-1)\,(M+4)\,(M+5)\,D_1
\end{aligned}
$$

In particular, setting $m = 0$ we find the central filter $b_0(k)$, which for the case $d = 3$ and $s = 3$, is referred to as “Henderson’s ideal formula:”

$$b_0(k) = w_k\,(\Phi_{00} + k^2\Phi_{02})$$

or, with $w_k = \big[(M+1)^2 - k^2\big]\big[(M+2)^2 - k^2\big]\big[(M+3)^2 - k^2\big]$:

$$b_0(k) = \frac{315\,(3M^2 + 12M - 4 - 11k^2)\,w_k}{8\,(2M+9)\,(2M+7)\,(2M+5)\,(2M+3)\,(M+3)\,(M+2)\,(M+1)\,(4M^2-1)} \tag{4.2.29}$$

The corresponding predictive/interpolating differentiation filters $b_t^{(i)}(k)$ are given by a similar expression:

$$b_t^{(i)}(k) = w_k \mathbf{u}_k^T \Phi \mathcal{D}^i \mathbf{u}_t \tag{4.2.30}$$

or, explicitly, for the $d = s = 3$ case and differentiation order $i = 0, 1, 2, 3$:

$$b_t^{(i)}(k) = w_k\,[1,\ k,\ k^2,\ k^3]\begin{bmatrix}\Phi_{00} & 0 & \Phi_{02} & 0 \\ 0 & \Phi_{11} & 0 & \Phi_{13} \\ \Phi_{20} & 0 & \Phi_{22} & 0 \\ 0 & \Phi_{31} & 0 & \Phi_{33}\end{bmatrix}\begin{bmatrix}0&0&0&0\\1&0&0&0\\0&2&0&0\\0&0&3&0\end{bmatrix}^i\begin{bmatrix}1\\t\\t^2\\t^3\end{bmatrix} \tag{4.2.31}$$

Example 4.2.1: *USD/Euro exchange rate.* Consider four methods of smoothing the USD/Euro foreign exchange rate for the years 1999-08. The monthly data are available from the web site: `http://research.stlouisfed.org/fred2/series/EXUSEU`

The upper-left graph in Fig. 4.2.2 shows the smoothing by an LPSM filter of length $N = 19$ and polynomial order $d = 3$. In the upper-right graph a minimum-R_s Henderson filter was used with $N = 19$, $d = 3$, and smoothness order $s = 3$.

The middle-left graph uses the SVD signal enhancement method with embedding order $M = 8$ and rank $r = 2$.

The middle-right graph uses the Whittaker-Henderson, or Hodrick-Prescott filter with smoothing parameter $\lambda = 100$ and smoothness order $s = 3$.

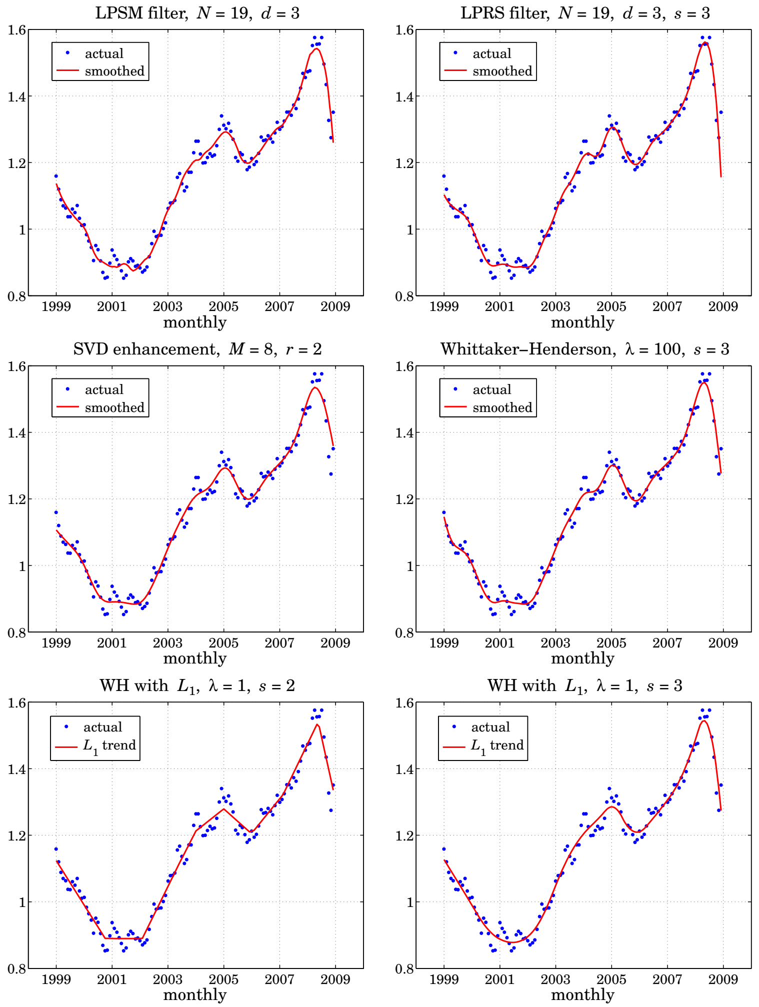


Fig. 4.2.2 Smoothing of USD/Euro exchange rate.

The lower left and right graphs use the Whittaker-Henderson regularization filter with the L_1 criterion with differentiation orders $s = 2$ and $s = 3$ and smoothing parameter $\lambda = 1$, implemented with the CVX package.[†]. The $s = 2$ case represents the smoothed signal in piece-wise linear form. The L_1 case is discussed further in Sec. 8.7.

The following MATLAB code illustrates the generation of the four graphs:

```
    Y = loadfile('exuseu.dat');                  % data file available in the OSP toolbox
```

[†] `http://cvxr.com/cvx/`

```
y = Y(:,4); t = taxis(y,12,1999);       % extract signal y_n from data file
                                        % the function taxis defines the t-axis
N=19; d=3; x1 = lpfilt(lpsm(N,d),y);    % LPSM filter
s=3; x2 = lpfilt(lprs(N,d,s),y);        % LPRS filter
M=8; r=2; x3 = svdenh(y,M,r);           % SVD enhancement
la=100; s=3; x4 = whsm(y,la,s);         % Whittaker-Henderson

s = 2; la = 1; N = length(y);           % Whittaker-Henderson with L_1 criterion
D = diff(eye(N),s);                     % for x6, use s = 3

cvx_begin                               % use CVX package to solve the L_1 problem
    variable x5(N)
    minimize( sum_square(y-x5) + la * norm(D*x5,1) )
cvx_end

figure; plot(t,y,'.', t,x1,'-'); figure; plot(t,y,'.', t,x2,'-');
figure; plot(t,y,'.', t,x3,'-'); figure; plot(t,y,'.', t,x4,'-');
figure; plot(t,y,'.', t,x5,'-'); figure; plot(t,y,'.', t,x6,'-');
```

All methods have comparable performance and can handle the end-point problem. □

The computational procedures implemented into the function `lprs` were outlined in Eq. (4.1.19). The related orthogonalized basis Q defined in Eq. (4.1.23) will be realized in terms of the Hahn orthogonal polynomials.

A direct consequence of upper-triangular nature of the matrix C in Eq. (4.2.23) is that the basis Q becomes an *eigenvector basis* for the matrix VW [123,99]. To see this, substitute $S = QR$ into (4.2.23),

$$VWQR = QRC \quad\Rightarrow\quad VWQ = Q\Lambda\,,\quad \Lambda = RCR^{-1} \tag{4.2.32}$$

Multiplying both sides by Q^TW and using the property $Q^TWQ = D$, we obtain:

$$Q^TWVWQ = Q^TWQ\Lambda = D\Lambda \tag{4.2.33}$$

Because R and C are both upper-triangular, so will be Λ and $D\Lambda$. But the left-hand side of (4.2.33) is a symmetric matrix, and so must be the right-hand side $D\Lambda$. This requires that $D\Lambda$ and hence Λ be a diagonal matrix, e.g., $\Lambda = \text{diag}([\lambda_0, \lambda_1, \dots, \lambda_d])$. This means that the rth column of Q is an eigenvector:

$$VW\mathbf{q}_r = \lambda_r\mathbf{q}_r\,,\quad r = 0, 1, \dots, d \tag{4.2.34}$$

Choosing $d = N-1$ would produce all the eigenvectors of VW. In this case, we have $Q^{-1} = D^{-1}Q^TW$ and we obtain the decomposition:

$$VW = Q\Lambda Q^{-1} = Q\Lambda D^{-1}Q^TW \quad\Rightarrow\quad V = Q(\Lambda D^{-1})Q^T$$

We also find for the inverse of $V = D_s^TD_s$:

$$V^{-1} = WQ\Lambda^{-1}D^{-1}Q^TW$$

There exist [93–95] similar and efficient ways to calculate $V^{-1} = (D_s^TD_s)^{-1}$. The eigenvalues λ_r can be shown to be [123]:

$$\lambda_r = \frac{(2s+r)!}{r!} = \prod_{i=1}^{2s}(r+i)\,,\quad r = 0, 1, \dots, d \tag{4.2.35}$$

As we see in the next section, the rth column $q_r(n)$ of Q is a Hahn polynomial of degree r in n, and hence $W\mathbf{q}_r$, or component-wise, $w_nq_r(n)$, will be a polynomial of degree $2s+r$. Moreover, because of the zeros of w_n, the polynomial $f_n = w_nq_r(n)$ can be extended to be over the range $-M-s \le n \le M+s$. Using the same reasoning as in Eq. (4.2.19), it follows that (4.2.34) can be written as

$$(-1)^s\nabla^{2s}f^{\text{ext}}_{n+s} = \lambda_rq_r(n)\,,\quad -M \le n \le M$$

Since this is valid as an identity in n, it is enough to match the highest powers of n from both sides, that is, n^r. Thus, on the two sides we have

$$f^{\text{ext}}_{n+s} = w_{n+s}q_r(n+s) = \underbrace{(-1)^s[(n+s)^{2s} + \cdots]}_{w_{n+s}}\underbrace{[a_{rr}(n+s)^r + \cdots]}_{q_r(n+s)}\,,\quad \text{or,}$$

$$(-1)^sf^{\text{ext}}_{n+s} = a_{rr}n^{2s+r} + \cdots\,,\quad \text{and also,}\quad q_r(n) = a_{rr}n^r + \cdots$$

where a_{rr} is the highest coefficient of $q_r(n)$ and the dots indicate lower powers of n. Dropping the a_{rr} constant, the eigenvector condition then becomes:

$$\nabla^{2s}[n^{2s+r} + \cdots] = \lambda_r[n^r + \cdots]$$

Each operation of ∇ on n^i lowers the power by one, that is, $\nabla(n^i) = i\,n^{i-1} + \cdots$, $\nabla^2(n^i) = i(i-1)n^{i-2} + \cdots$, $\nabla^3(n^i) = i(i-1)(i-2)n^{i-3} + \cdots$, etc. Thus, we have:

$$\nabla^{2s}[n^{2s+r} + \cdots] = (2s+r)(2s+r-1)(2s+r-2)\cdots(r+1)\,n^r + \cdots$$

which yields Eq. (4.2.35).

4.3 Hahn Orthogonal Polynomials

Starting with Chebyshev [104], the discrete Chebyshev/Gram polynomials have been used repeatedly in the least-squares polynomial fitting problem, LPSM filter design, and other applications [104–151]. Bromba and Ziegler [123] were the first to establish a similar connection between the Hahn orthogonal polynomials and the minimum-R_s problem. For a review of the Hahn polynomials, see Karlin and McGregor [113].

The Hahn polynomials $Q_r(x)$ of a discrete variable $x = 0, 1, 2, \dots, N-1$ and orders $r \le N-1$ satisfy a weighted orthogonality property of the form:

$$\sum_{x=0}^{N-1} w(x)Q_r(x)Q_m(x) = D_r\delta_{rm}\,,\quad r, m = 0, 1, \dots, N-1$$

where the weighting function $w(x)$ depends on two parameters α, β and is defined up to a normalization constant as follows:

$$w(x) = \frac{(\alpha+x)!}{x!}\cdot\frac{(\beta+N-1-x)!}{(N-1-x)!}\,,\quad x = 0, 1, \dots, N-1 \tag{4.3.1}$$

The length N can be even or odd, but here we will consider only the odd case and set as usual $N = 2M+1$. The interval $[0, N-1]$ can be mapped onto the symmetric interval $[-M, M]$ by making the change of variables $x = n + M$, with $-M \le n \le M$. Then, the weighting function becomes,

$$w(n) = \frac{(\alpha + M + n)!}{(M + n)!} \cdot \frac{(\beta + M - n)!}{(M - n)!}, \quad -M \le n \le M \tag{4.3.2}$$

Defining $q_r(n) = Q_r(x)\big|_{x=n+M}$, the orthogonality property now reads:

$$\boxed{\sum_{n=-M}^{M} w(n)\, q_r(n)\, q_m(n) = D_r \delta_{rm}}, \quad r, m = 0, 1, \dots, N-1 \tag{4.3.3}$$

The minimum-R_s problem corresponds to the particular choice $\alpha = \beta = s$. In this case, the weighting function $w(n)$ reduces to the Henderson weights of Eq. (4.2.21):

$$w(n) = \frac{(s + M + n)!}{(M + n)!} \cdot \frac{(s + M - n)!}{(M - n)!} = \prod_{i=1}^{s}(M + n + i) \cdot \prod_{i=1}^{s}(M - n + i), \quad \text{or,}$$

$$w(n) = \prod_{i=1}^{s}\big[(M + i)^2 - n^2\big], \quad -M \le n \le M \tag{4.3.4}$$

For $s = 0$, the weights reduce to $w(n) = 1$ corresponding to the discrete Chebyshev/Gram polynomials. Because the weights are unity, the Chebyshev/Gram polynomials can be regarded as discrete-time versions of the Legendre polynomials. In fact, they tend to the latter in the limit $N \to \infty$ [133]. Similarly, the Hahn polynomials may be regarded as discrete versions of the Jacobi polynomials. At the opposite limit, $s \to \infty$, the Hahn polynomials tend to the *Krawtchouk* polynomials [133], which are discrete versions of the Hermite polynomials [130]. We review Krawtchouk polynomials and their application to the design of maximally flat filters in Sec. 4.4.

In general, the Hahn polynomials are given in terms of the hypergeometric function ${}_3F_2(a_1, a_2, a_3; b_1, b_2; z)$. For $\alpha = \beta = s$, they take the following explicit form:

$$\boxed{q_r(n) = Q_r(x) = \sum_{k=0}^{r} a_{rk}\, x^{[k]}\Big|_{x=n+M} = \sum_{k=0}^{r} a_{rk}\,(n + M)^{[k]}}, \quad -M \le n \le M \tag{4.3.5}$$

where $x^{[k]}$ denotes the falling-factorial power,

$$x^{[k]} = x(x-1)\cdots(x-k+1) = \frac{x!}{(x-k)!} = \frac{\Gamma(x+1)}{\Gamma(x-k+1)} \tag{4.3.6}$$

The polynomial coefficients are:

$$\boxed{a_{rk} = (-1)^k \prod_{m=1}^{k}\left[\frac{(r - m + 1)\,(2s + r + m)}{(N - m)\,(s + m)\,m}\right]}, \quad k = 0, 1, \dots, r \tag{4.3.7}$$

where $a_{r0} = 1$. Expanding the product we have:

$$a_{rk} = \frac{(-1)^k r(r-1)\cdots(r-k+1)\cdot(2s+r+1)\,(2s+r+2)\cdots(2s+r+k)}{(N-1)\,(N-2)\cdots(N-k)\cdot(s+1)\,(s+2)\cdots(s+k)\cdot k!} \tag{4.3.8}$$

The polynomials satisfy the symmetry property ,

$$q_r(-n) = (-1)^r q_r(n) \tag{4.3.9}$$

The orthogonality property (4.3.3) is satisfied with the following values of D_r:

$$\boxed{D_r = \frac{(s!)^2}{(2M)!} \cdot \frac{r!\,(2M - r)!}{(2M)!} \cdot \frac{(2s + r + 1)\,(2s + r + 2)\cdots(2s + r + N)}{2s + 2r + 1}} \tag{4.3.10}$$

For minimum-R_s filter design with polynomial order $d \le N - 1$, only polynomials up to order d are needed, that is, $q_r(n)$, $r = 0, 1, \dots, d$. Arranging these as the columns of the $N \times (d+1)$ matrix $Q = [\mathbf{q}_0, \mathbf{q}_1, \dots, \mathbf{q}_d]$, the orthogonality property can be expressed as $Q^T W Q = D$, where $D = \mathrm{diag}\big([D_0, D_1, \dots, D_d]\big)$.

The relationship to the monomial basis $S = [\mathbf{s}_0, \mathbf{s}_1, \dots, \mathbf{s}_d]$ is through an upper-triangular invertible matrix R, that is, $S = QR$. This can be justified by noting that the power series of $q_r(n)$ in n is a linear combination of the monomials $s_i(n) = n^i$ for $i = 0, 1, \dots, r$. In fact, R can be easily constructed from the Hahn coefficients a_{rk} and the Stirling numbers.

Thus, the construction of the minimum-R_s filters outlined in Eq. (4.1.23) is explicitly realized by the Hahn polynomial basis matrix Q:

$$B = W Q D^{-1} Q^T \tag{4.3.11}$$

or, component-wise,

$$b_m(n) = B_{nm} = w(n) \sum_{r=0}^{d} \frac{q_r(n)\, q_r(m)}{D_r}, \quad -M \le n, m \le M \tag{4.3.12}$$

A more direct derivation of (4.3.11) is to perform the local polynomial fit in the Q-basis. The desired degree-d polynomial can be expanded in the linear combination:

$$\hat{y}_m = \sum_{i=0}^{d} c_i m^i = \sum_{r=0}^{d} a_r q_r(m) \quad \Rightarrow \quad \hat{\mathbf{y}} = S\mathbf{c} = Q\mathbf{a}$$

Then, minimize the weighted performance index with respect to $\mathbf{a}$:

$$\mathcal{J} = (\mathbf{y} - Q\mathbf{a})^T W (\mathbf{y} - Q\mathbf{a}) = \min$$

Using the condition $Q^T W Q = D$, the solution leads to the same B:

$$\mathbf{a} = D^{-1} Q^T W \mathbf{y} \quad \Rightarrow \quad \hat{\mathbf{y}} = Q\mathbf{a} = Q D^{-1} Q^T W \mathbf{y} = B^T \mathbf{y} \tag{4.3.13}$$

The computation of the basis Q is facilitated by the following MATLAB functions. We note first that the falling factorial powers are related to ordinary powers by the *Stirling numbers* of the first and second kind:

$$x^{[k]} = \sum_{i=0}^{k} S_1(k, i)\, x^i \quad \Leftrightarrow \quad x^k = \sum_{i=0}^{k} S_2(k, i)\, x^{[i]} \tag{4.3.14}$$

These numbers may be arranged into lower-triangular matrices S_1 and S_2, which are inverses of each other. For example, we have for $k = 0, 1, 2, 3$:

$$\begin{bmatrix} x^{[0]} \\ x^{[1]} \\ x^{[2]} \\ x^{[3]} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 2 & -3 & 1 \end{bmatrix} \begin{bmatrix} x^0 \\ x^1 \\ x^2 \\ x^3 \end{bmatrix} \quad \Leftrightarrow \quad \begin{bmatrix} x^0 \\ x^1 \\ x^2 \\ x^3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 3 & 1 \end{bmatrix} \begin{bmatrix} x^{[0]} \\ x^{[1]} \\ x^{[2]} \\ x^{[3]} \end{bmatrix}$$

$$S_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 2 & -3 & 1 \end{bmatrix}, \qquad S_2 = S_1^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 3 & 1 \end{bmatrix}$$

The MATLAB function `stirling` generates these matrices up to a desired order:

```
S = stirling(d,kind);     % Stirling numbers up to order d of kind = 1,2
```

A polynomial can be expressed in falling factorial powers or in ordinary powers. The corresponding coefficient vectors are related by the Stirling numbers:

$$P(x) = \sum_{k=0}^{d} a_k x^{[k]} = \sum_{i=0}^{d} c_i x^i \quad \Rightarrow \quad \mathbf{c} = S_1^T \mathbf{a}, \quad \mathbf{a} = S_2^T \mathbf{c}$$

The function `polval` allows the evaluation of a polynomial in falling (or rising) factorial powers or in ordinary powers at any vector of x values:

```
P = polval(a,z,type);     % polynomial evaluation in factorial powers
```

The functions `hahncoeff`, `hahnpol`, and `hahnbasis` allow the calculation of the Hahn coefficients (4.3.7), the evaluation of the polynomial $Q_r(x)$ at any vector of x's, and the construction of the Hahn basis $Q = [\mathbf{q}_0, \mathbf{q}_1, \dots, \mathbf{q}_d]$:

```
[a,c] = hahncoeff(N,r,s);     % Hahn polynomial coefficients a_rk
Q = hahnpol(N,r,s,x);         % evaluate Hahn polynomial Q_r(x)
[Q,D,L] = hahnbasis(N,d,s);   % Hahn basis Q = [q_0,q_1,...,q_d]
```

Like all orthogonal polynomials, the Hahn polynomials satisfy a three-term recurrence relation of the form:

$$nq_r(n) = \alpha_r q_{r+1}(n) + \beta_r q_r(n) + \gamma_r q_{r-1}(n) \tag{4.3.15}$$

that starts with $r = 0$ and $q_{-1}(n) = 0$ and ends at $r = N - 2$. The recurrence relation is a direct consequence of the property (which follows from (4.3.3)) that the order-r polynomial $q_r(n)$ is orthogonal to *every* polynomial of degree strictly less than r. Let us denote the weighted inner product by

$$(a, b) = \sum_{n=-M}^{M} w(n) a(n) b(n) \tag{4.3.16}$$

Then, since the polynomial $nq_r(n)$ has degree $r+1$, it can be expanded as a linear combination of the polynomials $q_i(n)$ up to degree $r+1$:

$$nq_r(n) = \sum_{i=0}^{r+1} c_i q_i(n)$$

The coefficients are determined using the orthogonality property by

$$(nq_r, q_i) = \sum_{j=0}^{r+1} c_j (q_j, q_i) = \sum_{j=0}^{r+1} c_j D_i \delta_{ij} = D_i c_i \quad \Rightarrow \quad c_i = \frac{(nq_r, q_i)}{D_i} \tag{4.3.17}$$

This implies that $c_i = 0$ for $i \le r - 2$, therefore, only the terms $i = r+1, r, r-1$ will survive, which is the recurrence relation. Indeed, we note that $(nq_r, q_i) = (q_r, nq_i)$ and that $nq_i(n)$ has degree $(i + 1)$. Therefore, as long as $i + 1 < r$, or, $i \le r - 2$, this inner product will be zero. It follows from (4.3.17) that:

$$\alpha_r = \frac{(nq_r, q_{r+1})}{D_{r+1}}, \quad \beta_r = \frac{(nq_r, q_r)}{D_r}, \quad \gamma_r = \frac{(nq_r, q_{r-1})}{D_{r-1}} \tag{4.3.18}$$

Because the weights $w(n)$ are symmetric, $w(n) = w(-n)$, and the polynomials satisfy, $q_r(-n) = (-1)^r q_r(n)$, it follows immediately that $\beta_r = 0$. The coefficient γ_r can be related to α_{r-1} by noting that

$$\alpha_{r-1} = \frac{(nq_{r-1}, q_r)}{D_r} = \frac{(nq_r, q_{r-1})}{D_r} \quad \Rightarrow \quad (nq_r, q_{r-1}) = D_r \alpha_{r-1}, \quad \text{and hence,}$$

$$\gamma_r = \frac{(nq_r, q_{r-1})}{D_{r-1}} = \frac{D_r \alpha_{r-1}}{D_{r-1}} \tag{4.3.19}$$

Moreover, α_r is related to the leading coefficients a_{rr} of the $q_r(n)$ polynomial. From the definition (4.3.5), we can write

$$q_r(n) = a_{rr} n^r + p_{r-1}(n), \quad q_{r+1}(n) = a_{r+1,r+1} n^{r+1} + p_r(n)$$

where $p_{r-1}(n)$ and $p_r(n)$ are polynomials of degree $r-1$ and r, respectively. Since $D_{r+1} = (q_{r+1}, q_{r+1})$, we have,

$$\alpha_r = \frac{(nq_r, q_{r+1})}{(q_{r+1}, q_{r+1})} = \frac{(a_{rr} n^{r+1} + np_{r-1},\, q_{r+1})}{(a_{r+1,r+1} n^{r+1} + p_r,\, q_{r+1})} = \frac{a_{rr}(n^{r+1},\, q_{r+1})}{a_{r+1,r+1}(n^{r+1},\, q_{r+1})} = \frac{a_{rr}}{a_{r+1,r+1}}$$

where we used the orthogonality of $q_{r+1}(n)$ with $np_{r-1}(n)$ and $p_r(n)$, both of which have order r. Thus,

$$\alpha_r = \frac{a_{rr}}{a_{r+1,r+1}} \tag{4.3.20}$$

Using Eqs. (4.3.7) and (4.3.10), the expressions for α_r and γ_r simplify into:

$$\alpha_r = -\frac{(2M - r)(2s + r + 1)}{2(2s + 2r + 1)}, \quad \gamma_r = -\frac{r(2M + 2s + r + 1)}{2(2s + 2r + 1)} \tag{4.3.21}$$

These satisfy the constraint $\alpha_r + \gamma_r = -M$, which follows from the recurrence relation and the conditions $q_r(-M) = a_{r0} = 1$ for all r. Next, we derive the Christoffel-Darboux identity which allows the simplification of the sum in (4.3.12). Setting $\beta_r = 0$, replacing $\gamma_r = \alpha_{r-1}D_r/D_{r-1}$ and dividing by D_r, the recurrence relation reads:

$$\frac{nq_r(n)}{D_r} = \frac{\alpha_r}{D_r}\,q_{r+1}(n) + \frac{\alpha_{r-1}}{D_{r-1}}\,q_{r-1}(n) \tag{4.3.22}$$

Multiplying by $q_r(m)$, interchanging the roles of n, m, and subtracting, we obtain:

$$\begin{aligned}
\frac{nq_r(n)q_r(m)}{D_r} &= \frac{\alpha_r}{D_r}\,q_{r+1}(n)q_r(m) + \frac{\alpha_{r-1}}{D_{r-1}}\,q_{r-1}(n)q_r(m)\\
\frac{mq_r(m)q_r(n)}{D_r} &= \frac{\alpha_r}{D_r}\,q_{r+1}(m)q_r(n) + \frac{\alpha_{r-1}}{D_{r-1}}\,q_{r-1}(m)q_r(n)\\
\frac{(n-m)q_r(n)q_r(m)}{D_r} &= \frac{\alpha_r}{D_r}\big[q_{r+1}(n)q_r(m) - q_r(n)q_{r+1}(m)\big] -\\
&\quad - \frac{\alpha_{r-1}}{D_{r-1}}\big[q_r(n)q_{r-1}(m) - q_{r-1}(n)q_r(m)\big]
\end{aligned}$$

Summing up over r, and using $q_{-1}(n) = 0$, the successive terms on the right-hand side cancel except for the last one, resulting in the Christoffel-Darboux identity:

$$(n-m)\sum_{r=0}^{d}\frac{q_r(n)q_r(m)}{D_r} = \frac{\alpha_d}{D_d}\big[q_{d+1}(n)q_d(m) - q_d(n)q_{d+1}(m)\big]\,, \quad \text{or,}$$

$$\boxed{\sum_{r=0}^{d}\frac{q_r(n)q_r(m)}{D_r} = \frac{\alpha_d}{D_d}\,\frac{q_{d+1}(n)q_d(m) - q_d(n)q_{d+1}(m)}{n-m}} \tag{4.3.23}$$

Using this identity into the filter equations (4.3.12), we find

$$\boxed{b_m(n) = w(n)\,\frac{\alpha_d}{D_d}\,\frac{q_{d+1}(n)q_d(m) - q_d(n)q_{d+1}(m)}{n-m}} \tag{4.3.24}$$

This is valid for $-M \le n, m \le M$ and for orders $0 \le d \le N-2$. At $n = m$, the numerator vanishes, so that the numerator and denominator have a common factor $n - m$, which cancels resulting in a polynomial of degree d in n and m. In particular, the central Henderson filters are:

$$\boxed{b_0(n) = w(n)\,\frac{\alpha_d}{D_d}\,\frac{q_{d+1}(n)q_d(0) - q_d(n)q_{d+1}(0)}{n}} \tag{4.3.25}$$

where either $q_d(0)$ or $q_{d+1}(0)$ is zero depending on whether d is odd or even. In fact for the two successive values $d = 2r$ and $d = 2r+1$, while the asymmetric filters $b_m(n)$ are different, the central filters are the same and given by:

$$b_0(n) = \frac{\alpha_{2r}}{D_{2r}}\,q_{2r}(0)\,\frac{q_{2r+1}(n)}{n} = -\frac{\alpha_{2r+1}}{D_{2r+1}}\,q_{2r+2}(0)\,\frac{q_{2r+1}(n)}{n} \tag{4.3.26}$$

the equality of the coefficients following by setting $d = 2r+1$ and $n = 0$ in Eq. (4.3.22).

Next, we derive explicit formulas for some specific cases. The first few Hahn polynomials of orders $d = 0, 1, 2, 3, 4, 5$ and arbitrary M and s are, for $-M \le n \le M$:

$$\begin{aligned}
q_0(n) &= 1\\
q_1(n) &= -\frac{n}{M}\\
q_2(n) &= \frac{(2s+3)n^2 - M(M+s+1)}{M(2M-1)(s+1)}\\
q_3(n) &= -\frac{(2s+5)n^3 - \big[3M^2 + (s+1)(3M-1)\big]n}{M(M-1)(2M-1)(s+1)}\\
q_4(n) &= \frac{(2s+5)(2s+7)n^4 - (2s+5)\big(6M^2 + 6(s+1)M - 4s-5\big)n^2}{M(M-1)(2M-1)(2M-3)(s+1)(s+2)}\\
&\quad + \frac{3M(M-1)(s+M+1)(s+M+2)}{M(M-1)(2M-1)(2M-3)(s+1)(s+2)}\\
q_5(n) &= -\frac{(2s+7)(2s+9)n^5 - 5(2s+7)\big(2M^2 + 2(s+1)M - 2s-3\big)n^3}{M(M-1)(M-2)(2M-1)(2M-3)(s+1)(s+2)}\\
&\quad - \frac{\big[15M^4 + 30(s+1)M^3 + 5(3s^3+s-7)M^2 - (s+1)(s+2)(25M-6)\big]n}{M(M-1)(M-2)(2M-1)(2M-3)(s+1)(s+2)}
\end{aligned} \tag{4.3.27}$$

They are normalized such that $q_r(-M) = 1$. Setting $s = 0$, we obtain the corresponding *discrete Chebyshev/Gram* polynomials:

$$\begin{aligned}
q_0(n) &= 1\\
q_1(n) &= -\frac{n}{M}\\
q_2(n) &= \frac{3n^2 - M(M+1)}{M(2M-1)}\\
q_3(n) &= -\frac{5n^3 - (3M^2+3M-1)n}{M(M-1)(2M-1)}\\
q_4(n) &= \frac{35n^4 - 5(6M^2+6M-5)n^2 + 3M(M^2-1)(M+2)}{2M(M-1)(2M-1)(2M-3)}\\
q_5(n) &= -\frac{63n^5 - 35(2M^2+2M-3)n^3 + (15M^4+30M^3-35M^2-50M+12)n}{2M(M-1)(M-2)(2M-1)(2M-3)}
\end{aligned} \tag{4.3.28}$$

The central Henderson filters for the cases $d = 0, 1$, $d = 2, 3$, and $d = 4, 5$ are as follows for general M and s. For $d = 0, 1$:

$$b_0(n) = \frac{(2s+1)!\,(2M)!}{(s!)^2\,(2M+2s+1)!}\,w(n) \tag{4.3.29}$$

where $w(n)$ is given by Eq. (4.3.4). For $d = 2, 3$, we have:

$$b_0(n) = \frac{(M+s+1)(2s+3)!\,(2M)!\big(3M^2 + (s+1)(3M-1) - (2s+5)n^2\big)}{(2M-1)(s!)^2\,(2M+2s+3)!}\,w(n) \tag{4.3.30}$$

This generalizes Henderson's ideal formula (4.2.29) to arbitrary s. For $s = 1, 2$, it simplifies into:

$$s = 1, \quad b_0(n) = \frac{15(3M^2 + 6M - 2 - 7n^2)w_1(n)}{2(M+1)(2M+3)(2M+5)(4M^2-1)}$$

$$s = 2, \quad b_0(n) = \frac{105(M^2 + 3M - 1 - 3n^2)w_2(n)}{2(M+1)(M+2)(2M+3)(2M+5)(2M+7)(4M^2-1)}$$

where $w_1(n)$ and $w_2(n)$ correspond to (4.3.4) with $s = 1$ and $s = 2$. The case $s = 0$ is, of course, the same as Eq. (3.3.17). For the case $d = 4, 5$, we find:

$$\begin{aligned} b_0(n) &= \frac{(M+s+1)(M+s+2)(2s+5)!\,(2M)!}{2(2M-1)(2M-3)\big((s+2)!\big)^2(2M+2s+5)!}\cdot w(n)\cdot \\ &\quad\cdot\Big[(2s+7)(2s+9)n^4 - 5(2s+7)\big(2M^2 + 2(s+1)M - 2s-3\big)n^2 + \\ &\quad + 15M^4 + 30(s+1)M^3 + 5(3s^2+s-7)M^2 + (s+1)(s+2)(25M-6)\Big] \end{aligned} \tag{4.3.31}$$

Eqs. (4.3.29)–(4.3.31), as well as the case $d = 6, 7$, have been implemented into the MATLAB function `lprs2`, with usage:

```
b0 = lprs2(N,d,s);      % exact forms of the Henderson filters b0(n) for 0 ≤ d ≤ 6
```

The asymmetric interpolation filters $b_t(n)$ can be obtained by replacing the discrete variable m by t in Eqs. (4.3.12) and (4.3.24):

$$b_t(n) = w(n)\sum_{r=0}^{d}\frac{q_r(n)q_r(t)}{D_r} = w(n)\frac{\alpha_d}{D_d}\,\frac{q_{d+1}(n)q_d(t) - q_d(n)q_{d+1}(t)}{n-t} \tag{4.3.32}$$

Some specific cases are as follows. For $d = 0$, we have:

$$b_t(n) = \frac{(2s+1)!\,(2M)!}{(s!)^2\,(2M+2s+1)!}\,w(n) \tag{4.3.33}$$

For $d = 1$,

$$b_t(n) = \frac{4(2s+1)!\,(2M-1)!}{(s!)^2\,(2M+2s+2)!}\,w(n)\big[M^2 + (s+1)M + (2s+3)nt\big] \tag{4.3.34}$$

For $d = 2$:

$$\begin{aligned} b_t(n) &= \frac{4(2s+1)!\,(2M-1)!}{(s!)^2\,(2M+2s+2)!}\,w(n)\Big[M(M+s+1)\big[3M^2 + 3(s+1)M - s-1\big] \\ &\quad + (s+1)(2M-1)(2M+2s+3)nt - M(M+s+1)(2s+5)(n^2+t^2) \\ &\quad + (s+1)(2M-1)(2M+2s+3)n^2t^2\Big] \end{aligned} \tag{4.3.35}$$

The corresponding predictive differentiation filters are obtained by differentiating with respect to t.

The above closed-form expressions were obtained with the following simple Maple procedures that define the Hahn coefficients a_{rk}, the Hahn polynomials $q_r(n)$ and their norms D_r, and the interpolation filters $b_t(n)$:

```
factpow := proc(x,k) product((x-m), m=0..k-1); end proc;

a := proc(M,r,s,k)
        (-1)^k * product((r-m+1)*(2*s+r+m)/(2*M+1-m)/(s+m)/m, m=1..k);
     end proc;

Q := proc(M,r,s,n) if r=0 then 1; else
        sum(a(M,r,s,k)*factpow(n+M,k), k=0..r);
     end if; end proc;

Dr := proc(M,r,s) GAMMA(s+1)^2 * GAMMA(r+1) * GAMMA(2*M+1-r)
        * product(2*s+r+i, i=1..(2*M+1)) / GAMMA(2*M+1)^2 / (2*s+2*r+1);
      end proc;

B := proc(M,d,s,n,t)
        sum(Q(M,r,s,n)/Dr(M,r,s)*Q(M,r,s,t), r=0..d);
     end proc;
```

where `factpow` defines the falling-factorial powers, and it is understood that the result from the procedure `B(M,d,s,n,t)` must be multiplied by the Henderson weights $w(n)$.

There are other useful choices for the weighting function $w(n)$, such as binomial, which are similar to gaussian weights and lead to the Krawtchouk orthogonal polynomials, or exponentially decaying $w(n) = \lambda^n$, with $n \geq 0$ and $0 < \lambda < 1$, leading to the discrete Laguerre polynomials [135,136] and exponential smoothers. However, these choices do not have an equivalent minimum-NRR characterization. Even so, the smoothing filters are efficiently computed in the orthogonal polynomial basis by:

$$B = WS(S^TWS)^{-1}S^T = WQD^{-1}Q^T\,, \quad Q^TWQ = D \tag{4.3.36}$$

4.4 Maximally-Flat Filters and Krawtchouk Polynomials

Greville [84] has shown that in the limit $s \to \infty$ the minimum-R_s filters tend to maximally flat FIR filters that satisfy the usual flatness constraints at dc, that is, $B^{(i)}(\omega)\big|_{\omega=0} = \delta(i)$, for $i = 0, 1, \dots, d$, but also have monotonically decreasing magnitude responses and satisfy $(2M-d)$ additional flatness constraints at the Nyquist frequency, $\omega = \pi$. They are identical to the well-known maximally flat filters introduced by Herrmann [174]. Bromba and Ziegler [123,178] have shown that their impulse responses are given in terms of the Krawtchouk orthogonal polynomials [109,130,133]. Meer and Weiss [140] have derived the corresponding differentiation filters based on the Krawtchouk polynomials for application to images. Here, we look briefly at these properties.

The Krawtchouk polynomials are characterized by a parameter p such that $0 < p < 1$ and are defined over the symmetric interval $-M \leq n \leq M$ by [133]

$$\bar{q}_r(n) = \sum_{k=0}^{r}\frac{(-1)^k r(r-1)\cdots(r-k+1)p^{-k}}{(N-1)(N-2)\cdots(N-k)\cdot k!}(n+M)^{[k]} \tag{4.4.1}$$

where $N = 2M + 1$ and $r = 0, 1, \dots, N-1$. They satisfy the orthogonality property,

$$\sum_{n=-M}^{M}\bar{w}(n)\bar{q}_r(n)\bar{q}_m(n) = \bar{D}_r\delta_{rm} \tag{4.4.2}$$

with the following binomial weighting function and norms, where $q = 1 - p$:

$$\begin{aligned}\bar{w}(n) &= \binom{2M}{M+n} p^{M+n} q^{M-n} = \frac{(2M)!}{2^{2M}(M+n)!\,(M-n)!}\, p^{M+n} q^{M-n} \\ \bar{D}_r &= \frac{r!\,(2M-r)!}{(2M)!}\,\frac{q^r}{p^r}\end{aligned} \tag{4.4.3}$$

In the limit $s \to \infty$, the Hahn polynomials tend to the special Krawtchouk polynomials with the parameter $p = q = 1/2$. To see this, we note that the Hahn polynomials are normalized such that $q_r(-M) = 1$, and we expect that they would have a straightforward limit as $s \to \infty$. Indeed, it is evident that the limit of the Hahn coefficients (4.3.8) is

$$\bar{a}_{rk} = \lim_{s\to\infty} a_{rk} = \frac{(-1)^k r(r-1)\cdots(r-k+1)\cdot 2^k}{(N-1)(N-2)\cdots(N-k)\cdot k!} \tag{4.4.4}$$

and therefore, the Hahn polynomials will tend to

$$\bar{q}_r(n) = \sum_{k=0}^{r} \frac{(-1)^k r(r-1)\cdots(r-k+1)\cdot 2^k}{(N-1)(N-2)\cdots(N-k)\cdot k!}\,(n+M)^{[k]} \tag{4.4.5}$$

which are recognized as a special case of (4.4.1) with $p = 1/2$. The Henderson weights (4.3.4) and norms (4.3.10) diverge as $s \to \infty$, but we may normalize them by a common factor, such as $s^{2M}(s!)^2$, so that they will converge. The limits of the rescaled weights and norms are:

$$\begin{aligned}\bar{w}(n) &= \lim_{s\to\infty}\left[\frac{(2M)!\,w(n)}{2^{2M}s^{2M}(s!)^2}\right] = \lim_{s\to\infty}\left[\frac{(2M)!\,(s+M+n)!\,(s+M-n)!}{2^{2M}s^{2M}(s!)^2\,(M+n)!\,(M-n)!}\right] \\ \bar{D}_r &= \lim_{s\to\infty}\left[\frac{(2M)!\,D_r}{2^{2M}s^{2M}(s!)^2}\right] \\ &= \lim_{s\to\infty}\left[\frac{r!\,(2M-r)!}{(2M)!}\cdot\frac{(2s+r+1)(2s+r+2)\cdots(2s+r+N)}{2^{2M}s^{2M}(2s+2r+1)}\right]\end{aligned}$$

They are easily seen to lead to Eqs. (4.4.3) with $p = 1/2$, that is,

$$\begin{aligned}\bar{w}(n) &= \frac{1}{2^{2M}}\binom{2M}{M+n} = \frac{(2M)!}{2^{2M}(M+n)!\,(M-n)!} \\ \bar{D}_r &= \frac{r!\,(2M-r)!}{(2M)!}\end{aligned} \tag{4.4.6}$$

The first few of the Krawtchouk polynomials are:

$$\begin{aligned}\bar{q}_0(n) &= 1 \\ \bar{q}_1(n) &= -\frac{n}{M} \\ \bar{q}_2(n) &= \frac{2n^2 - M}{M(2M-1)} \\ \bar{q}_3(n) &= -\frac{2n^3 - (3M-1)n}{M(M-1)(2M-1)} \\ \bar{q}_4(n) &= \frac{4n^4 - (12M-8)n^2 + 3M(M-1)}{M(M-1)(2M-1)(2M-3)} \\ \bar{q}_5(n) &= -\frac{4n^5 - 20(M-1)n^3 + (15M^2-25M+6)n}{M(M-1)(M-2)(2M-1)(2M-3)}\end{aligned} \tag{4.4.7}$$

These polynomials satisfy the three-term recurrence relation:

$$n\bar{q}_r(n) = \bar{\alpha}_r\bar{q}_{r+1}(n) + \bar{\gamma}_r\bar{q}_{r-1}(n)\,, \quad \bar{\alpha}_r = -\frac{2M-r}{2}\,, \quad \bar{\gamma}_r = -\frac{r}{2} \tag{4.4.8}$$

with the coefficients $\bar{\alpha}_r, \bar{\gamma}_r$ obtained from Eq. (4.3.21) in the limit $s \to \infty$. The three-term relations lead to the usual Christoffel-Darboux identity from which we may obtain the asymmetric predictive filters:

$$\bar{b}_t(n) = \bar{w}(n)\sum_{r=0}^{d}\frac{\bar{q}_r(n)\bar{q}_r(t)}{\bar{D}_r} = \bar{w}(n)\,\frac{\bar{\alpha}_d}{\bar{D}_d}\,\frac{\bar{q}_{d+1}(n)\bar{q}_d(t) - \bar{q}_d(n)\bar{q}_{d+1}(t)}{n-t} \tag{4.4.9}$$

Differentiation with respect to t gives the corresponding predictive differentiation filters. Some examples are as follows. For $d = 0$ and $d = 1$, we have, respectively

$$\bar{b}_t(n) = \bar{w}(n)\,, \quad \bar{b}_t(n) = \bar{w}(n)\,\frac{2nt+M}{M} \tag{4.4.10}$$

For $d = 2$, the smoothing and first-order differentiation filters are:

$$\begin{aligned}\bar{b}_t(n) &= \bar{w}(n)\,\frac{4n^2t^2 - 2M(n^2+t^2) + 2(2M-1)nt + M(3M-1)}{M(2M-1)} \\ \dot{\bar{b}}_t(n) &= \bar{w}(n)\,\frac{2(2M-1)n - 4Mt + 8n^2t}{M(2M-1)}\end{aligned} \tag{4.4.11}$$

and setting $t = 0$, the central filters simplify into:

$$\bar{b}_0(n) = \bar{w}(n)\,\frac{3M-1-2n^2}{2M-1}\,, \quad \dot{\bar{b}}_0(n) = \bar{w}(n)\,\frac{2n}{M} \tag{4.4.12}$$

For $d = 3$, we have:

$$\begin{aligned}\bar{b}_t(n) = \frac{\bar{w}(n)}{3M(M-1)(2M-1)}\Big[&8n^3t^3 - 4(3M-1)(n^3t + nt^3) + 12(M-1)n^2t^2 \\ &- 6M(M-1)(n^2+t^2) + (30M^2-30M+8)nt - 3M(M-1)(3M-1)\Big]\end{aligned} \tag{4.4.13}$$

As expected, setting $t = 0$ produces the same result as the $d = 2$ case. Numerically, the smoothing and differentiation filters can be calculated by passing the Krawtchouk weights $\bar{w}(n)$ into the functions `lpsm`, `lpdiff`, and `lpinterp`:

```
W  = diag(hend(N,inf));      % Krawtchouk weights
B  = lpsm(N,d,W);            % smoothing filters
Bi = lpdiff(N,d,i,W);        % i-th derivative filters
b  = lpinterp(N,d,t,i,W);    % interpolation filters b_t
```

The function $\text{hend}(N, s)$, with $s = \infty$, calculates the Krawtchouk weights of Eq. (4.4.6). In turn, the filter matrices B or $B^{(i)}$ may be passed into the filtering function `lpfilt`. Alternatively, one can call `lprs` with $s = \infty$:

```
B = lprs(N,d,inf);      % LPRS with Krawtchouk weights, maximally-flat filters
```

It is well-known [84,174–187] that the maximally-flat FIR filters of length $N = 2M+1$ and polynomial order $d = 2r + 1$ have frequency responses given by the following equivalent expressions:

$$\begin{aligned} B_0(\omega) &= \sum_{i=0}^{r}\binom{M}{i}x^i(1-x)^{M-i} = 1-\sum_{i=r+1}^{M}\binom{M}{i}x^i(1-x)^{M-i} \\ &= (1-x)^{M-r}\sum_{i=0}^{r}\binom{M-r+i-1}{i}x^i\,, \qquad \text{where } x = \sin^2\left(\frac{\omega}{2}\right) \end{aligned} \tag{4.4.14}$$

Near $\omega \simeq 0$ and near $\omega \simeq \pi$, the second and third expressions have the following expansions that exhibit the desired flatness constraints [123]:

$$\begin{aligned} \omega \simeq 0 \quad &\Rightarrow \quad B_0(\omega) \simeq 1 - (\text{const.})\,\omega^{2r+2} = 1 - (\text{const.})\,\omega^{d+1} \\ \omega \simeq \pi \quad &\Rightarrow \quad B_0(\omega) \simeq (\text{const.})\,(\omega-\pi)^{2M-2r} = (\text{const.})\,(\omega-\pi)^{2M-d+1} \end{aligned} \tag{4.4.15}$$

The first implies the flatness constraints at dc, $B_0^{(i)}(0) = \delta(i)$, for $i = 0, 1, \dots, d$, and the second, the flatness constraints at Nyquist, $B_0^{(i)}(\pi) = 0$, for $i = 0, 1, \dots, 2M-d$.

Example 4.4.1: For $d = 2$ or $r = 1$, the z-transform of $b_0(n)$ in Eq. (4.4.12) can be calculated explicitly resulting in:

$$B_0(z) = \left[\frac{(1+z^{-1})(1+z)}{4}\right]^{M-1}\frac{1}{4}\big[2(M+1) - (M-1)(z+z^{-1})\big]$$

With $z = e^{j\omega}$ we may write

$$\begin{aligned} x &= \sin^2\left(\frac{\omega}{2}\right) = \frac{(1-z^{-1})(1-z)}{4} = \frac{2-z-z^{-1}}{4} \quad \Rightarrow \quad \frac{z+z^{-1}}{4} = \frac{1}{2} - x \\ 1 - x &= \cos^2\left(\frac{\omega}{2}\right) = \frac{(1+z^{-1})(1+z)}{4} \end{aligned}$$

Thus, we may express $B_0(z)$ in terms of the variable x:

$$B_0(z) = (1-x)^{M-1}\big[1 + (M-1)x\big]$$

which corresponds to Eq. (4.4.14) for $r = 1$. □

Example 4.4.2: Fig. 4.4.1 shows the frequency responses $B_0(\omega)$ for the values $N = 13$, $r = 2$, ($d = 4, 5$), and the smoothness parameter values: $s = 3$, $s = 6$, $s = 9$, and $s = \infty$.

Because $b_0(n)$ is symmetric about $n = 0$, the quantities $B_0(\omega)$ are real-valued. In the limit $s \to \infty$, the response becomes positive and monotonically decreasing. The following MATLAB code illustrates the generation of the bottom two graphs and verifies Eq. (4.4.14):

```
N=13; r=2; d = 2*r+1; M = floor(N/2);

B = lprs(N,d,9);   b9   = B(:,M+1);          % LPRS filter with s = 9
B = lprs(N,d,inf); binf = B(:,M+1);          % LPRS with Krawtchouk weights

f = linspace(0,1,1001); w = pi*f; x = sin(w/2).^2;
B9   = real(exp(j*w*M) .* freqz(b9,1,w));          % frequency responses
Binf = real(exp(j*w*M) .* freqz(binf,1,w));

Bth = 0;
for i=0:r,
   Bth = Bth + nchoosek(M,i) * x.^i .* (1-x).^(M-i);     % Eq. (4.4.14)
end

norm(Bth-Binf)                        % compare Eq. (4.4.14) with output of LPSM

figure; plot(f,B9); figure; plot(f,Binf);
```

The calls to `lprs` and `lpsm` return the full smoothing matrices B from which the central column $\mathbf{b}_0$ is extracted.

The frequency response function `freqz` expects its filter argument to be causal. The factor $e^{j\omega M}$ compensates for that, corresponding to a time-advance by M units. □

Finally, we note that the Krawtchouk binomial weighting function $\bar{w}(n)$ tends to a gaussian for large M, which is a consequence of the De Moivre-Laplace theorem,

$$\bar{w}(n) = \frac{(2M)!}{2^{2M}(M+n)!\,(M-n)!} \simeq \frac{1}{\sqrt{\pi M}}\,e^{-n^2/M}\,, \quad -M \le n \le M \tag{4.4.16}$$

In fact, the two sides of (4.4.16) are virtually indistinguishable for $M \ge 10$.

4.5 Missing Data and Outliers

The presence of outliers in the observed signal can cause large distortions in the smoothed signal. The left graph of Fig. 4.5.1 shows what can happen. The two vertical lines indicate the region in which there are four strong outliers, which cause the smoothed curve to deviate drastically from the desired signal.

One possible solution [53,165] is to ignore the outliers and estimate the smoothed values from the surrounding available samples using a filter window that spans the outlier region. The same procedure can be used if some data samples are missing. Once the outliers or missing values have been interpolated, one can apply the weighted LPSM filters as usual. The right graph in Fig. 4.5.1 shows the four adjusted interpolated samples. The resulting smoothed signal now estimates the desired signal more accurately.

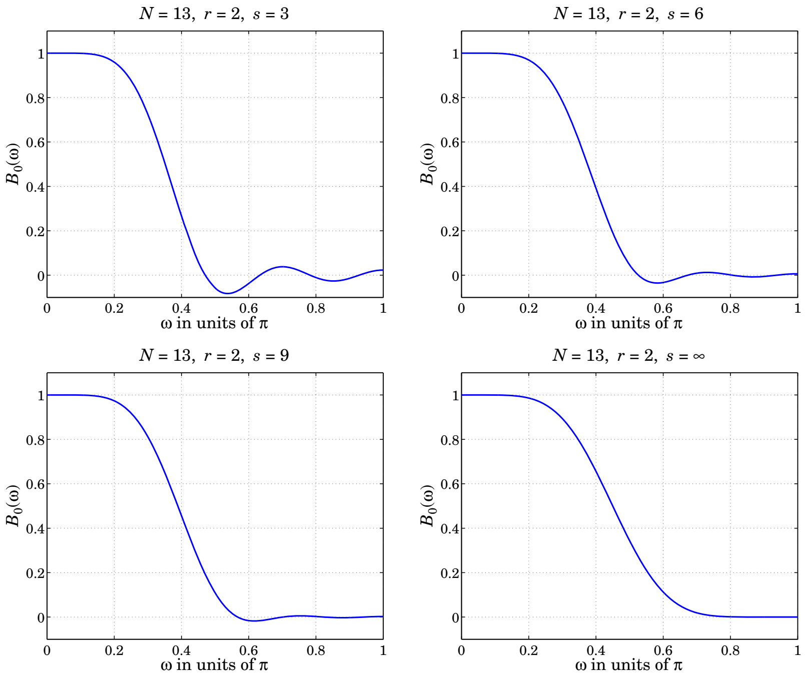


Fig. 4.4.1 Frequency responses of minimum-R_s and maximally-flat filters.

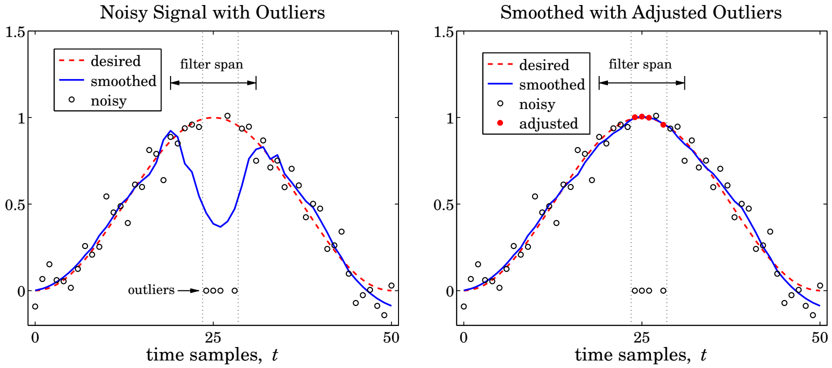


Fig. 4.5.1 Smoothing with missing data or outliers.

This solution can be implemented by replacing the outliers or the missing data by zeros (or, any other values), and assign zero weights to them in the least-squares polynomial fitting problem.

Given a long observed signal y_n, $n = 0, 1, \dots, L-1$, let us assume that in the vicinity of $n = n_0$ there is an outlier or missing sample at the time instant n_0+m, where m lies in the interval $-M \le m \le M$, as shown in Fig. 4.5.2. Several outliers or missing data may be present, not necessarily adjacent to each other, each being characterized by a similar index m.

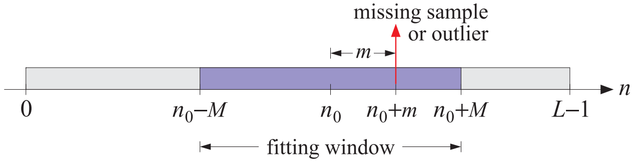


Fig. 4.5.2 Missing sample or outlier and the data window used for estimating it.

The outlier samples y_{n_0+m} can be replaced by zeros and their estimated values, $\hat{y}_{n_0+m}$, can be calculated from the surrounding samples using a filter of length $N = 2M+1$. The corresponding least-squares polynomial-fitting problem is defined by

$$\mathcal{J} = \sum_{m=-M}^{M} p_m w_m \left(y_{n_0+m} - \sum_{i=0}^{d} c_i m^i\right)^2 = \min \tag{4.5.1}$$

where w_m are the usual Henderson weights and the p_m are zero at the indices for the missing data, and unity otherwise. Let $\mathbf{y} = [y_{n_0-M}, \dots, y_{n_0}, \dots, y_{n_0+M}]^T$, and denote by $W, \mathcal{P}$ the corresponding diagonal matrices of the weights w_m, p_m. Then, (4.5.1) reads:

$$\mathcal{J} = (\mathbf{y} - S\mathbf{c})^T \mathcal{P} W (\mathbf{y} - S\mathbf{c}) = \min, \tag{4.5.2}$$

leading to the orthogonality conditions and the solution for $\mathbf{c}$:

$$S^T W \mathcal{P} (\mathbf{y} - S\mathbf{c}) = 0 \quad \Rightarrow \quad \mathbf{c} = (S^T \mathcal{P} W S)^{-1} S^T W \mathcal{P} \mathbf{y} \tag{4.5.3}$$

where we assumed that $S^T \mathcal{P} W S$ is invertible.[†] The estimated samples will be:

$$\hat{\mathbf{y}} = S\mathbf{c} = S(S^T \mathcal{P} W S)^{-1} S^T W \mathcal{P} \mathbf{y} = B^T \mathbf{y} \tag{4.5.4}$$

with the filter matrix,

$$B = \mathcal{P} W S (S^T \mathcal{P} W S)^{-1} S^T \tag{4.5.5}$$

We note that $\mathcal{P}$ is a projection matrix ($\mathcal{P}^T = \mathcal{P}$ and $\mathcal{P}^2 = \mathcal{P}$) and commutes with W, $\mathcal{P}W = W\mathcal{P}$, because both are diagonal. Defining $\mathcal{Q} = I - \mathcal{P}$ to be the complementary projection matrix, the estimated signal can be decomposed in two parts: $\hat{\mathbf{y}} = \mathcal{P}\hat{\mathbf{y}} + \mathcal{Q}\hat{\mathbf{y}}$, with $\mathcal{Q}\hat{\mathbf{y}}$ being the part that contains the estimated missing values or adjusted outliers.

The quantity $\mathcal{P}\mathbf{y}$ represents the samples that are being used to make the estimates, whereas $\mathcal{Q}\mathbf{y}$ corresponds to the missing samples and can be set to zero or to an arbitrary vector $\mathcal{Q}\mathbf{y}_{\text{arb}}$, in other words, we may replace $\mathbf{y}$ by $\mathcal{P}\mathbf{y} + \mathcal{Q}\mathbf{y}_{\text{arb}}$ without affecting the solution of Eq. (4.5.4). This so because $\mathcal{P}(\mathcal{P}\mathbf{y} + \mathcal{Q}\mathbf{y}_{\text{arb}}) = \mathcal{P}\mathbf{y}$.

[†] This requires that the number of outliers within the data window be at most $N - d - 1$.

Once the estimated missing values have been obtained, we may replace $Q\mathbf{y}_{\text{arb}}$ by $\mathcal{Q}\hat{\mathbf{y}}$ and recompute the ordinary W-weighted least-squares estimate from the adjusted vector $\mathcal{P}\mathbf{y}+\mathcal{Q}\hat{\mathbf{y}}$. This produces the same $\hat{\mathbf{y}}$ as in (4.5.4). Indeed, one can show that,

$$\hat{\mathbf{y}} = S(S^T\mathcal{P}WS)^{-1}S^TW\mathcal{P}\mathbf{y} = S(S^TWS)^{-1}S^TW(\mathcal{P}\mathbf{y}+\mathcal{Q}\hat{\mathbf{y}}) \tag{4.5.6}$$

To see this, start with the orthogonality equation (4.5.3), and replace $\mathcal{P}\hat{\mathbf{y}} = \hat{\mathbf{y}} - \mathcal{Q}\hat{\mathbf{y}}$:

$$S^TW\mathcal{P}(\mathbf{y}-\hat{\mathbf{y}}) = 0 \quad\Rightarrow\quad S^TW\mathcal{P}\mathbf{y} = S^TW\mathcal{P}\hat{\mathbf{y}} = S^TW(\hat{\mathbf{y}}-\mathcal{Q}\hat{\mathbf{y}})\,, \quad \text{or,}$$

$$S^TW(\mathcal{P}\mathbf{y}+\mathcal{Q}\hat{\mathbf{y}}) = S^TW\hat{\mathbf{y}} = S^TWS(S^T\mathcal{P}WS)^{-1}W\mathcal{P}\mathbf{y}$$

from which Eq. (4.5.6) follows by multiplying both sides by $S(S^TWS)^{-1}$. The MATLAB function `lpmissing` implements the calculation of B in (4.5.5):

```
B = lpmissing(N,d,m,s);       % filter matrix for missing data
```

The following MATLAB code illustrates the generation of Fig. 4.5.1:

```
t = (0:50)'; x0 = (1-cos(2*pi*t/50))/2;      % desired signal

seed=2005; randn('state',seed);
y = x0 + 0.1 * randn(51,1);                  % noisy signal

n0 = 25; m = [-1 0 1 3];                     % four outlier indices relative to n0
y(n0+m+1) = 0;                               % four outlier or missing values

N= 13; d = 2; s = 0; M=(N-1)/2;              % filter specs
x = lpfilt(lprs(N,d,s),y);                   % distorted smoothed signal

B = lpmissing(N,d,m,s);                      % missing-data filter B

yhat = B'*y(n0-M+1:n0+M+1);                  % apply B to the block n0-M ≤ n ≤ n0+M
ynew = y; ynew(n0+m+1) = yhat(M+1+m);        % new signal with interpolated outlier values

xnew = lpfilt(lprs(N,d,s),ynew);             % recompute smoothed signal

figure; plot(t,x0,'--', t,y,'o', t,x,'-');        % left graph
figure; plot(t,x0,'--', t,y,'o', t,xnew,'-');     % right graph
hold on; plot(n0+m,yhat(M+1+m),'.');
```

The above method of introducing zero weights at the outlier locations can be automated and applied to the entire signal. Taking a cue from Cleveland's LOESS method [192] discussed in the next section, we may apply the following procedure.

Given a length-L signal y_n, $n = 0,1,\dots,L-1$, with $L \ge N$, an LPSM or LPRS filter with design parameters N,d,s can be applied to y_n to get a preliminary estimate of the smoothed signal $\hat{x}_n$, and compute the error residuals $e_n = y_n - \hat{x}_n$, that is,

$$\begin{aligned} B &= \text{lprs}(N,d,s)\\ \hat{\mathbf{x}} &= \text{lpfilt}(B,\mathbf{y})\\ \mathbf{e} &= \mathbf{y}-\hat{\mathbf{x}} \end{aligned} \tag{4.5.7}$$

From the error residual $\mathbf{e}$, one may compute a set of "robustness" weights r_n by using the median of $|e_n|$ as a normalization factor in the bisquare function:

$$\mu = \text{median}(|e_n|)\,, \quad r_n = W\left(\frac{e_n}{K\mu}\right), \quad n = 0,1,\dots,L-1 \tag{4.5.8}$$

where K is a constant such as $K = 2$–6, and $W(u)$ is the bisquare function,

$$W(u) = \begin{cases} (1-u^2)^2, & \text{if } |u| \le 1 \\ 0, & \text{otherwise} \end{cases} \tag{4.5.9}$$

If a residual e_n deviates too far from the median, that is, $|e_n| > K\mu$, then the robustness weight r_n is set to zero. A new estimate $\hat{x}_n$ can be calculated at each time n by defining the diagonal matrix P in terms of the robustness weights in the neighborhood of n, and then calculating the estimate using the c_0 component of the vector $\mathbf{c}$ in Eq. (4.5.3), that is,

$$\begin{aligned} P_n &= \text{diag}([r_{n-M},\dots,r_n,\dots,r_{n+M}])\\ \hat{x}_n &= c_0 = \mathbf{u}_0^T(S^TP_nWS)^{-1}S^TWP_n\mathbf{y}(n) \end{aligned} \tag{4.5.10}$$

where $\mathbf{u}_0 = [1,0,\dots,0]^T$ and $\mathbf{y}(n) = [y_{n-M},\dots,y_n,\dots,y_{n+M}]^T$. Eq. (4.5.10) may be used for $M \le n \le L-1-M$. For $0 \le n < M$ and $L-1-M < n \le L-1$ the values of $\hat{x}_n$ can be obtained from the first M and last M outputs of $\hat{\mathbf{y}}$ in (4.5.4) applied to the first and last length-N data vectors and robustness weights:

$$\mathbf{y} = [y_0, y_1, \dots, y_{N-1}]^T\,, \quad P = \text{diag}([r_0, r_1, \dots, r_{N-1}])$$

$$\mathbf{y} = [y_{L-N}, y_{L-N+1}, \dots, y_{L-1}]^T\,, \quad P = \text{diag}([r_{L-N}, r_{L-N+1}, \dots, r_{L-1}])$$

From the new estimates $\hat{x}_n$, one can compute the new residuals $e_n = y_n - \hat{x}_n$, and repeat the procedure of Eqs. (4.5.8)–(4.5.10) a few more times. A total of 3–4 iterations is typically adequate. The MATLAB function `rlpfilt` implements the above steps:

```
[x,r] = rlpfilt(y,N,d,s,Nit)       % robust local polynomial filtering
```

Its outputs are the estimated signal $\hat{x}_n$ and the robustness weights r_n. The median scaling factor K is an additional optional input, which otherwise defaults to $K = 6$.

If the residuals e_n are gaussian-distributed with variance σ^2, then $\mu = 0.6745\sigma$. The default value $K = 6$ (Cleveland [192]) corresponds to allowing through 99.99 percent of the residuals. Other possible values are $K = \sqrt{6} = 2.44$ (Loader [224]) and $K = 4$ allowing respectively 90 and 99 percent of the values.

Fig. 4.5.3 shows the effect of increasing the number of robustness iterations. It is the same example as that in Fig. 4.5.1, but we have added another four outliers in the vicinity of $n = 10$. The upper-left graph corresponds to ordinary filtering without any robustness weights. One observes the successive improvement of the estimate as the number of iterations increases.

The following MATLAB code illustrates the generation of the lower-right graph. The signal y_n is generated exactly as in the previous example; the outlier values are then introduced around $n = 10$ and $n = 25$:

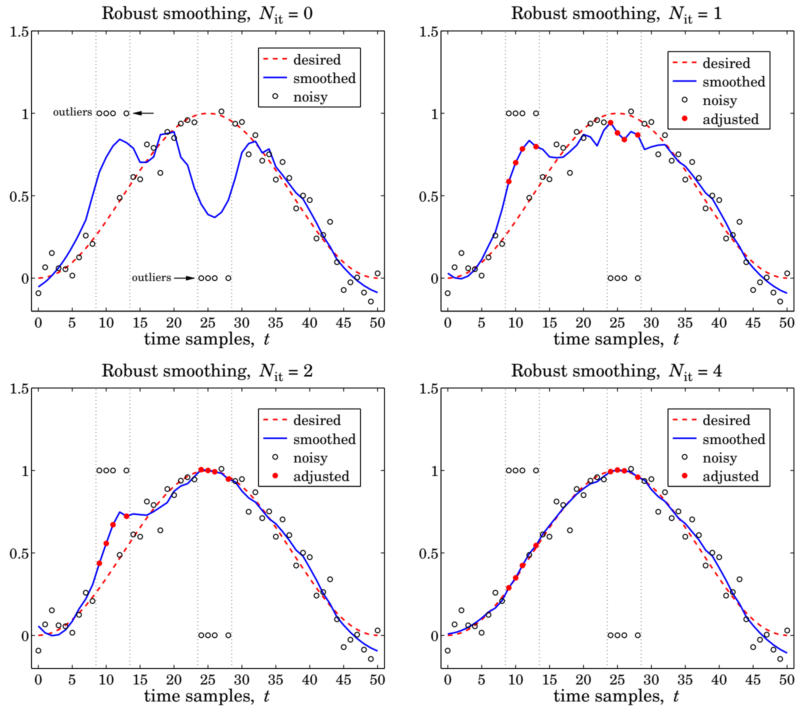


Fig. 4.5.3 Robust smoothing with outliers.

```
n1=10; n2=25; m = [-1 0 1 3];                    % outlier indices relative to n1 and n2
y(n1+m+1)=1;  y(n2+m+1)=0;                       % outlier values

Nit=4; K=4; x = rlpfilt(y,N,d,s,Nit,K);          % robust LP filtering

plot(t,x0,'--', t,y,'o', t,x,'-', n1+m,x(n1+m+1),'.', n2+m,x(n2+m+1),'.');
```

4.6 Problems

4.1 Using binomial identities, prove the equivalence of the three expressions in Eq. (4.4.14) for the maximally-flat filters. Then, show Eq. (4.4.15) and determine the proportionality constants indicated as (const.).

5
Local Polynomial Modeling

5.1 Weighted Local Polynomial Modeling

The methods of weighted least-squares local polynomial modeling and robust filtering can be generalized to unequally-spaced data in a straightforward fashion. Such methods provide enough flexibility to model a wide variety of data, including surfaces, and have been explored widely in recent years [188–231]. For equally-spaced data, the weighted performance index centered at time n was:

$$\mathcal{J}_n = \sum_{m=-M}^{M} \big(y_{n+m} - p(m)\big)^2 w(m) = \min\,, \quad p(m) = \sum_{r=0}^{d} c_i m^r \tag{5.1.1}$$

The value of the fitted polynomial $p(m)$ at $m = 0$ represents the smoothed estimate of y_n, that is, $\hat{x}_n = c_0 = p(0)$. Changing summation indices to $k = n + m$, Eq. (5.1.1) may be written in the form:

$$\mathcal{J}_n = \sum_{k=n-M}^{n+M} \big(y_k - p(k-n)\big)^2 w(k-n) = \min\,, \quad p(k-n) = \sum_{r=0}^{d} c_i (k-n)^r \tag{5.1.2}$$

For a set of N unequally-spaced observations $\{t_k, y(t_k)\}$, $k = 0, 1, \dots, N-1$, we wish to interpolate smoothly at some time instant t, not necessarily coinciding with one of the observation times t_k, but lying in the interval $t_0 \le t \le t_{N-1}$. A generalization of the performance index (5.1.2) is to introduce a t-dependent window bandwidth h_t, and use only the observations that lie within that window, $|t_k - t| \le h_t$, to perform the polynomial fit:

$$\mathcal{J}_t = \sum_{|t_k - t| \le h_t} \big(y(t_k) - p(t_k - t)\big)^2 w(t_k - t) = \min\,, \quad p(t_k - t) = \sum_{r=0}^{d} c_r (t_k - t)^r \tag{5.1.3}$$

The estimated/interpolated value at t will be $\hat{x}_t = c_0 = p(0)$, and the estimated first derivative, $\hat{\dot{x}}_t = c_1 = \dot{p}(0)$, and so on for the higher derivatives, with $r!\,c_r$ representing the rth derivative. As illustrated in Fig. 5.1.1, the fitted polynomial,

$$p(x-t) = \sum_{r=0}^{d} c_r (x-t)^r\,, \qquad t - h_t \le x \le t + h_t$$

is local in the sense that it fits the observations only within the local window $[t-h_t, t+h_t]$. The quantity $\hat{y}_k = p(t_k - t)$ represents the estimated value of the kth observation y_k within that window.

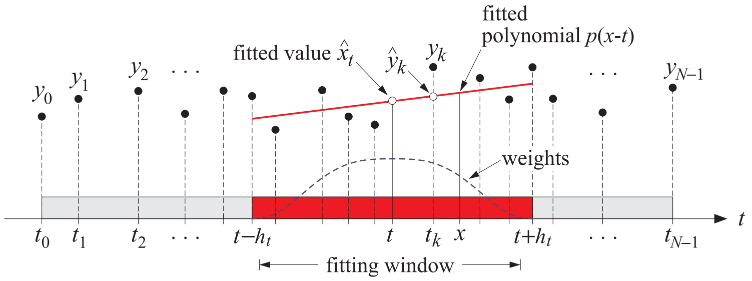


Fig. 5.1.1 Local polynomial modeling.

The weighting function $w(t_k - t)$ is chosen to have bandwidth $\pm h_t$. This can be accomplished by using a function $W(u)$ with finite support over the standardized range $-1 \le u \le 1$, and setting $u = (t_k - t)/h_t$:

$$w(t_k - t) = W\left(\frac{t_k - t}{h_t}\right) \tag{5.1.4}$$

Some typical choices for $W(u)$ are as follows [224]:

$$\begin{array}{ll} \text{1. Tricube,} & W(u) = (1-|u|^3)^3 \\ \text{2. Bisquare,} & W(u) = (1-u^2)^2 \\ \text{3. Triweight,} & W(u) = (1-u^2)^3 \\ \text{4. Epanechnikov,} & W(u) = 1-u^2 \\ \text{5. Gaussian,} & W(u) = e^{-\alpha^2 u^2/2} \\ \text{6. Exponential,} & W(u) = e^{-\alpha|u|} \\ \text{7. Rectangular,} & W(u) = 1 \end{array} \tag{5.1.5}$$

where all types have support $|u| \le 1$ and vanish for $|u| > 1$. A typical value for α in the gaussian and exponential cases is $\alpha = 2.5$. The curve shown in Fig. 5.1.1 is the tricube function; because it vanishes at $u = \pm 1$, the observations that fall exactly at the edges of the window do not contribute to the fit. The MATLAB function `locw` generates the above functions at any vector of values of u:

```
W = locw(u,type);      % local polynomial weighting functions W(u)
```

where `type` takes the values 1–7 as listed in Eq. (5.1.5). The bisquare, triweight, and Epanechnikov functions are special cases of the more general $W(u) = (1-u^2)^s$, which may be thought of as the large-M limit of the Henderson weights; in the limit $s \to \infty$ they tend to a gaussian, as in the Krawtchouk case. The various window functions are depicted in Fig. 5.1.2.

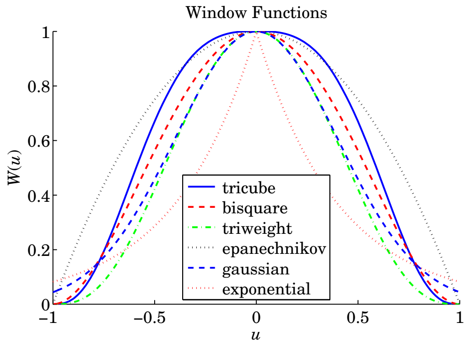


Fig. 5.1.2 Window functions.

Because of the assumed finite extent of the windows, the summation in Eq. (5.1.3) can be extended to run over all N observations, as is often done in the literature:

$$\mathcal{J}_t = \sum_{k=0}^{N-1}\big(y(t_k) - p(t_k - t)\big)^2 w(t_k - t) = \min\,, \quad p(t_k - t) = \sum_{r=0}^{d} c_r (t_k - t)^r \tag{5.1.6}$$

We prefer the form of Eq. (5.1.3) because it emphasizes the local nature of the fitting window. Let N_t be the number of observations that fall within the interval $[t-h_t, t+h_t]$. We may cast the performance index (5.1.3) in a compact matrix form by defining the $N_t{\times}1$ vector of observations $\mathbf{y}_t$, the $N_t{\times}(d{+}1)$ basis matrix S_t, and the $N_t{\times}N_t$ diagonal matrix of weights by

$$\begin{aligned} \mathbf{y}_t &= [\cdots, y(t_k), \cdots]^T, \qquad \text{for } t - h_t \le t_k \le t + h_t \\ S_t &= \begin{bmatrix} \vdots & \vdots & & \vdots & & \vdots \\ 1 & (t_k - t) & \cdots & (t_k - t)^r & \cdots & (t_k - t)^d \\ \vdots & \vdots & & \vdots & & \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ \mathbf{u}^T(t_k - t) \\ \vdots \end{bmatrix} \\ W_t &= \operatorname{diag}\big([\cdots, w(t_k - t), \cdots]\big) \end{aligned} \tag{5.1.7}$$

where $\mathbf{u}^T(t_k - t)$ is the k-th row of S_t, defined in terms of the $(d{+}1)$-dimensional vector $\mathbf{u}^T(\tau) = [1, \tau, \tau^2, \dots, \tau^d]$. For example, if $t{-}h_t < t_3 < t_4 < t_5 < t_6 < t{+}h_t$, then $N_t = 4$ and for a polynomial order $d = 2$, we have:

$$\mathbf{y}_t = \begin{bmatrix} y(t_3) \\ y(t_4) \\ y(t_5) \\ y(t_6) \end{bmatrix}, \quad S_t = \begin{bmatrix} 1 & (t_3 - t) & (t_3 - t)^2 \\ 1 & (t_4 - t) & (t_4 - t)^2 \\ 1 & (t_5 - t) & (t_5 - t)^2 \\ 1 & (t_6 - t) & (t_6 - t)^2 \end{bmatrix}$$

$$W_t = \begin{bmatrix} w(t_3 - t) & 0 & 0 & 0 \\ 0 & w(t_4 - t) & 0 & 0 \\ 0 & 0 & w(t_5 - t) & 0 \\ 0 & 0 & 0 & w(t_6 - t) \end{bmatrix}$$

With these definitions, Eq. (5.1.3) can be written as

$$\boxed{\mathcal{J}_t = (\mathbf{y}_t - S_t\mathbf{c})^T W_t(\mathbf{y}_t - S_t\mathbf{c}) = \min} \tag{5.1.8}$$

with solution for the coefficient vector $\mathbf{c} = [c_0, c_1, \dots, c_d]^T$:

$$\boxed{\mathbf{c} = (S_t^T W_t S_t)^{-1} S_t^T W_t \mathbf{y}_t} \tag{5.1.9}$$

The quantity $\hat{\mathbf{y}}_t = S_t\mathbf{c}$ represents the polynomial estimate of the local observation vector $\mathbf{y}_t$. It can be written as

$$\boxed{\hat{\mathbf{y}}_t = B_t^T \mathbf{y}_t\,, \quad B_t = W_t S_t (S_t^T W_t S_t)^{-1} S_t^T} \tag{5.1.10}$$

where the $N_t \times N_t$ matrix B_t generalizes (4.1.5), and satisfies a similar polynomial-preserving property as (4.1.6),

$$B_t^T S_t = S_t \tag{5.1.11}$$

Defining the usual $(d+1)$-dimensional unit vector $\mathbf{u}_0 = [1, 0, \dots, 0]^T$, we obtain the estimated value at time t by $\hat{x}_t = c_0 = \mathbf{u}_0^T\mathbf{c}$, and the first derivative by $\hat{\dot{x}}_t = c_1 = \mathbf{u}_1^T\mathbf{c}$, where $\mathbf{u}_1 = [0, 1, 0, \dots, 0]^T$,

$$\boxed{\begin{aligned} \hat{x}_t &= \mathbf{u}_0^T (S_t^T W_t S_t)^{-1} S_t^T W_t \mathbf{y}_t \\ \hat{\dot{x}}_t &= \mathbf{u}_1^T (S_t^T W_t S_t)^{-1} S_t^T W_t \mathbf{y}_t \end{aligned}} \tag{5.1.12}$$

Thus, the effective estimation weights are:

$$\mathbf{h}(t) = W_t S_t (S_t^T W_t S_t)^{-1} \mathbf{u}_0\,, \quad \hat{x}_t = \mathbf{h}^T(t)\mathbf{y}_t \tag{5.1.13}$$

Component-wise, we can write:

$$\hat{x}_t = \mathbf{h}^T(t)\mathbf{y}_t = \sum_{|t_k - t| \le h_t} h_k(t) y_k \tag{5.1.14}$$

where $y_k = y(t_k)$ and

$$\boxed{h_k(t) = w(t_k - t)\mathbf{u}^T(t_k - t)\,(S_t^T W_t S_t)^{-1}\mathbf{u}_0} \tag{5.1.15}$$

We note that $\mathbf{u}_0, \mathbf{u}_1$ are related to the vector $\mathbf{u}(\tau)$ and its derivative by $\mathbf{u}_0 = \mathbf{u}(0)$ and $\mathbf{u}_1 = \dot{\mathbf{u}}(0)$. We also have,

$$S_t^T W_t S_t = \sum_{|t_k - t| \le h_t} \mathbf{u}(t_k - t)\mathbf{u}^T(t_k - t) w(t_k - t) \tag{5.1.16}$$

or, component-wise,

$$(S_t^T W_t S_t)_{ij} = \sum_{|t_k - t| \le h_t} (t_k - t)^{i+j} w(t_k - t)\,, \quad i,j = 0, 1, \dots, d \tag{5.1.17}$$

The solution is particularly easy in the special cases $d = 0$, corresponding to local constant fitting, and $d = 1$, corresponding to local linear fits. The case $d = 0$ leads to the so-called *kernel smoothing* approach first proposed by Nadaraya and Watson [188,189]. In this case $\mathbf{u}(\tau) = [1]$ and we find:

$$S_t^T W_t S_t = \sum_{|t_k - t| \le h_t} w(t_k - t)\,, \quad h_k(t) = \frac{w(t_k - t)}{\displaystyle\sum_{|t_k - t| \le h_t} w(t_k - t)}$$

$$\hat{x}_t = \sum_{|t_k - t| \le h_t} h_k(t) y_k = \frac{\displaystyle\sum_{|t_k - t| \le h_t} w(t_k - t) y_k}{\displaystyle\sum_{|t_k - t| \le h_t} w(t_k - t)} \qquad \text{(kernel smoothing)} \tag{5.1.18}$$

For $d = 1$, we have $\mathbf{u}(\tau) = [1, \tau]^T$, and we obtain

$$S_t^T W_t S_t = \sum_{|t_k - t| \le h_t} \begin{bmatrix} 1 & (t_k - t) \\ (t_k - t) & (t_k - t)^2 \end{bmatrix} w(t_k - t) \equiv \begin{bmatrix} s_0(t) & s_1(t) \\ s_1(t) & s_2(t) \end{bmatrix}$$

$$(S_t^T W_t S_t)^{-1} = \frac{1}{s_0(t) s_2(t) - s_1^2(t)} \begin{bmatrix} s_2(t) & -s_1(t) \\ -s_1(t) & s_0(t) \end{bmatrix}$$

which gives for the filter weights $h_k(t)$:

$$h_k(t) = w(t_k - t)\,\frac{s_2(t) - (t_k - t)s_1(t)}{s_0(t) s_2(t) - s_1^2(t)} \qquad \text{(locally linear fits)} \tag{5.1.19}$$

In general, the invertibility of $S_t^T W_t S_t$ requires that $N_t \ge d + 1$. The QR factorization can be used to implement the above computations efficiently. If the weight function $W(u)$ vanishes at the end-points $u = \pm 1$, as in the tricube case, then the window interval must exclude those end-points. In other words, the diagonal entries of W_t are assumed to be strictly positive. Defining U to be the diagonal square root factor of W_t and carrying out the QR factorization of the matrix US_t, we obtain the efficient algorithm:

$$\begin{aligned} & U = \text{sqrt}(W_t)\,, \quad U \text{ is diagonal so that } U^T = U \text{ and } W_t = U^T U = U^2 \\ & US_t = QR\,, \quad Q^T Q = I_{d+1}\,, \quad R = (d{+}1) \times (d{+}1) \text{ upper-triangular} \\ & \mathbf{c} = R^{-1} Q^T U \mathbf{y}_t \end{aligned} \tag{5.1.20}$$

The above steps are equivalent to reducing the problem to an ordinary unweighted least-squares problem, that is, $\mathbf{c}$ is recognized to be the unique least-squares solution of the full-rank, overdetermined, $N_t \times (d+1)$-dimensional system $(US_t)\mathbf{c} = U\mathbf{y}_t$. Indeed, it follows from Eq. (15.4.10) of Chap. 15 that $\mathbf{c}$ is given by:

$$\mathbf{c} = \big[(US_t)^T (US_t)\big]^{-1} (US_t)^T (U\mathbf{y}_t) = (S_t^T W_t S_t)^{-1} S_t^T W_t \mathbf{y}_t \tag{5.1.21}$$

where $\big[(US_t)^T (US_t)\big]^{-1} (US_t)^T$ is the pseudoinverse of US_t. The corresponding performance indices are equivalent:

$$\mathcal{J}_t = (\mathbf{y}_t - S_t\mathbf{c})^T W_t (\mathbf{y}_t - S_t\mathbf{c}) = \|U\mathbf{y}_t - US_t\mathbf{c}\|^2 = \min$$

In MATLAB the least-squares solution (5.1.21) can be obtained by the backslash operation: $\mathbf{c} = (US_t)\backslash(U\mathbf{y}_t)$. The construction of the quantities $\mathbf{y}_t, S_t, W_t$ is straightforward. Given the column vectors of observation times and observations,

$$t_{\text{obs}} = [t_0, t_1, \dots, t_{N-1}]^T, \quad y_{\text{obs}} = [y(t_0), y(t_1), \dots, y(t_{N-1})]^T \tag{5.1.22}$$

we may determine, with the help of `locw`, the column vector of indices k for which t_k lies in the local window, and then carry out the procedure (5.1.21):

```
w = locw((tobs - t)/h_t, type);    % weights of all observation times relative to a given t and h_t
k = find(w);                       % column vector of indices of nonzero weights within window
yt = yobs(k);                      % column vector of corresponding local observations y_t
Wt = diag(w(k));                   % diagonal matrix of nonzero local weights W_t
St = [];
for r=0:d,
   St = [St, (tobs(k) - t).^r];    % construct local polynomial basis S_t column-wise
end
U = sqrt(Wt);                      % diagonal square root of W_t
c = (U*St)\(U*yt);                 % least-squares solution
```

Most of the w's obtained from the first line of code are zero, except for those t_k that lie within the local window $t \pm h_t$. The second line, `k = find(w)`, finds the latter. These steps have been incorporated into the MATLAB function `locpol`:

```
[xhat,C] = locpol(tobs,yobs,t,h,d,type);    % local polynomial modeling
```

where `tobs,yobs` are as in (5.1.22), `t,h` are L-dimensional vectors of times and bandwidths at which to carry out the fit, and `d,type` are the polynomial order and window type, with default values $d = 1$, `type` $= 1$. The output `xhat` is the L-dimensional vector of estimates $\hat{x}_t$, and `C` is an $L\times(d+1)$ matrix, whose ith row is the vector $[c_0, c_1, \dots, c_d]$ of polynomial coefficients corresponding to the ith fitting time and bandwidth $t(i), h(i)$. Thus, the first column of `C` is the same as `xhat`, while the second column contains the first derivatives. Separating the first column of `C` into `xhat` is done only for convenience in using the function.

The choice of the bandwidth h_t is an important consideration that influences the quality of the estimate $\hat{x}_t$. Too large an h_t will oversmooth the signal but reduce the noise (i.e., increasing bias but lowering variance), and too small an h_t will undersmooth the signal and not reduce the noise as much (i.e., reducing bias and increasing variance).

Two simple bandwidth choices are the *fixed* and the *nearest-neighbor* bandwidths. In the fixed case, one chooses the same bandwidth at each fitting time, that is, $h_t = h$, for all t. In the nearest-neighbor case, one chooses a fixed number, say K, of observations to lie within each local window, where K is a fraction of the total number of observations N, that is, $K = \lfloor \alpha N \rfloor$, truncated to an integer, where $\alpha \leq 1$. Typical values are $\alpha = 0.2$–0.8. Given K, one calculates the distances of all the observation times from t, that is, $|t_k - t|$, $k = 0, 1, \dots, N-1$, then sorts them in increasing order, and picks h_t to be the Kth shortest distance, and therefore, there will be K observations satisfying $|t_k - t| \leq h_t$. In summary, the fixed case selects $h_t = h$ but with varying N_t, and the nearest-neighbor case selects varying h_t but with fixed $N_t = K$.

The MATLAB function `locband` may be used to calculate the bandwidths h_t at each t, using either the fixed method, or the nearest-neighbor method:

```
h = locband(tobs,t,alpha,h0);    % bandwidth for local polynomial regression
```

where if $\alpha = 0$, the fixed bandwidth h_0 is selected, and if $0 < \alpha < 1$, the K-nearest bandwidths are selected, where t is a length-L vector of fitting times.

Example 5.1.1: As an example, consider the following 16 observation times t_{obs}, and 5 fitting times t, and choose $\alpha = 0.25$ so that $K = \alpha N = 0.25\times 16 = 4$:

$$t_{\text{obs}} = [0.5,\ 0.8,\ 1.1,\ 1.2,\ 1.8,\ 2.4,\ 2.5,\ 3.4,\ 3.5,\ 3.7,\ 4.0,\ 4.2,\ 4.9,\ 5.0,\ 5.1,\ 6.2]$$
$$t = [0.5,\ 1.5,\ 2.9,\ 3.6,\ 5.1]$$

then one finds the corresponding bandwidths for each of the five t's

$$\texttt{h = locband(tobs,t,0.25,0)} = [0.7,\ 0.7,\ 0.6,\ 0.6,\ 0.9]$$

and the corresponding local intervals, each containing $K = 4$ observation times:

h_t	$t - h_t$	t	$t + h_t$	included t_ks
0.7	−0.2	0.5	1.2	0.5, 0.8, 1.1, 1.2
0.7	0.8	1.5	2.2	0.8, 1.1, 1.2, 1.8
0.6	2.3	2.9	3.5	2.4, 2.5, 3.4, 3.5
0.6	3.5	4.1	4.7	3.5, 3.7, 4.0, 4.2
0.9	4.2	5.1	6.0	4.2, 4.9, 5.0, 5.1

By contrast, had we chosen a fixed bandwidth, say $h = 0.7$ (the average of the above five), then the corresponding intervals and included observation times would have been:

h_t	$t - h_t$	t	$t + h_t$	included t_ks
0.7	−0.2	0.5	1.2	0.5, 0.8, 1.1, 1.2
0.7	0.8	1.5	2.2	0.8, 1.1, 1.2, 1.8
0.7	2.2	2.9	3.6	2.4, 2.5, 3.4, 3.5
0.7	2.9	3.6	4.3	3.4, 3.5, 3.7, 4.0, 4.2
0.7	4.4	5.1	5.8	4.9, 5.0, 5.1

where now the number N_t of included observations depends on t. As can be seen from this example, both the nearest-neighbor and fixed bandwidth choices adapt well at the end-points of the available observations. □

Choosing t to be one of the observation times, $t = t_i$, Eq. (5.1.12) can be written in the simplified notation:

$$\hat{x}_i = \mathbf{u}_0^T (S_i^T W_i S_i)^{-1} S_i^T W_i \mathbf{y}_i \equiv \mathbf{h}_i^T \mathbf{y}_i\,, \quad \mathbf{h}_i^T = \mathbf{u}_0^T (S_i^T W_i S_i)^{-1} S_i^T W_i \tag{5.1.23}$$

where $\hat{x}_i, S_i, W_i, \mathbf{y}_i$ are the quantities $\hat{x}_t, S_t, W_t, \mathbf{y}_t$ evaluated at $t = t_i$. Component-wise,

$$\hat{x}_i = \sum_{|t_j - t_i| \leq h_i} \mathbf{u}_0^T (S_i^T W_i S_i)^{-1} \mathbf{u}(t_j - t_i)\, w(t_j - t_i)\, y_j = \sum_{|t_j - t_i| \leq h_i} H_{ij}\, y_j \tag{5.1.24}$$

where the matrix elements H_{ij} are,

$$H_{ij} = h_j(t_i) = \mathbf{u}_0^T (S_i^T W_i S_i)^{-1} \mathbf{u}(t_j - t_i)\, w(t_j - t_i) \tag{5.1.25}$$

Similarly, one may express $S_i^T W_i S_i$ and $S_i^T W_i \mathbf{y}_i$ as,

$$\begin{aligned} S_i^T W_i S_i &= \sum_{|t_j-t_i|\le h_i} \mathbf{u}(t_j-t_i)\mathbf{u}^T(t_j-t_i)\,w(t_j-t_i) \\ S_i^T W_i \mathbf{y}_i &= \sum_{|t_j-t_i|\le h_i} \mathbf{u}(t_j-t_i)\,w(t_j-t_i)\,y_j \end{aligned} \tag{5.1.26}$$

Because the factor $w(t_j-t_i)$ vanishes outside the local window $t_i \pm h_i$, the summations in (5.1.24) and (5.1.26) over t_j can be extended to run over all N observations. Defining the N-dimensional vectors $\hat{\mathbf{x}} = [\hat{x}_0, \hat{x}_1, \dots, \hat{x}_{N-1}]^T$ and $\mathbf{y} = [y_0, y_1, \dots, y_{N-1}]^T$, we may write (5.1.24) in the compact matrix form:

$$\hat{\mathbf{x}} = H\mathbf{y} \tag{5.1.27}$$

The filtering matrix H is also known as the "hat" matrix or the "smoothing" matrix. Its diagonal elements H_{ii} play a special role in bandwidth selection, where $w_0 = w(0)$,[†]

$$H_{ii} = h_i(t_i) = w_0\,\mathbf{u}_0^T(S_i^T W_i S_i)^{-1}\mathbf{u}_0 \tag{5.1.28}$$

5.2 Bandwidth Selection

There exist various automatic schemes for choosing the bandwidth. Such schemes may at best be used as guidelines. Ultimately, one must rely on one's judgment in making the final choice.

A popular bandwidth selection method is the so-called *cross-validation* criterion that selects the bandwidth h that minimizes the sum of squared prediction errors:

$$\text{CV}(h) = \frac{1}{N}\sum_{i=0}^{N-1}(y_i - \hat{x}_i^-)^2 = \min \tag{5.2.1}$$

where $\hat{x}_i^-$ is the estimate or prediction of the sample $x_i = x(t_i)$ obtained by *deleting* the ith observation y_i and basing the estimation on the remaining observations, where we are assuming the usual additive-noise model $y(t_i) = x(t_i) + v(t_i)$ with $x(t_i)$ representing the desired signal to be extracted from $y(t_i)$. We show below that the predicted estimate $\hat{x}_i^-$ is related to the estimate $\hat{x}_i$ based on all observations by the relationship:

$$\hat{x}_i^- = \frac{\hat{x}_i - H_{ii}y_i}{1 - H_{ii}} \tag{5.2.2}$$

where H_{ii} is given by (5.1.28). It follows from (5.2.2) that the corresponding estimation errors will be related by:

$$y_i - \hat{x}_i^- = \frac{y_i - \hat{x}_i}{1 - H_{ii}} \tag{5.2.3}$$

[†] $w_0 = 1$ for all the windows in Eq. (5.1.5), but any other normalization can be used.

and therefore, the CV index can be expressed as:

$$\text{CV}(h) = \frac{1}{N}\sum_{i=0}^{N-1}(y_i - \hat{x}_i^-)^2 = \frac{1}{N}\sum_{i=0}^{N-1}\left(\frac{y_i - \hat{x}_i}{1 - H_{ii}}\right)^2 = \min \tag{5.2.4}$$

A related selection criterion is based on the *generalized cross-validation* index, which replaces H_{ii} by its average over i, that is,

$$\text{GCV}(h) = \frac{1}{N}\sum_{i=0}^{N-1}\left(\frac{y_i - \hat{x}_i}{1 - \bar{H}}\right)^2 = \min\,, \quad \bar{H} = \frac{1}{N}\sum_{i=0}^{N-1}H_{ii} = \frac{1}{N}\,\text{tr}(H) \tag{5.2.5}$$

If the bandwidth is to be selected by the nearest-neighbor method, then, the CV and GCV indices may be regarded as functions of the fractional parameter α and minimized. Similarly, one could consider minimizing with respect to the polynomial order d, although in practice d is usually chosen to be 1 or 2.

Eq. (5.2.2) can be shown as follows. If the $t_j = t_i$ observation is deleted from Eq. (5.1.23), the corresponding optimum polynomial coefficients and optimum estimate will be given by

$$\mathbf{c}_- = (S_i^T W_i S_i)_-^{-1}(S_i^T W_i \mathbf{y}_i)_-\,, \quad \hat{x}_i^- = \mathbf{u}_0^T\mathbf{c}_-$$

where the minus subscripts indicate that the $t_j = t_i$ terms are to be omitted. It follows from Eq. (5.1.26) that

$$\begin{aligned} S_i^T W_i S_i &= (S_i^T W_i S_i)_- + w_0\mathbf{u}_0\mathbf{u}_0^T \\ S_i^T W_i \mathbf{y}_i &= (S_i^T W_i \mathbf{y}_i)_- + w_0\mathbf{u}_0 y_i \end{aligned} \tag{5.2.6}$$

and then,

$$\mathbf{c}_- = [S_i^T W_i S_i - w_0\mathbf{u}_0\mathbf{u}_0^T]^{-1}[S_i^T W_i \mathbf{y}_i - w_0\mathbf{u}_0 y_i] \tag{5.2.7}$$

Setting $F_i = S_i^T W_i S_i$ and noting that $\mathbf{c} = F_i^{-1}S_i^T W_i \mathbf{y}_i$ or $S_i^T W_i \mathbf{y}_i = F_i\mathbf{c}$, we may write,

$$\mathbf{c}_- = [F_i - w_0\mathbf{u}_0\mathbf{u}_0^T]^{-1}[F_i\mathbf{c} - w_0\mathbf{u}_0 y_i]$$

Using the matrix inversion lemma, we have,

$$[F_i - w_0\mathbf{u}_0\mathbf{u}_0^T]^{-1} = F_i^{-1} + \frac{w_0F_i^{-1}\mathbf{u}_0\mathbf{u}_0^T F_i^{-1}}{1 - w_0\mathbf{u}_0^T F_i^{-1}\mathbf{u}_0} \tag{5.2.8}$$

Noting that $H_{ii} = w_0\mathbf{u}_0^T F_i^{-1}\mathbf{u}_0$, we obtain,

$$\begin{aligned} \mathbf{c}_- &= \left[F_i^{-1} + \frac{w_0F_i^{-1}\mathbf{u}_0\mathbf{u}_0^T F_i^{-1}}{1 - H_{ii}}\right][F_i\mathbf{c} - w_0\mathbf{u}_0 y_i] \\ &= \left[I + \frac{w_0F_i^{-1}\mathbf{u}_0\mathbf{u}_0^T}{1 - H_{ii}}\right][\mathbf{c} - w_0F_i^{-1}\mathbf{u}_0 y_i] \\ &= \mathbf{c} - w_0F_i^{-1}\mathbf{u}_0 y_i + \frac{w_0F_i^{-1}\mathbf{u}_0[\mathbf{u}_0^T\mathbf{c} - w_0\mathbf{u}_0^T F_i^{-1}\mathbf{u}_0 y_i]}{1 - H_{ii}} \end{aligned}$$

and since $\hat{x}_i = \mathbf{u}_0^T\mathbf{c}$, we find,

$$\mathbf{c}_- = \mathbf{c} - w_0F_i^{-1}\mathbf{u}_0 y_i + \frac{w_0F_i^{-1}\mathbf{u}_0[\hat{x}_i - H_{ii}y_i]}{1 - H_{ii}} = \mathbf{c} + w_0F_i^{-1}\mathbf{u}_0\,\frac{\hat{x}_i - y_i}{1 - H_{ii}}$$

from which we find for $\hat{x}_i^- = \mathbf{u}_0^T \mathbf{c}_-$,

$$\hat{x}_i^- = \hat{x}_i + \frac{H_{ii}(\hat{x}_i - y_i)}{1 - H_{ii}} = \frac{\hat{x}_i - H_{ii} y_i}{1 - H_{ii}} \tag{5.2.9}$$

In practice, the CV and GCV indices are evaluated over a certain range of the smoothing parameter h or α to look for a minimum. The MATLAB function `locgcv` evaluates these indices at any vector of parameter values:

```
[GCV,CV] = locgcv(tobs,yobs,d,type,b,btype);      % CV and GCV evaluation
```

where `type` is the window type, `b` is either a vector of hs or a vector of αs at which to evaluate CV and GCV, and the string `btype` takes the values `'f'` or `'nn'` specifying whether the parameter vector `b` corresponds to a fixed or nearest-neighbor bandwidth.

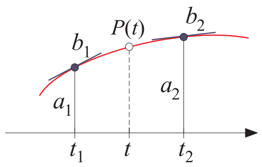
5.3 Local Polynomial Interpolation

The primary advantage of local polynomial modeling is its flexibility and ease of smoothing unequally-spaced data. Its main disadvantage is the potentially high computational cost, that is, the calculations (5.1.12) must be performed for each t, and generally a dense set of such t's might be required in order to get a visually smooth curve.

One way to cut down the cost is to evaluate the smoothed values $\hat{x}_t$ at a less dense grid of ts, and then interpolate smoothly between the computed points. This is akin to what plotting programs do by connecting the dots by straight-line segments (linearly interpolating)—the result being a visually continuous curve. But here, we can do better than just connecting the dots because we have available the slopes at each grid point. These slopes are contained in the second column of the fitting matrix C resulting from `locpol`, assuming of course that $d \geq 1$.

Consider two time instants t_1, t_2 at which the fitted signal values are a_1, a_2 with corresponding slopes b_1, b_2, as shown below. The lowest-degree polynomial $P(t)$ interpolating between the two points t_1, t_2 that matches the fitted values and their slopes at t_1 and t_2 is a cubic polynomial—the method being known as cubic Hermite interpolation. The four polynomial coefficients are fixed uniquely by the four conditions:

$$\begin{aligned} P(t_1) &= a_1\,, \quad \dot{P}(t_1) = b_1 \\ P(t_2) &= a_2\,, \quad \dot{P}(t_2) = b_2 \end{aligned}$$



which result into the cubic polynomial, where $T = t_2 - t_1$,

$$\begin{aligned} P(t) = \left(\frac{t - t_2}{T}\right)^2 &\left[a_1 + (Tb_1 + 2a_1)\left(\frac{t - t_1}{T}\right)\right] \\ + \left(\frac{t - t_1}{T}\right)^2 &\left[a_2 + (Tb_2 - 2a_2)\left(\frac{t - t_2}{T}\right)\right] \end{aligned} \tag{5.3.1}$$

For local-polynomial orders $d \geq 1$, we use Eq. (5.3.1) to interpolate at a denser grid of points between the less dense grid of fitted points. For the special case, $d = 0$, the slopes are not available and we can only use linear interpolation, that is,

$$P(t) = a_1 + (a_2 - a_1)\left(\frac{t - t_1}{T}\right) \tag{5.3.2}$$

The MATLAB function `locval` takes the output matrix `C` from `locpol` corresponding to a grid of fitting points t, and computes the interpolated points y_{grid} at the denser grid of points t_{grid}:

```
ygrid = locval(C,t,tgrid);      % interpolating local polynomial fits
```

The auxiliary function `locgrid` helps establish a uniform grid between the t points:

```
tgrid = locgrid(t, Ngrid);      % uniform grid with respect to t
```

which is simply a shorthand for,

```
tgrid = linspace(min(t), max(t), Ngrid);
```

Example 5.3.1: The motorcycle acceleration dataset [231] has served as a benchmark in many studies of local polynomial modeling and spline smoothing. The ordinate represents head acceleration (in units of g) during impact, and the abscissa is the time (in msec).

Fig. 5.3.1 shows a plot of the GCV index as a function of the nearest-neighbor fractional parameter α on the left, and as a function of the fixed bandwidth h on the right, for the two polynomial orders $d = 1, 2$.

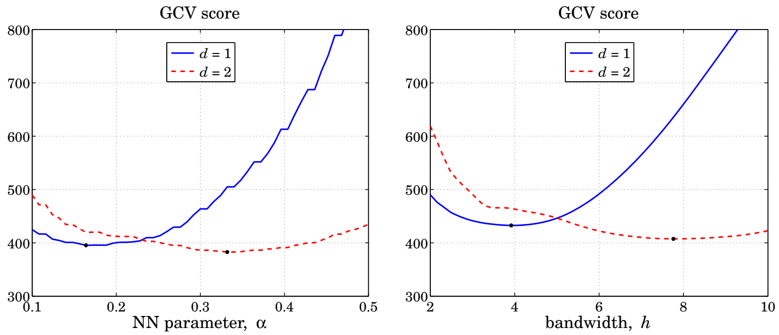


Fig. 5.3.1 GCV score for nearest-neighbor (left) and fixed bandwidths (right).

The "optimal" values of these parameters that minimize the GCV (and indicated by dots on the graphs) are as follows, where the subscripts indicate the value of d:

$$\alpha_1 = 0.16\,, \quad \alpha_2 = 0.33\,, \quad h_1 = 3.9\,, \quad h_2 = 7.8$$

The graphs (for $d = 1$) were produced by the MATLAB code:

```
Y = loadfile('mcyc.dat');                      % file included in the OSP toolbox
tobs = Y(:,1);  yobs = Y(:,2);                 % 133 data points

alpha = linspace(0.1, 0.5, 51);                % vary over 0.1 ≤ α ≤ 0.5

d=1; type=1;
gcv = locgcv(tobs,yobs,d,type,alpha,'nn');     % GCV as function of α
[F,i] = min(gcv);  alpha1 = alpha(i);          % minimum at α = α1

figure; plot(alpha,gcv);                       % left graph

h = linspace(2, 10, 51);                       % vary over 2 ≤ h ≤ 10
gcv = locgcv(tobs,yobs,d,type,h,'f');          % GCV as function of h
[F,i] = min(gcv);  h1 = h(i);                  % minimum at h = h1
```

Fig. 5.3.2 shows the local polynomial fits corresponding to the above optimal parameter values. The left graph shows the nearest-neighbor cases for $d = 1, 2$, and the right graph, the fixed bandwidth cases. The tricube window was used (`type=1`).

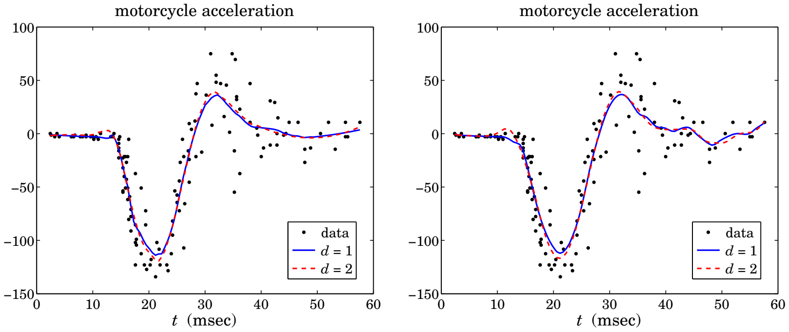


Fig. 5.3.2 Nearest-neighbor (left) and fixed bandwidths (right).

In all cases, the actual fitting was performed at 100 equally-spaced points t within the observation range t_{obs} and were connected by straight-line segments by the plotting program, instead of being interpolated by `locval`. Continuing with the above MATLAB code, the graphs were generated by

```
t = locgrid(tobs,101);                             % equally-spaced fitting times

h = locband(tobs, t, alpha1, 0);                   % NN bandwidths at each t
x1 = locpol(tobs,yobs,t,h,1,type);                 % fit at times t with d = 1

h = locband(tobs, t, alpha2, 0);
x2 = locpol(tobs,yobs,t,h,2,type);                 % fit at times t with d = 2

figure; plot(tobs,yobs,'.', t,x1,'-', t,x2,'--');  % left graph

h = locband(tobs, t, 0, h1);                       % fixed bandwidths at each t
x1 = locpol(tobs,yobs,t,h,1,type);                 % fit at times t with d = 1

h = locband(tobs, t, 0, h2);
```

```
x2 = locpol(tobs,yobs,t,h,2,type);                 % fit at times t with d = 2

figure; plot(tobs,yobs,'.', t,x1,'-', t,x2,'--');  % right graph
```

Fig. 5.3.3 demonstrates the Hermite interpolation procedure. The fitting times are 20 equally-spaced points spanning the observation interval t_{obs}. The 20 fitted points are then interpolated at 100 equally-spaced points over t_{obs}. The interpolated curves are essentially identical to those fitted earlier at 100 points.

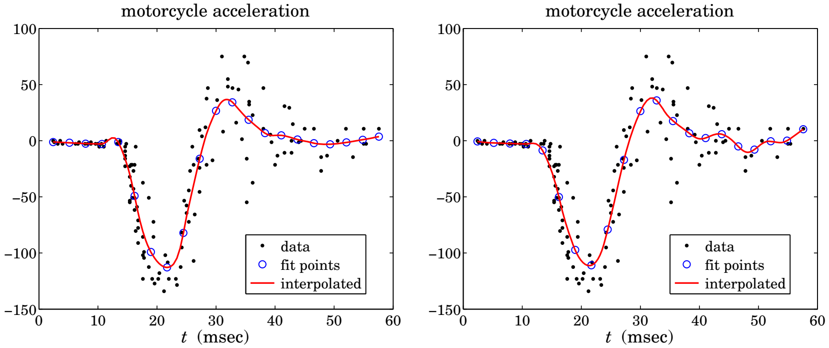


Fig. 5.3.3 Nearest-neighbor (left) and fixed bandwidths (right).

The polynomial order was $d = 1$ and the bandwidth parameters were $\alpha_1 = 0.21$ for the left graph and $h_1 = 4.4$ for the right one. The left graph was generated by the code segment:

```
tf = locgrid(tobs,21);                             % fitting times

h = locband(tobs, tf, alpha1, 0);                  % NN bandwidths at tf
[xf,C] = locpol(tobs,yobs,tf,h,1,type);            % fitted values and derivatives

tint = locgrid(tf,101);                            % interpolation times
xint = locval(C, tf, tint);                        % interpolated values

figure; plot(tobs,yobs,'.', tf,xf,'o', tint,xint,'-');
```

Example 5.3.2: The ethanol dataset [230] is also a benchmark example for smoothing techniques. The ordinate NO_x represents nitric oxide concentrations in the engine exhaust gases, and the abscissa E is the equivalence ratio, which is a measure of the richness of the ethanol/air mixture.

The GCV and CV bandwidth selection criteria tend sometimes to result in undersmoothed signals. This can be seen in Fig. 5.3.4 in which the GCV criterion for fixed bandwidth selects the values $h_1 = 0.039$ and $h_2 = 0.058$, for orders $d = 1, 2$.

As can be seen, the resulting fits are jagged, and can benefit form increasing the fitting bandwidth somewhat. The minima of the GCV plot are fairly broad and any neighboring values of the bandwidth would be just as good in terms of the GCV value. A similar effect happens in this example for the nearest-neighbor bandwidth method, in which the GCV criterion selects the value $\alpha = 0.19$ corresponding to jagged graph (not shown). Fig. 5.3.5

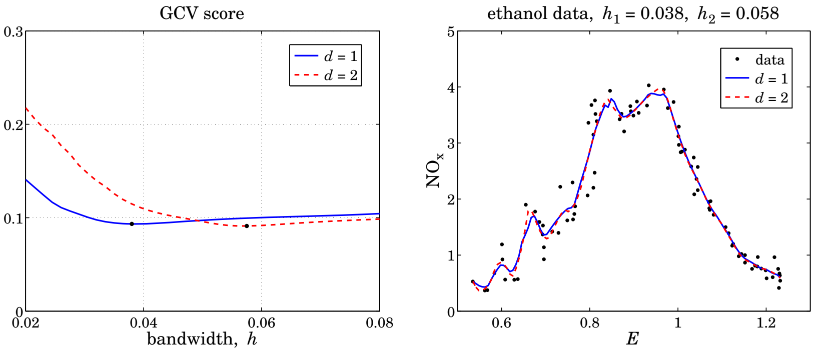


Fig. 5.3.4 GCV and local polynomial fits with $d = 1, 2$.

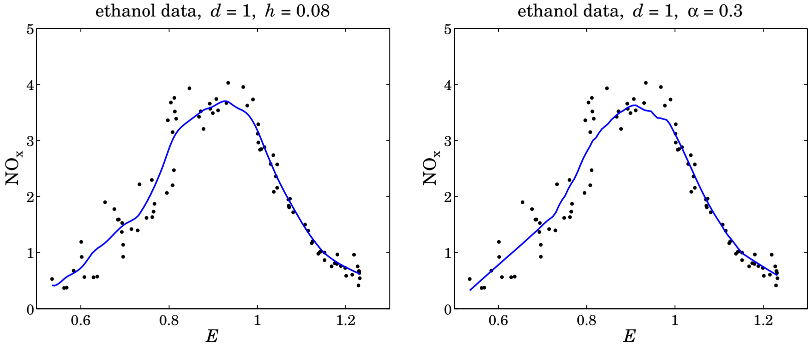


Fig. 5.3.5 Fits with fixed (left) and nearest-neighbor (right) bandwidths.

shows the fits when the fixed bandwidth is increased to $h = 0.08$ and the nearest-neighbor one to $\alpha = 0.3$. The resulting fits are much smoother.

The MATLAB code for generating the graphs of Fig. 5.3.4 is as follows:

```
Y = loadfile('ethanol.dat');                  % file available in OSP toolbox
tobs = Y(:,1);  yobs = Y(:,2);                % data

t = locgrid(tobs,101);                        % uniform grid of 101 fitting points

h = linspace(0.02, 0.08, 41);                 % vary h over 0.02 ≤ h ≤ 0.08
gcv1 = locgcv(tobs,yobs,1,1,h,'f');           % GCV as function of h
gcv2 = locgcv(tobs,yobs,2,1,h,'f');

figure; plot(h,gcv1,'-', h,gcv2,'--');        % left graph

h = locband(tobs, t, 0, h1);                  % fixed bandwidths at t
x1 = locpol(tobs,yobs,t,h,1,1);               % fit with d = 1 and tricube window

h = locband(tobs, t, 0, h2);
```

```
x2 = locpol(tobs,yobs,t,h,2,1);                         % fit with d = 2 and tricube window

figure; plot(tobs,yobs,'.', t,x1,'-', t,x2,'--');       % right graph
```

The MATLAB code for generating Fig. 5.3.5 is as follows:

```
h0 = 0.08; h = locband(tobs, t, 0, h0);                 % fixed bandwidth case
x1 = locpol(tobs,yobs,t,h,1,1);                         % order d = 1, tricube window
figure; plot(tobs,yobs,'.', t,x1,'-');                  % left graph

alpha = 0.3; h = locband(tobs, t, alpha, 0);            % nearest-neighbor bandwidth case
x1 = locpol(tobs,yobs,t,h,1,1); x1 = C(:,1);            % order d = 1, tricube windowm
figure; plot(tobs,yobs,'.', t,x1,'-');                  % right graph
```

Fig. 5.3.6 shows a fit at 10 fitting points and interpolated over 101 points. The fitting parameters are as in the right graph of Fig. 5.3.5. The following code generates Fig. 5.3.6:

```
tf = locgrid(tobs,10);                                  % fitting points
alpha = 0.3; h = locband(tobs, tf, alpha, 0);           % nearest-neighbor bandwidths
[xf,C] = locpol(tobs,yobs,tf,h,1,1);                    % order 1, tricube window

ti = locgrid(tf,101); yi = locval(C,tf,ti);             % interpolated points

figure; plot(tobs,yobs,'.', ti,yi,'-', tf,xf,'o');
```

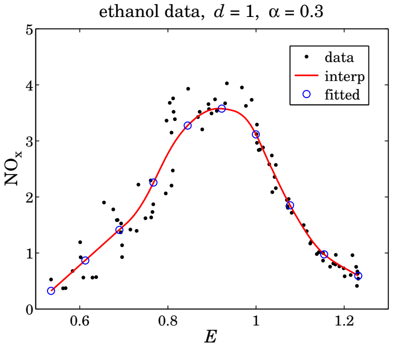


Fig. 5.3.6 Interpolated fits.

5.4 *Variable Bandwidth*

The issue of selecting the right bandwidth has been studied extensively, with approaches ranging from finding an optimum bandwidth that minimizes a selection criterion such as the GCV to using a locally-adaptive criterion that allows the bandwidth to automatically adapt to the local nature of the signal with different bandwidths being used in different parts of the signal [188–231].

There is no selection criterion that is universally successful or universally agreed upon and one must use one's judgment and visual inspection to decide how much smoothing is satisfactory. The basic idea is always to reduce the bandwidth in regions where the curvature of the signal is high in order not to oversmooth.

The function `locpol` can accept a different bandwidth h_t for each fitting time t. As we saw in the above examples, the function `locband` generates such bandwidths for input to `locpol`. However, `locband` generates either fixed or or nearest-neighbor bandwidths and is not adaptive to the local nature of the signal. One could manually, divide the range of the signal in non-overlapping regions and use a different fixed bandwidth in each region. In some cases, as in the Doppler example below, this is possible but in other cases a more automatic way of adapting is desirable.

A naive, but as we see in the examples below, quite effective way is to estimate the curvature, say κ_t, of the signal and define the bandwidth in terms of a suitable decreasing function $h_t = f(\kappa_t)$. We may define the curvature in terms of the estimate of the second derivative of the signal and normalize it to its maximum value:

$$\kappa_t = \frac{|\hat{\ddot{x}}_t|}{\max\limits_t |\hat{\ddot{x}}_t|} \tag{5.4.1}$$

The second derivative $\hat{\ddot{x}}_t$ can be estimated by performing a local polynomial fit with polynomial order $d \geq 2$ using a fixed bandwidth h_0 or a nearest-neighbor bandwidth α. If one could determine a bandwidth range $[h_{\text{min}}, h_{\text{max}}]$ such that h_{max} would provide an appropriate amount of smoothing in certain parts of the signal and h_{min} would be appropriate in regions where the signal appears to have larger curvature, then one may choose $h_{\text{min}} \leq h_0 \leq h_{\text{max}}$, with $h_0 = h_{\text{max}}$ as an initial trial value. An ad hoc but very simple choice for the bandwidth function $f(\kappa_t)$ then could be

$$h_t = h_{\text{max}} \left(\frac{h_{\text{min}}}{h_{\text{max}}}\right)^{\kappa_t} \tag{5.4.2}$$

Other simple choices are possible, for example,

$$h_t = \frac{h_{\text{max}} h_{\text{min}}}{h_{\text{min}} + (h_{\text{max}} - h_{\text{min}})\kappa_t^p}$$

for some power p. Since κ_t varies in $0 \leq \kappa_t \leq 1$, these choices interpolate between h_{max} at $\kappa_t = 0$ when the curvature is small, and h_{min} at $\kappa_t = 1$ when the curvature is large.

We illustrate the use of (5.4.2) with the three examples in Figs. 5.4.1–5.4.3, and we make a different bandwidth choice for Fig. 5.4.4. All four examples have been used as benchmarks in studying wavelet denoising methods [821] and we will be discussing them again in that context in Sec. 10.7.

In all cases, we use a second-order polynomial to determine the curvature, and then perform a locally linear fit ($d = 1$) using the variable bandwidth. Fig. 5.4.1 illustrates the test function "bumps" defined by

$$s(t) = \sum_{i=1}^{11} \frac{a_i}{\left[1 + |t - t_i|/w_i\right]^4}\,, \quad 0 \leq t \leq 1$$

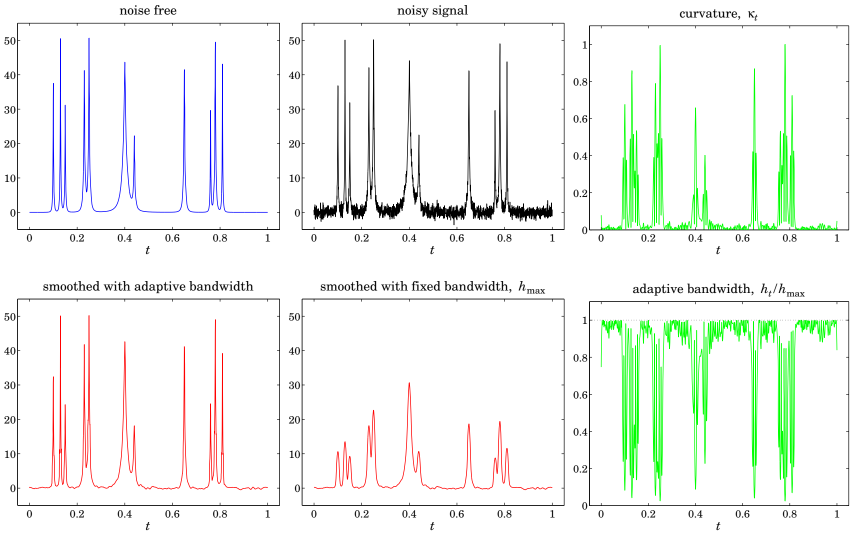


Fig. 5.4.1 Bumps function.

with the parameter values:

$$t_i = [10, 13, 15, 23, 25, 40, 44, 65, 76, 78, 81]/100$$

$$a_i = [40, 50, 30, 40, 50, 42, 21, 43, 31, 51, 42]\cdot 1.0523$$

$$w_i = [5, 5, 6, 10, 10, 30, 10, 10, 5, 8, 5]/1000$$

The function $s(t)$ is sampled at $N = 2048$ equally-spaced points t_n in the interval $[0, 1)$ and zero-mean white gaussian noise of variance $\sigma^2 = 1$ is added so that the noisy signal is $y_n = s_n + v_n$, where $s_n = s(t_n)$. The factor 1.0523 in the amplitudes a_i ensures that the signal-to-noise ratio has the standard benchmark value $\sigma_s/\sigma_v = 7$, where σ_s is the standard deviation of s_n, that is, $\sigma_s = \text{std}(s)$. The bandwidth range is defined by $h_{\text{max}} = 0.01$ and $h_{\text{min}} = 0.00025$. The value for h_{max} was chosen so that the flat portions of the signal between peaks are adequately smoothed.

The curvature κ_t, estimated using the bandwidth $h_0 = h_{\text{max}}$, is shown in the upper right graph. The corresponding variable bandwidth h_t derived from Eq. (5.4.2) is shown in the bottom-right graph. The bottom-left graph shows the resulting local linear fit using the variable bandwidth h_t, while the bottom-middle graph shows the fit using the fixed bandwidth h_{max}. Although h_{max} is adequate for smoothing the valleys of the signal, it is too large for the peaks and results in broadened peaks of reduced heights. On the other hand, the variable bandwidth preserves the peaks fairly well, while achieving comparable smoothing of the valleys. The MATLAB code for this example was as follows:

```
N=2048; t=linspace(0,1,N); s=zeros(size(t));
F = inline('1./(1 + abs(t)).^4');              % bumps function
```

```
ti = [10 13 15 23 25 40 44 65 76 78 81]/100;
ai = [40,50,30,40,50,42,21,43,31,51,42] * 1.0523;
wi = [5,5,6,10,10,30,10,10,5,8,5]/1000;

for i=1:length(ai),                          % construct signal
   s = s + ai(i)*F((t-ti(i))/wi(i));
end

hmax=10e-3; hmin=2.5e-4; h0=hmax;            % bandwidth limits

seed=2009; randn('state',seed);              % noisy signal
y = s + randn(size(t));

d=2; type=1;                                 % fit with d = 2 and tricube window
[x,C] = locpol(t,y,t,h0,d,type);             % using fixed bandwidth h0
kt = abs(C(:,3)); kt = kt/max(kt);           % curvature, κt
ht = hmax * (hmin/hmax).^kt;                 % bandwidth, ht

d=1; type=1;                                 % fit with d = 1
xv = locpol(t,y,t,ht,d,type);                % use variable bandwidth ht
xf = locpol(t,y,t,h0,d,type);                % use fixed bandwidth h0

figure; plot(t,s);  figure; plot(t,y);  figure; plot(t,kt);         % upper graphs
figure; plot(t,xv); figure; plot(t,xf); figure; plot(t,ht/hmax);    % lower graphs
```

Fig. 5.4.2 shows the “blocks” function defined by

$$s(t)=\sum_{i=1}^{11} a_i F(t-t_i)\,,\quad F(t)=\frac{1}{2}(1+\operatorname{sign} t)\,,\quad 0\le t\le 1$$

with the same delays t_i as above and amplitudes:

$$a_i=[40,-50,30,-40,50,-42,21,43,-31,21,-42]\cdot 0.3655$$

The noisy signal is $y_n = s_n + v_n$ with zero-mean unit-variance white noise. The amplitude factor 0.3655 in a_i is adjusted to give the same SNR as above, $\operatorname{std}(s)/\sigma = 7$. The MATLAB code generating the six graphs is identical to the above, except for the part that defines the signal and the bandwidth limits $h_{\max} = 0.03$ and $h_{\min} = 0.0015$:

```
N=2048; t=linspace(0,1,N); s=zeros(size(t));

ti = [10 13 15 23 25 40 44 65 76 78 81]/100;
ai = [40,-50,30,-40,50,-42,21,43,-31,21,-42] * 0.3655;

for i=1:length(ai),
    s = s + ai(i) * (1 + sign(t - ti(i)))/2;          % blocks signal
end

hmax=0.03; hmin=0.0015; h0=hmax;                      % bandwidth limits
```

We observe that the flat parts of the signal are smoothed equally well by the variable and fixed bandwidth choices, but in the fixed case, the edges are smoothed too much. The “HeaviSine” signal shown in Fig. 5.4.3 is defined by

$$s(t)=\big[4\sin(4\pi t)-\operatorname{sign}(t-0.3)-\operatorname{sign}(0.72-t)\big]\cdot 2.357\,,\quad 0\le t\le 1$$

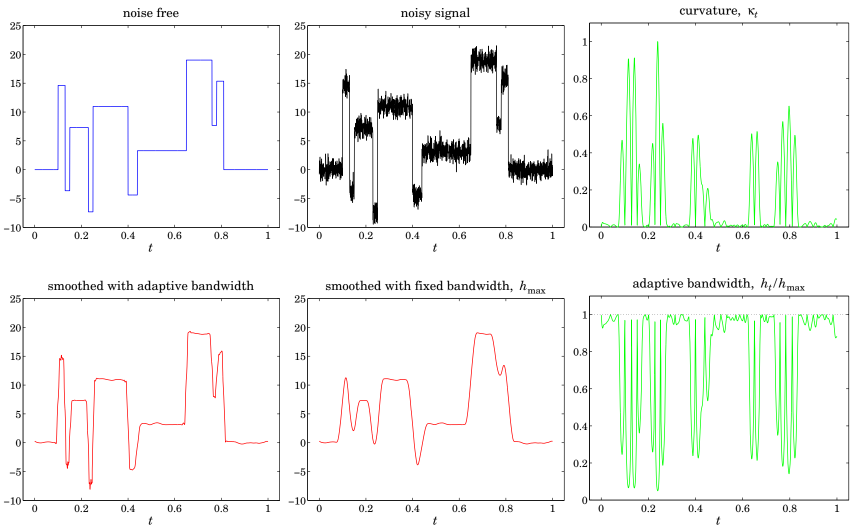


Fig. 5.4.2 Blocks function.

where the factor 2.357 is adjusted to give $\operatorname{std}(s)= 7$. The graphs shown in Fig. 5.4.3 are again generated by the identical MATLAB code, except for the parts defining the signal and bandwidths:

```
s = (4*sin(4*pi*t)-sign(t-0.3)-sign(0.72-t))*2.357;        % HeaviSine signal

hmax=0.035; hmin=0.0035; h0=hmax;                          % bandwidth limits
```

We note that the curvature κ_t is significantly large—and the bandwidth h_t is significantly small—only near the discontinuity points. The fixed bandwidth case smoothes the discontinuities too much, whereas the variable bandwidth tends to preserve them while reducing the noise equally well in the rest of the signal.

In the “doppler” example shown in Fig. 5.4.4, noticing that the curvature κ_t is significantly large only in the range $0 \le t \le 0.2$, we have followed a simpler strategy to define a variable bandwidth (although the choice (5.4.2) still works well). We took a fixed but small bandwidth over the range $0 \le t \le 0.2$ and transitioned gradually to a larger bandwidth for $0.2 \le t \le 1$. The signal is defined by

$$s(t)=24\sqrt{t(1-t)}\,\sin\left(\frac{2.1\pi}{t+0.05}\right)\,,\quad 0\le t\le 1$$

The auxiliary unit-step function `ustep` was used to define the two-step bandwidth with a given rise time. The MATLAB code generating the six graphs was as follows:

```
N = 2048; t = linspace(0,1,N);

s = 24*sqrt(t.*(1-t)) .* sin(2.1*pi./(t+0.05));    % doppler signal
```

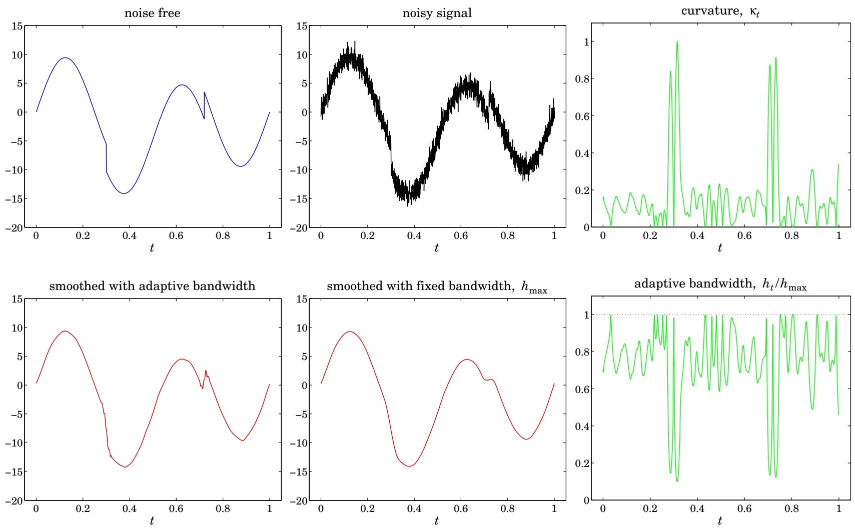


Fig. 5.4.3 HeaviSine function.

```
seed=2009; randn('state',seed);                         % noisy signal
y = s + randn(size(t));
```

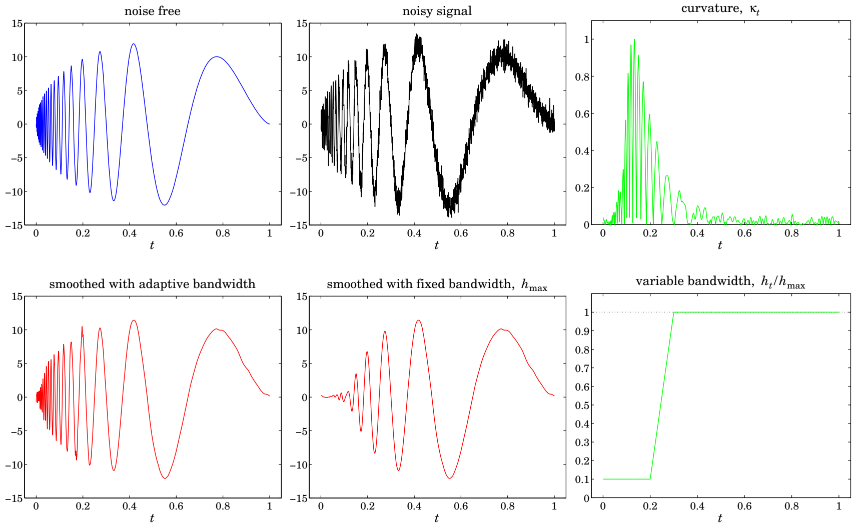


Fig. 5.4.4 Doppler function.

```
hmax=0.02; hmin=0.002; h0=hmax;                     % bandwidth limits

d=2; type=1;                                        % fit with d = 2 and tricube window
[x,C] = locpol(t,y,t,h0,d,type);                    % using fixed bandwidth h0
kt = abs(C(:,3)); kt = kt/max(kt);                  % curvature, κt
ht = hmin + (hmax-hmin) * ustep(t-0.2, 0.1);        % two-step bandwidth, ht
                                                    % ustep is in the OSP toolbox
d=1; type=1;
xv = locpol(t,y,t,ht,d,type);                       % fixed bandwidth h0
xf = locpol(t,y,t,h0,d,type);                       % fixed bandwidth h0

figure; plot(t,s);   figure; plot(t,y);   figure; plot(t,kt);          % upper graphs
figure; plot(t,xv); figure; plot(t,xf); figure; plot(t,ht/hmax);       % lower graphs
```

The local polynomial fitting results from these four benchmark examples are very comparable with the wavelet denoising approach discussed in Sec. 10.7.

5.5 *Repeated Observations*

Until now we had implicitly assumed that the observations were unique, that is, one observation $y(t_k)$ at each time t_k. However, in experimental data one often has repeated observations at a given t_k, all of which are listed as part of the data set. This is in fact true of both the motorcycle and the ethanol data sets. For example, in the motorcycle data, we have six repeated observations at $t = 14.6$,

k	t_k	y_k
$\vdots$	$\vdots$	$\vdots$
22	14.6	−13.3
23	14.6	−5.4
24	14.6	−5.4
25	14.6	−9.3
26	14.6	−16.0
27	14.6	−22.8
$\vdots$	$\vdots$	$\vdots$

and there other similar instances within the data set. In fact, among the 133 given observations, only 94 correspond to unique observation times.

To handle repeated observations one possibility is to simply keep one and ignore the rest—but which one? A better possibility is to allow all of them to be part of the least-squares performance index. It is easy to see that this is equivalent to replacing each group of repeated observations by their average and modifying the weighting function by the corresponding multiplicity of the group.

Let n_k denote the multiplicity of the observations at time t_k, that is, let $y_i(t_k)$, $i = 1, 2, \dots, n_k$ be the observation values that are given at the unique observation time t_k. Then, the performance index (5.1.3) must be modified to include all of the $y_i(t_k)$:

$$\mathcal{J}_t = \sum_{|t_k - t| \le h_t} \sum_{i=1}^{n_k} \big[y_i(t_k) - \mathbf{u}^T(t_k - t)\mathbf{c}\big]^2 w(t_k - t) = \min \tag{5.5.1}$$

Setting the gradient with respect to **c** to zero, gives the normal equations:

$$\sum_{|t_k-t|\le h_t}\sum_{i=1}^{n_k} w(t_k-t)\,\mathbf{u}(t_k-t)\mathbf{u}^T(t_k-t)\,\mathbf{c} = \sum_{|t_k-t|\le h_t} w(t_k-t)\,\mathbf{u}(t_k-t)\sum_{i=1}^{n_k} y_i(t_k)$$

Defining the average of the n_k observations,

$$\bar{y}(t_k) = \frac{1}{n_k}\sum_{i=1}^{n_k} y_i(t_k)$$

and noting that the left-hand side has no dependence on i, we obtain:

$$\sum_{|t_k-t|\le h_t} n_k w(t_k-t)\,\mathbf{u}(t_k-t)\mathbf{u}^T(t_k-t)\mathbf{c} = \sum_{|t_k-t|\le h_t} n_k w(t_k-t)\,\mathbf{u}(t_k-t)\bar{y}(t_k) \tag{5.5.2}$$

This is recognized to be the solution of an equivalent least-squares local polynomial fitting problem in which each unique t_k is weighted by $n_k w_k(t_k - t)$ with the kth observation replaced by the average $\bar{y}(t_k)$, that is,

$$\bar{\mathcal{J}}_t = \sum_{|t_k-t|\le h_t}\big[\bar{y}(t_k) - \mathbf{u}^T(t_k-t)\mathbf{c}\big]^2 n_k w(t_k-t) = \min \tag{5.5.3}$$

Internally, the function `locpol` calls the function `avobs`, which takes in the raw data `tobs,yobs` and determines the unique observation times `ta`, averaged observations `ya`, and their multiplicities `na`:

```
[ta,ya,na] = avobs(tobs,yobs);      % average repeated observations
```

For example, if

```
tobs = [ 1   1   1   3   3   5   5   3   4   7   9   9   9   9];
yobs = [20  22  21  11  12  13  15  19  21  25  28  29  31  32];
```

the function first sorts the ts in increasing order,

```
tobs = [ 1   1   1   3   3   3   4   5   5   7   9   9   9   9];
yobs = [20  21  22  11  12  19  21  13  15  25  28  29  31  32];
```

and then returns the output,

```
ta = [ 1   3   4   5   7   9];
ya = [21  14  21  14  25  30];
na = [ 3   3   1   2   1   4];
```

5.6 *Loess Smoothing*

Loess, which is a shorthand for *lo*cal regr*ess*ion, is a method proposed by Cleveland [192] for handling data with outliers. A version of it was discussed in Sec. 4.5. The method carries out a local polynomial regression using a nearest-neighbor bandwidth and the tricube window function, and then uses the resulting error residuals to iteratively readjust the window weights giving less importance to the outliers.

The method is described as follows [192]. Given the N-dimensional vectors of observation times and observations $t_{\text{obs}}, y_{\text{obs}}$, the nearest-neighbor bandwidth parameter α, and the polynomial order d, the method begins by performing a preliminary fit to all the observation times. For example, in the notation of the `locband` and `locpol` functions:

$$\begin{aligned} h &= \texttt{locband}(t_{\text{obs}}, t_{\text{obs}}, \alpha, 0); &&\text{(find local bandwidths at } t_{\text{obs}})\\ \hat{x} &= \texttt{locpol}(t_{\text{obs}}, y_{\text{obs}}, t_{\text{obs}}, h, d, 1); &&\text{(perform fit at all } t_{\text{obs}}) \end{aligned} \tag{5.6.1}$$

where the last argument of `locpol` designates the use of the tricube window. From the resulting N-dimensional signal $\hat{x}_k$, $k = 0, 1, \dots, N-1$, we calculate the corresponding error residuals e_k and use their median to calculate “robustness” weights r_k:

$$\begin{aligned} e_k &= y_k - \hat{x}_k\,, \quad k = 0, 1, \dots, N-1\\ \mu &= \operatorname*{median}_{0\le k\le N-1}\big(|e_k|\big)\\ r_k &= W\left(\frac{e_k}{6\mu}\right) \end{aligned} \tag{5.6.2}$$

where $W(u)$ is the bisquare function defined in (5.1.5). The local polynomial fitting is now repeated at all observation points t_{obs}, but instead of using the weights $w(t_k - t_{\text{obs}})$ for the kth observation’s contribution to the fit, one uses the modified weights $r_k w(t_k - t_{\text{obs}})$. The new residuals are then computed as in (5.6.2) and the process is repeated a few more times or until convergence (i.e., until the estimated signal $\hat{x}_k$ no longer changes).

After the final iteration resulting in the final values of the r_ks, one can carry out the fit at any other time point t, but again using weights $r_k w(t_k - t)$ for the contribution of the kth observation, that is, the weight matrix W_t in Eq. (5.1.7) is replaced by

$$W_t = \operatorname{diag}\big([\cdots, r_k w(t_k - t), \cdots]\big)$$

The MATLAB function `loess` implements these steps:

```
[xhat,C] = loess(tobs,yobs,t,alpha,d,Nit);      % Loess smoothing
```

where t are the final fitting times and `xhat` and `C` have the same meaning as in `locpol`. This function is similar in spirit to the robust local polynomial filtering function `rlpfilt` that was discussed in Sec. 4.5.

Example 5.6.1: Fig. 5.6.1 shows the same example as that of Fig. 4.5.3, with nearest-neighbor bandwidth parameter $\alpha = 0.4$ and an order-2 polynomial. The graphs show the results of $N_{\text{it}} = 0, 2, 4, 6$ iterations—the first one corresponding to ordinary fitting with no robustness iterations. The MATLAB code for the top two graphs was:

```
t = (0:50); x0 = (1 - cos(2*pi*t/50))/2;       % desired signal
seed=2005; randn('state',seed);
y = x0 + 0.1 * randn(size(x0));                % noisy signal

m = [-1 0 1 3];                                % outlier indices
n0=25; y(n0+m+1) = 0;                          % outlier values
n1=10; y(n1+m+1) = 1;
```

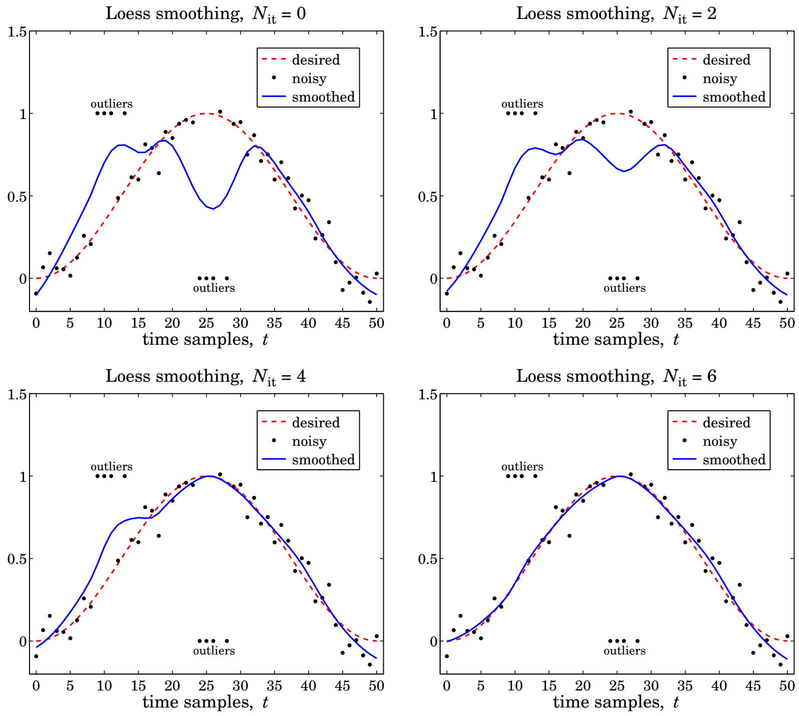


Fig. 5.6.1 Loess smoothing with $d = 2$, $\alpha = 0.4$, and different iterations.

```
alpha=0.4; d=2;                                % bandwidth parameter and polynomial order

Nit=0; x = loess(t,y,t,alpha,d,Nit);           % loess fit at t
figure; plot(t,x0,'--', t,y,'.', t,x,'-');     % left graph

Nit=2; x = loess(t,y,t,alpha,d,Nit);           % loess fit at t
figure; plot(t,x0,'--', t,y,'.', t,x,'-');     % right graph
```

The loess fit was performed at all t. We observe how successive iterations gradually diminish the distorting influence of the outliers. □

5.7 Problems

5.1 Prove the matrix inversion lemma identity (5.2.8). Using this identity, show that

$$H_{ii} = \frac{H_{ii}^-}{1 + H_{ii}^-}\,, \quad \text{where } H_{ii}^- = w_0 \mathbf{u}_0^T F_i^- \mathbf{u}_0\,, \quad F_i^- = (S_i^T W_i S_i)_-$$

then, argue that $0 \le H_{ii} \le 1$.

6

Exponential Smoothing

6.1 Mean Tracking

l

The exponential smoother, also known as an exponentially-weighted moving average (EWMA) or more simply an exponential moving average (EMA) filter is a simple, effective, recursive smoothing filter that can be applied to real-time data. By contrast, the local polynomial modeling approach is typically applied off-line to a block a signal samples that have already been collected.

The exponential smoother is used routinely to track stock market data and in forecasting applications such as inventory control where, despite its simplicity, it is highly competitive with other more sophisticated forecasting methods [232–279].

We have already encountered it in Sec. 2.3 and compared it to the plain FIR averager. Here, we view it as a special case of a weighted local polynomial smoothing problem using a causal window and exponential weights, and discuss some of its generalizations. Both the exponential smoother and the FIR averager are applied to data that are assumed to have the typical form:

$$y_n = a_n + v_n \tag{6.1.1}$$

where a_n is a low-frequency trend component, representing an average or estimate of the *local level* of the signal, and v_n a random, zero-mean, broadband component, such as white noise. If a_n is a deterministic signal, then by taking expectations of both sides we see that a_n represents the mean value of y_n, that is, $a_n = E[y_n]$. If y_n is stationary, then a_n is a constant, independent of n.

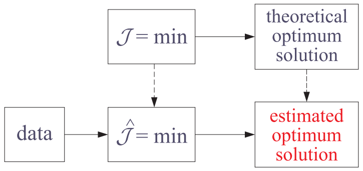
The output of either the FIR or the exponential smoothing filter tracks the signal a_n. To see how such filters arise in the context of estimating the mean level of a signal, consider first the stationary case. The mean $m = E[y_n]$ minimizes the following variance performance index (e.g., see Example 1.3.5):

$$\mathcal{J} = E\big[(y_n - a)^2\big] = \min \quad \Rightarrow \quad a_{\mathrm{opt}} = m = E[y_n] \tag{6.1.2}$$

with minimized value $\mathcal{J}_{\min} = \sigma_y^2$. This result is obtained by setting the gradient with respect to a to zero:

$$\frac{\partial \mathcal{J}}{\partial a} = -2E[y_n - a] = 0 \tag{6.1.3}$$

In general, given a theoretical performance index $\mathcal{J}$, one must replace it in practice by an experimental one, say $\hat{\mathcal{J}}$, expressible in terms of the actual available data. The minimization of $\hat{\mathcal{J}}$ provides then estimates of the parameters or signals to be estimated.



Depending on the index $\hat{\mathcal{J}}$, the estimates may be calculated in a block processing manner using an entire block of data, or, on a sample-by-sample basis with the estimate being updated in real time in response to each new data sample. All adaptive filtering algorithms follow the latter approach.

We illustrate these ideas with the help of the simple performance index (6.1.2). We will apply this approach extensively in Chap. 16. Four possible practical definitions for $\hat{\mathcal{J}}$ that imitate (6.1.2) are:

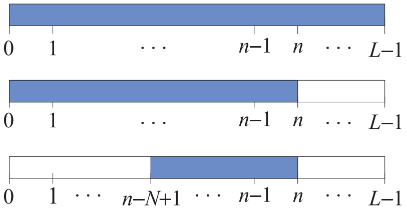
$$\hat{\mathcal{J}} = \sum_{n=0}^{L-1}(y_n - \hat{a})^2 = \min \tag{6.1.4a}$$

$$\hat{\mathcal{J}} = \sum_{k=0}^{n}(y_k - \hat{a})^2 = \min \tag{6.1.4b}$$

$$\hat{\mathcal{J}} = \sum_{k=n-N+1}^{n}(y_k - \hat{a})^2 = \min \tag{6.1.4c}$$

$$\hat{\mathcal{J}} = \sum_{k=0}^{n}\lambda^{n-k}(y_k - \hat{a})^2 = \min \tag{6.1.4d}$$

The first assumes a length-L block of data $[y_0, y_1, \dots, y_{L-1}]$. The last three are suitable for real-time implementations, where n denotes the current time. The second and fourth use the first $n+1$ data $[y_0, y_1, \dots, y_n]$, while the third uses a length-N sliding window $[y_{n-N+1}, \dots, y_{n-1}, y_n]$. The third choice leads to the FIR averager, also known as the *simple moving average* (SMA), and the fourth, to the exponential smoother, or, *exponential moving average* (EMA), where we require that the exponential "forgetting factor" λ be in the range $0 < \lambda < 1$. These time ranges are depicted below.



In order for the $\hat{\mathcal{J}}$s to be unbiased estimates of $\mathcal{J}$, the above expressions should have been divided by the sum of their respective weights, namely, the quantities L, $(n+1)$, N, and $(1+\lambda+\cdots+\lambda^n)$, respectively. However, such factors do not affect the minimization solutions, which are easily found to be:

$$\hat{a} = \frac{y_0 + y_1 + \cdots + y_{L-1}}{L} \tag{6.1.5a}$$

$$\hat{a}_n = \frac{y_0 + y_1 + \cdots + y_n}{n+1} \tag{6.1.5b}$$

$$\hat{a}_n = \frac{y_n + y_{n-1} + \cdots + y_{n-N+1}}{N} \tag{6.1.5c}$$

$$\hat{a}_n = \frac{y_n + \lambda y_{n-1} + \lambda^2 y_{n-2} + \cdots \lambda^n y_0}{1 + \lambda + \lambda^2 + \cdots + \lambda^n} \tag{6.1.5d}$$

We have tacked on a subscript n to the last three to emphasize their dependence of their performance index on the current time instant n. Eqs. (6.1.4c) and (6.1.5c) tentatively assume that $n \geq N-1$; for $0 \leq n < N-1$, one should use the running average (6.1.4b) and (6.1.5b). Initialization issues are discussed further in Sections 6.6 and 6.19.

All four estimates are *unbiased* estimators of the true mean m. Their quality as estimators can be judged by their variances, which are (assuming that $y_n - m$ are mutually independent):

$$\sigma_{\hat{a}}^2 = E\big[(\hat{a} - m)^2\big] = \frac{\sigma_y^2}{L} \tag{6.1.6a}$$

$$\sigma_{\hat{a}_n}^2 = E\big[(\hat{a}_n - m)^2\big] = \frac{\sigma_y^2}{n+1} \tag{6.1.6b}$$

$$\sigma_{\hat{a}_n}^2 = E\big[(\hat{a}_n - m)^2\big] = \frac{\sigma_y^2}{N} \tag{6.1.6c}$$

$$\sigma_{\hat{a}_n}^2 = E\big[(\hat{a}_n - m)^2\big] = \sigma_y^2\,\frac{1-\lambda}{1+\lambda}\cdot\frac{1+\lambda^{n+1}}{1-\lambda^{n+1}} \tag{6.1.6d}$$

The first two, corresponding to ordinary sample averaging, are asymptotically consistent estimators having variances that tend to zero as $L \to \infty$ or $n \to \infty$. The last two are not consistent. However, their variances can be made as small as desired by proper choice of the parameters N or λ.

The exponential smoothing filter may also be derived from a different point of view. The estimates (6.1.5) are the *exact* least-squares solutions of the indices (6.1.4). An alternative to using the exact solutions is to derive an LMS (least-mean-square) type of adaptive algorithm which minimizes the performance index iteratively using a steepest-descent algorithm that replaces the theoretical gradient (6.1.3) by an "instantaneous" one in which the expectation instruction is ignored:

$$\frac{\partial \mathcal{J}}{\partial a} = -2E[y_n - a] \quad\longrightarrow\quad \widehat{\frac{\partial \mathcal{J}}{\partial a}} = -2\big[y_n - \hat{a}_{n-1}\big] \tag{6.1.7}$$

The LMS algorithm then updates the previous estimate by adding a correction in the direction of the negative gradient using a small positive adaptation parameter μ:

$$\Delta a = -\mu\,\widehat{\frac{\partial \mathcal{J}}{\partial a}}\,, \qquad \hat{a}_n = \hat{a}_{n-1} + \Delta a \tag{6.1.8}$$

The resulting difference equation is identical to that of the steady-state exponential smoother (see Eq. (6.1.11) below),

$$\hat{a}_n = \hat{a}_{n-1} + 2\mu(y_n - \hat{a}_{n-1})$$

In adaptive filtering applications, the use of the exponentially discounted type of performance index (6.1.4d) leads to the so-called recursive least-squares (RLS) adaptive filters, which are in general different from the LMS adaptive filters. They happened to coincide in this particular example because of the simplicity of the problem.

The sample mean estimators (6.1.5a) and (6.1.5b) are geared to stationary data, whereas (6.1.5c) and (6.1.5d) can track nonstationary changes in the statistics of y_n. If y_n is nonstationary, then its mean $a_n = E[y_n]$ would be varying with n and a good estimate should be able to track it well and efficiently. To see the problems that arise in using the sample mean estimators in the nonstationary case, let us cast Eqs. (6.1.5b) and (6.1.5d) in recursive form. Both can be written as follows:

$$\hat{a}_n = (1-\alpha_n)\hat{a}_{n-1} + \alpha_n y_n = \hat{a}_{n-1} + \alpha_n(y_n - \hat{a}_{n-1}) \tag{6.1.9}$$

where the gain parameter α_n is given by

$$\alpha_n = \frac{1}{n+1}, \quad \alpha_n = \frac{1}{1+\lambda+\cdots+\lambda^n} = \frac{1-\lambda}{1-\lambda^{n+1}} \tag{6.1.10}$$

for (6.1.5b) and (6.1.5d), respectively. The last side of Eq. (6.1.9) is written in a so-called "predictor/corrector" Kalman filter form, where the first term $\hat{a}_{n-1}$ is a tentative prediction of $\hat{a}_n$ and the second term is the correction obtained by multiplying the "prediction error" $(y_n - \hat{a}_{n-1})$ by a positive gain factor α_n. This term always corrects in the right direction, that is, if $\hat{a}_{n-1}$ overestimates/underestimates y_n then the error tends to be negative/positive reducing/increasing the value of $\hat{a}_{n-1}$.

There is a dramatic difference between the two estimators. For the sample mean, the gain $\alpha_n = 1/(n+1)$ tends to zero rapidly with increasing n. For stationary data, the estimate $\hat{a}_n$ will converge quickly to the true mean. Once n is fairly large, the correction term becomes essentially unimportant because the gain is so small. If after converging to the true mean the statistics of y_n were to suddenly change with a new value of the mean, the sample-mean estimator $\hat{a}_n$ would have a very hard time responding to such a change and converging to the new value because the new changes are communicated only through the already very small correction term.

On the other hand, for the exponential smoother case (6.1.5d), the gain tends to a constant for large n, that is, $\alpha_n \to \alpha = 1-\lambda$. Therefore, the correction term remains finite and can communicate the changes in the statistics. The price one pays for that is that the estimator is not consistent. Asymptotically, the estimator (6.1.5d) becomes the ordinary exponential smoothing filter described by the difference equation,

$$\hat{a}_n = \lambda\hat{a}_{n-1} + \alpha y_n = \hat{a}_{n-1} + \alpha(y_n - \hat{a}_{n-1}) \tag{6.1.11}$$

Its transfer function and asymptotic variance are:

$$H(z) = \frac{\alpha}{1-\lambda z^{-1}}, \quad \sigma^2_{\hat{a}_n} = E[(\hat{a}_n - m)^2] = \sigma_y^2\frac{1-\lambda}{1+\lambda} \tag{6.1.12}$$

The quantity $\sigma^2_{\hat{a}_n}/\sigma^2_y$ is the NRR of this filter. The differences in the behavior of the sample-mean and exponential smoother can be understood by inspecting the corresponding performance indices, which may be written in an expanded form:

$$\begin{aligned}\mathcal{J} &= (y_n-\hat{a})^2 + (y_{n-1}-\hat{a})^2 + (y_{n-2}-\hat{a})^2 + \cdots + (y_0-\hat{a})^2 \\ \mathcal{J} &= (y_n-\hat{a})^2 + \lambda(y_{n-1}-\hat{a})^2 + \lambda^2(y_{n-2}-\hat{a})^2 + \cdots + \lambda^n(y_0-\hat{a})^2\end{aligned} \tag{6.1.13}$$

The first index weighs all terms equally, as it should for stationary data. The second index emphasizes the terms arising from the most recent observation y_n and exponentially forgets, or discounts, the earlier observations and thus can respond more quickly to new changes. Even though the second index appears to have an ever increasing number of terms, in reality, the effective number of terms that are significant is finite and can be estimated by the formula:

$$\bar{n} = \frac{\displaystyle\sum_{n=0}^{\infty} n\lambda^n}{\displaystyle\sum_{n=0}^{\infty}\lambda^n} = \frac{\lambda}{1-\lambda} \tag{6.1.14}$$

This expression is only a guideline and other possibilities exist. For example, one can define $\bar{n}$ to be the effective time constant of the filter:

$$\lambda^{\bar{n}} = \epsilon \quad\Rightarrow\quad \bar{n} = \frac{\ln\epsilon}{\ln\lambda} \simeq \frac{\ln(\epsilon^{-1})}{1-\lambda}, \quad \text{for } \lambda \lesssim 1 \tag{6.1.15}$$

where ϵ is a small user-specified parameter such as $\epsilon = 0.01$. The sliding window estimator (6.1.5c) is recognized as a length-N FIR averaging filter of the type we considered in Sec. 2.4. It also can track a nonstationary signal at the expense of not being a consistent estimator. Requiring that it achieve the same variance as the exponential smoother gives the conditions:

$$\boxed{\frac{1}{N}\sigma_y^2 = \frac{1-\lambda}{1+\lambda}\sigma_y^2} \quad\Rightarrow\quad \lambda = \frac{N-1}{N+1} \quad\Rightarrow\quad \alpha = 1-\lambda = \frac{2}{N+1} \tag{6.1.16}$$

Such conditions are routinely used to set the parameters of FIR and exponential smoothing filters in inventory control applications and in tracking stock market data. A similar weighted average as in Eq. (6.1.14) can be defined for any filter by:

$$\boxed{\bar{n} = \frac{\displaystyle\sum_n nh_n}{\displaystyle\sum_n h_n}} \quad \text{(effective filter lag)} \tag{6.1.17}$$

where h_n is the filter's impulse response. Eq. (6.1.17) may also be expressed in terms of the filter's transfer function $H(z) = \sum_n h_n z^{-n}$ and its derivative $H'(z) = dH(z)/dz$ evaluated at DC, that is, at $z = 1$:

$$\bar{n} = -\left.\frac{H'(z)}{H(z)}\right|_{z=1} \quad \text{(effective filter lag)} \tag{6.1.18}$$

Alternatively, $\bar{n}$ is recognized as the filter's *group delay* at DC, that is, given the frequency response $H(\omega)=\sum_n h_n e^{-j\omega n} = |H(\omega)|e^{j\arg H(\omega)}$, we have (Problem 6.1):

$$\bar{n} = -\frac{d}{d\omega}\arg H(\omega)\bigg|_{\omega=0} \qquad \text{(group delay at DC)} \tag{6.1.19}$$

The exponential smoother is a special case of (6.1.17) with $h_n = \alpha\lambda^n u(n)$, where $u(n)$ is the unit-step function. We may apply this definition also to the FIR averager filter that has $h_n = 1/N$, for $n = 0, 1, \dots, N-1$,

$$\bar{n} = \frac{1}{N}\sum_{n=0}^{N-1} n = \frac{N-1}{2}$$

The FIR averager can be mapped into an "equivalent" exponential smoother by equating the $\bar{n}$ lags of the two filters, that is,

$$\boxed{\bar{n} = \frac{N-1}{2} = \frac{\lambda}{1-\lambda}} \tag{6.1.20}$$

This condition is exactly equivalent to condition (6.1.16) arising from matching the NRRs of the two filters. The two equations,

$$E\big[(\hat{a}_n - m)^2\big] = \frac{1-\lambda}{1+\lambda}\sigma_y^2 = \frac{1}{N}\sigma_y^2\,, \qquad \bar{n} = \frac{\lambda}{1-\lambda} = \frac{N-1}{2} \tag{6.1.21}$$

capture the main tradeoff between variance and speed in using an exponential smoother or an equivalent FIR averager, that is, the closer λ is to unity or the larger the N, the smaller the variance and the better the estimate, but the longer the transients and the slower the speed of response.

We summarize the difference equations for the exact exponential smoother (6.1.5d) and the steady-state one (6.1.11),

$$\begin{aligned} \hat{a}_n &= \frac{\lambda - \lambda^{n+1}}{1-\lambda^{n+1}}\,\hat{a}_{n-1} + \frac{\alpha}{1-\lambda^{n+1}}\,y_n = \hat{a}_{n-1} + \frac{\alpha}{1-\lambda^{n+1}}\,(y_n - \hat{a}_{n-1}) \\ \hat{a}_n &= \lambda\hat{a}_{n-1} + \alpha\,y_n = \hat{a}_{n-1} + \alpha(y_n - \hat{a}_{n-1}) \end{aligned} \tag{6.1.22}$$

Clearly, the second is obtained in the large-n limit of the first, but in practice the steady one is often used from the start at $n = 0$ because of its simplicity.

To start the recursions at $n = 0$, one needs to specify the initial value $\hat{a}_{-1}$. For the exact smoother, $\hat{a}_{-1}$ can have an arbitrary value because its coefficient vanishes at $n = 0$. This gives for the first smoothed value $\hat{a}_0 = 0\cdot\hat{a}_{-1} + 1\cdot y_0 = y_0$. For the steady smoother it would make sense to also require that $\hat{a}_0 = y_0$, which would imply that $\hat{a}_{-1} = y_0$ because then

$$\hat{a}_0 = \lambda\hat{a}_{-1} + \alpha y_0 = \lambda y_0 + \alpha y_0 = y_0$$

There are other reasonable ways of choosing $\hat{a}_{-1}$, for example one could take it to be the average of a few initial values of y_n. The convolutional solution of the steady smoother with arbitrary nonzero initial conditions is obtained by convolving the filter's impulse response $\alpha\lambda^n u(n)$ with the causal input y_n plus adding a transient term arising from the initial value:

$$\hat{a}_n = \alpha\sum_{k=0}^{n}\lambda^{n-k}y_k + \lambda^{n+1}\hat{a}_{-1} \tag{6.1.23}$$

The influence of the initial value disappears exponentially.

Example 6.1.1: Fig. 6.1.1 illustrates the ability of the sample mean and the exponential smoother to track a sudden level change.

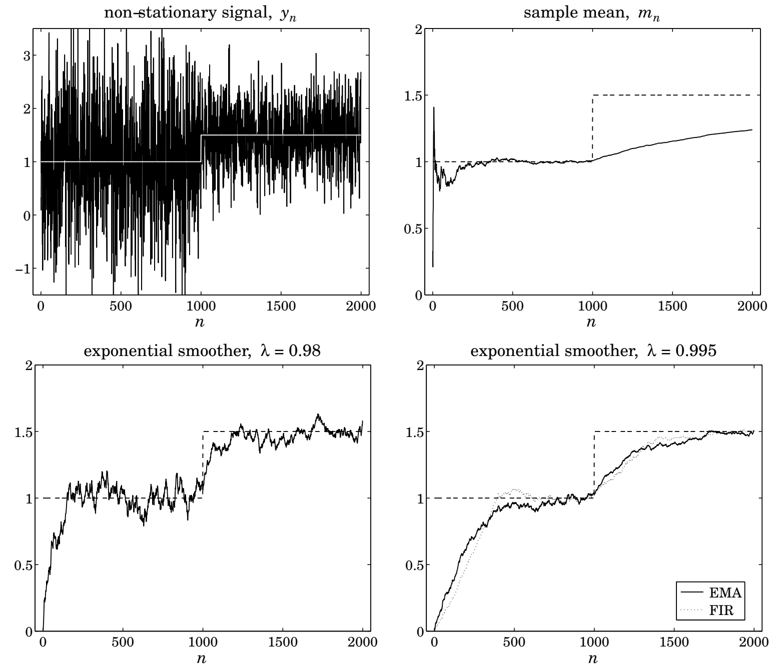


Fig. 6.1.1 Mean tracking with sample mean, exponential smoother, and FIR averager.

The first 1000 samples of the signal y_n depicted on the upper-left graph are independent gaussian samples of mean and variance $m_1 = 1$, $\sigma_1 = 1$. The last 1000 samples are gaussian samples with $m_2 = 1.5$ and $\sigma_2 = 0.5$.

The upper-right graph shows the sample mean computed recursively using (6.1.9) with $\alpha_n = 1/(n+1)$ and initialized at $\hat{a}_{-1} = 0$ (although the initial value does not matter since $\alpha_0 = 1$). We observe that the sample mean converges very fast to the first value of $m_1 = 1$, with its fluctuations becoming smaller and smaller because of its decreasing variance (6.1.6b). But it is extremely slow responding to the sudden change in the mean.

The bottom two graphs show the steady-state exponential smoother initialized at $\hat{a}_{-1} = 0$ with the two values of the forgetting factor $\lambda = 0.98$ and $\lambda = 0.995$. For the smaller λ the convergence is quick both at the beginning and after the change, but the fluctuations quantified by (6.1.21) remain finite and do not improve even after convergence. For the larger λ, the fluctuations are smaller, but the learning time constant is longer. In the bottom-right graph, we have also added the equivalent FIR averager with N related to λ by (6.1.16), which gives $N = 399$. Its learning speed and fluctuations are comparable to those of the exponential smoother. □

Example 6.1.2: Fig. 6.1.2 shows the daily Dow-Jones Industrial Average (DJIA) from Oct. 1, 2007 to Dec. 31, 2009. In the left graph an exponential smoothing filter is used with $\lambda = 0.9$. In the right graph, an FIR averager with an equivalent length of $N = (1+\lambda)/(1-\lambda) = 19$ is used. The data were obtained from `http://finance.yahoo.com`.

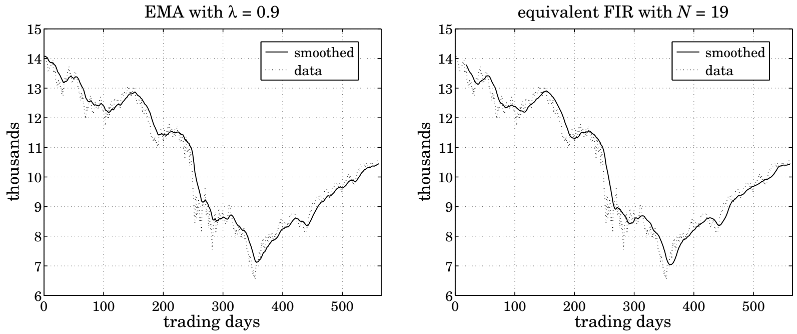


Fig. 6.1.2 Dow-Jones industrial average from 10-Oct-2007 to 31-Dec-2009.

The following code fragment generates the two graphs:

```
Y = loadfile('dow-oct07-dec09.dat');             % data file in OSP toolbox
y = Y(:,4)/1000;                                 % extract closing prices
n = (0:length(y)-1);

la = 0.9; al = 1-la;
s0 = la*y(1);                                    % s0 is the initial state
m = filter(al, [1,-la], y, s0);                  % filter with initial state
% m = stema(y,0,la, y(1));                       % equivalent calculation

figure; plot(n,m,'-', n,y,':');

N = round((1+la)/(1-la));
h = ones(N,1)/N;                                 % FIR averager
x = filter(h,1,y);

figure; plot(n(N:end),x(N:end),'-', n,y,':');    % discard first N−1 outputs
```

The initial value was set such that to get $\hat{a}_0 = y_0$ for the EMA. The built-in function `filter` allows one to specify the initial state. Because `filter` uses the transposed realization, in order to have $\hat{a}_0 = y_0$, the initial state must be chosen as $s_{\text{in}} = \lambda y_0$. This follows from the sample processing algorithm of the transposed realization for the EMA filter (6.1.12), which reads as follows where s is the state:

```
for each input sample $y$ do:
    $\hat{a} = s + \alpha y$
    $s = \lambda\hat{a}$
```

or

$$\boxed{\begin{aligned}\hat{a}_n &= s_n + \alpha y_n\\ s_{n+1} &= \lambda\hat{a}_n\end{aligned}}$$

Thus, in order for the first pass to give $\hat{a}_0 = y_0$, the initial state must be such that $s_0 = \hat{a}_0 - \alpha y_0 = \lambda y_0$. The FIR averager was run with zero initial conditions and therefore, the first $N-1$ outputs were discarded as transients. After $n \ge N$, the EMA and the FIR outputs are comparable since they have the same $\bar{n}$. □

Example 6.1.3: It is evident by inspecting the graphs of the previous example that both the EMA and the FIR filter outputs are lagging behind the data signal. To see this lag more clearly, consider a noiseless signal consisting of three straight-line segments defined by,

$$s_n = \begin{cases} 20 + 0.8n, & 0 \le n < 75 \\ 80 - 0.3(n-75), & 75 \le n < 225 \\ 35 + 0.4(n-225), & 225 \le n \le 300 \end{cases}$$

Fig. 6.1.3 shows the corresponding output from an EMA with $\lambda = 0.9$ and an equivalent FIR averager with $N = 19$ as in the previous example. The dashed line is the signal s_n and the solid lines, the corresponding filter outputs.

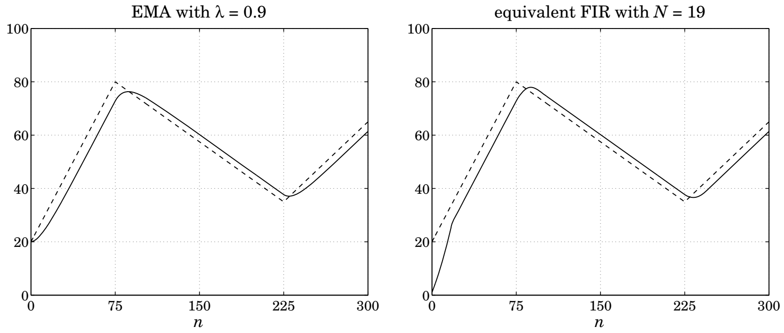


Fig. 6.1.3 Lag introduced by EMA and FIR averager filters.

The EMA was run with initial value $\hat{a}_{-1} = s_0 = 20$. The FIR filter was run with zero initial conditions, and therefore, its first $N-1$ outputs are transients. The amount of delay introduced by the filters is exactly equal to the quantity $\bar{n}$ of Eq. (6.1.20). □

The delay $\bar{n}$ is a consequence of the causality of the filters. Symmetric non-causal filters, such as the LPSM or LPRS filters, do not introduce a delay, that is, $\bar{n} = 0$.

To see how such a delay arises, consider an arbitrary causal filter h_n and a causal input that is a linear function of time, $x_n = a + bn$, for $n \ge 0$. The corresponding convolutional output will be:

$$y_n = \sum_{k=0}^{n} h_k x_{n-k} = \sum_{k=0}^{n} h_k\big[a + b(n-k)\big] = (a+bn)\sum_{k=0}^{n} h_k - b\sum_{k=0}^{n} k h_k$$

For large n, we may replace the upper limit of the summations by $k=\infty$,

$$y_n = (a+bn)\sum_{k=0}^{\infty}h_k - b\sum_{k=0}^{\infty}kh_k = (a+bn)\sum_{k=0}^{\infty}h_k - b\bar{n}\sum_{k=0}^{\infty}h_k = \big[a+b(n-\bar{n})\big]\sum_{k=0}^{\infty}h_k$$

where we used the definition (6.1.17) for $\bar{n}$. For filters that have unity gain at DC, the sum of the filter coefficients is unity, and we obtain,

$$y_n = a+b(n-\bar{n}) = x_{n-\bar{n}} \tag{6.1.24}$$

Such delays are of concern in a number of applications, such as the real-time monitoring of financial data. For FIR filters, the problem of designing noise reducing filters with a prescribed amount of delay $\bar{n}$ has already been discussed in Sec. 3.8. However, we discuss it a bit further in Sec. 6.10 and 6.15 emphasizing its use in stock market trading. The delay $\bar{n}$ can also be controlled by the use of higher-order exponential smoothing discussed in Sec. 6.5.

6.2 Forecasting and State-Space Models

We make a few remarks on the use of the first-order exponential smoother as a forecasting tool. As we already mentioned, the quantity $\hat{a}_{n-1}$ can be viewed as a prediction of y_n based on the past observations $\{y_0, y_1, \dots, y_{n-1}\}$. To emphasize this interpretation, let us denote it by $\hat{y}_{n/n-1} = \hat{a}_{n-1}$, and the corresponding prediction or forecast error by $e_{n/n-1} = y_n - \hat{y}_{n/n-1}$. Then, the steady exponential smoother can be written as,

$$\hat{y}_{n+1/n} = \hat{y}_{n/n-1} + \alpha e_{n/n-1} = \hat{y}_{n/n-1} + \alpha(y_n - \hat{y}_{n/n-1}) \tag{6.2.1}$$

As discussed in Chapters 1 and 12, if the prediction is to be optimal, then the prediction error $e_{n/n-1}$ must be a white noise signal, called the innovations of the sequence y_n and denoted by $\varepsilon_n = e_{n/n-1}$. It represents that part of y_n that cannot be predicted from its past. This interpretation implies a certain innovations signal model for y_n. We may derive it by working with z-transforms. In the z-domain, Eq. (6.2.1) reads,

$$z\hat{Y}(z) = \hat{Y}(z) + \alpha E(z) = \hat{Y}(z) + \alpha\big(Y(z) - \hat{Y}(z)\big) = \lambda\hat{Y}(z) + \alpha Y(z) \tag{6.2.2}$$

Therefore, the transfer functions from $Y(z)$ to $\hat{Y}(z)$ and from $Y(z)$ to $E(z)$ are,

$$\hat{Y}(z) = \left(\frac{\alpha z^{-1}}{1-\lambda z^{-1}}\right)Y(z)\,, \quad E(z) = \left(\frac{1-z^{-1}}{1-\lambda z^{-1}}\right)Y(z) \tag{6.2.3}$$

In the time domain, using the notation $\nabla y_n = y_n - y_{n-1}$, we may write the latter as

$$\nabla y_n = \varepsilon_n - \lambda\varepsilon_{n-1} \tag{6.2.4}$$

Thus, y_n is an integrated ARMA process, ARIMA(0,1,1), or more simply an integrated MA process, IMA(1,1). In other words, if y_n is such a process, then the exponential smoother forecast $\hat{y}_{n/n-1}$ is optimal in the mean-square sense [242].

The innovations representation model can also be cast in an ordinary Wiener and Kalman filter form of the type discussed in Chap. 11. The state and measurement equations for such a model are:

$$\begin{aligned} x_{n+1} &= x_n + w_n \\ y_n &= x_n + v_n \end{aligned} \tag{6.2.5}$$

where w_n, v_n are zero-mean white-noise signals that are mutually uncorrelated. This model is referred to as a “constant level” state-space model, and represents a random-walk observed in noise. The optimal prediction estimate $\hat{x}_{n/n-1}$ of the state x_n is equivalent to $\hat{a}_{n-1}$. The equivalence between EMA and this model results in the following relationship between the parameters α and $q = \sigma_w^2/\sigma_v^2$:

$$q = \frac{\alpha^2}{1-\alpha} \quad\Rightarrow\quad \alpha = \frac{\sqrt{q^2+4q}-q}{2} \tag{6.2.6}$$

We defer discussion of such state-space models until chapters 11 and 13.

6.3 Higher-Order Polynomial Smoothing Filters

We recall that in fitting a local polynomial of order d to a local block of data $\{y_{n-M}, \dots, y_n, \dots, y_{n+M}\}$, the performance index was

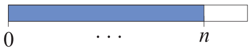
$$\mathcal{J} = \sum_{k=-M}^{M}\big[y_{n+k} - p(k)\big]^2 = \sum_{k=-M}^{M}\big[y_{n+k} - \mathbf{u}_k^T\mathbf{c}\big]^2 = \min$$

where $p(k)$ is a dth degree polynomial, representing the estimate $\hat{y}_{n+k} = p(k)$,

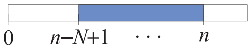
$$p(k) = \mathbf{u}_k^T\mathbf{c} = [1, k, \dots, k^d]\begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_d \end{bmatrix} = \sum_{i=0}^{d} c_i k^i$$

and we defined the monomial basis vector $\mathbf{u}_k = [1, k, k^2, \dots, k^d]^T$. The higher-order exponential smoother is obtained by restricting the data range to $\{y_0, y_1, \dots, y_n\}$ and using exponential weights, and similarly, the corresponding FIR version will be restricted to $\{y_{n-N+1}, \dots, y_{n-1}, y_n\}$. The resulting performance indices are then,

$$\mathcal{J}_n = \sum_{k=-n}^{0}\lambda^{-k}\big[y_{n+k} - \mathbf{u}_k^T\mathbf{c}\big]^2 = \min$$



$$\mathcal{J}_n = \sum_{k=-N+1}^{0}\big[y_{n+k} - \mathbf{u}_k^T\mathbf{c}\big]^2 = \min$$



or, replacing the summation index k by $-k$, the performance indices read,

$$\begin{aligned} &\text{(EMA)}\quad \mathcal{J}_n = \sum_{k=0}^{n}\lambda^k\big[y_{n-k} - \mathbf{u}_{-k}^T\mathbf{c}\big]^2 = \min \\ &\text{(FIR)}\quad \mathcal{J}_n = \sum_{k=0}^{N-1}\big[y_{n-k} - \mathbf{u}_{-k}^T\mathbf{c}\big]^2 = \min \end{aligned} \tag{6.3.1}$$

In both cases, we may interpret the quantities $p(\pm\tau)=\mathbf{u}_{\pm\tau}^T\mathbf{c}$ as the estimates $\hat{y}_{n\pm\tau}$. We will denote them by $\hat{y}_{n\pm\tau/n}$ to emphasize their causal dependence only on data up to the current time n. In particular, the quantity $c_0=\mathbf{u}_0^T\mathbf{c}=p(0)$ represents the estimate $\hat{y}_n$, or $\hat{y}_{n/n}$, that is, an estimate of the local level of the signal. Similarly, $c_1=\dot{p}(0)=\dot{\mathbf{u}}_\tau^T\mathbf{c}|_{\tau=0}$ represents the local slope, and $2c_2=\ddot{p}(0)$, the local acceleration. Eqs. (6.1.4d) and (6.1.4c) are special cases of (6.3.1) corresponding to $d=0$.

Both indices in Eq. (6.3.1) can be written in the following compact vectorial form, whose solution we have already obtained in previous chapters:

$$\mathcal{J}=(\mathbf{y}-S\mathbf{c})^TW(\mathbf{y}-S\mathbf{c})=\min \quad\Rightarrow\quad \mathbf{c}=(S^TWS)^{-1}S^TW\mathbf{y} \tag{6.3.2}$$

where the data vector $\mathbf{y}$ is defined as follows in the EMA and FIR cases,

$$\mathbf{y}(n)=\begin{bmatrix} y_n \\ y_{n-1} \\ \vdots \\ y_0 \end{bmatrix}, \quad \mathbf{y}(n)=\begin{bmatrix} y_n \\ y_{n-1} \\ \vdots \\ y_{n-N+1} \end{bmatrix} \tag{6.3.3}$$

with the polynomial basis matrices S,

$$S_n=\big[\mathbf{u}_0,\mathbf{u}_{-1},\dots,\mathbf{u}_{-n}\big]^T, \quad S_N=\big[\mathbf{u}_0,\mathbf{u}_{-1},\dots,\mathbf{u}_{-N+1}\big]^T=\begin{bmatrix} u_0^T \\ \mathbf{u}_{-1}^T \\ \vdots \\ \mathbf{u}_{-k}^T \\ \vdots \\ \mathbf{u}_{-N+1}^T \end{bmatrix} \tag{6.3.4}$$

with, $\mathbf{u}_{-k}^T=\big[1,(-k),(-k)^2,\dots,(-k)^d\big]$, and weight matrices W in the two cases,

$$W_n=\mathrm{diag}\big([1,\lambda,\dots,\lambda^n]\big), \quad\text{or,}\quad W=I_N \tag{6.3.5}$$

The predicted estimates can be written in the filtering form:

$$\boxed{\hat{y}_{n+\tau/n}=\mathbf{u}_\tau^T\mathbf{c}(n)=\mathbf{h}_\tau^T(n)\mathbf{y}(n)} \tag{6.3.6}$$

where in the exponential smoothing case,

$$\boxed{\begin{aligned} \mathbf{c}(n)&=(S_n^TW_nS_n)^{-1}S_n^TW_n\mathbf{y}(n) \\ \mathbf{h}_\tau(n)&=W_nS_n(S_n^TW_nS_n)^{-1}\mathbf{u}_\tau \end{aligned}} \quad \text{(EMA)} \tag{6.3.7}$$

We will see in Eq. (6.5.19) and more explicitly in (6.6.5) that $\mathbf{c}(n)$ can be expressed recursively in the time n. Similarly, for the FIR case, we find:

$$\boxed{\begin{aligned} \mathbf{c}(n)&=(S_N^TS_N)^{-1}S_N^T\mathbf{y}(n) \\ \mathbf{h}_\tau&=S_N(S_N^TS_N)^{-1}\mathbf{u}_\tau \end{aligned}} \quad \text{(FIR)} \tag{6.3.8}$$

We note also that the the solution for $\mathbf{c}$ in Eq. (6.3.2) can be viewed as the least-squares solution of the over-determined linear system, $W^{1/2}S\mathbf{c}=W^{1/2}\mathbf{y}$, which is particularly convenient for the numerical solution using MATLAB's backslash operation,

$$\mathbf{c}=(W^{1/2}S)\setminus(W^{1/2}\mathbf{y}) \tag{6.3.9}$$

In fact, this corresponds to an alternative point of view to filtering and is followed in the so-called “linear regression” indicators in financial market trading, as we discuss in Sec. 6.18, where the issue of the initial transients, that is, the evaluation of $\mathbf{c}(n)$ for $0\le n\le N-1$ in the FIR case, is also discussed.

In the EMA case, the basis matrices S_n are full rank for $n\ge d$. For $0\le n<d$, we may restrict the polynomial order d to to $d_n=n$ and thus obtain the first d_n coefficients of the vector $\mathbf{c}(n)$, and set the remaining coefficients to zero. For the commonly used case of $d=1$, this procedure amounts to setting $\mathbf{c}(0)=[y_0,\,0]^T$. Similarly, in the FIR case, we must have $N\ge d+1$ to guarantee the full rank of S_N.

6.4 Linear Trend FIR Filters

The exact solutions of the FIR case have already been found in Sec. 3.8. The $d=1$ and $d=2$ closed-form solutions were given in Eqs. (3.8.10) and (3.8.11). The same expressions are valid for both even and odd N. For example, replacing $M=(N-1)/2$ in (3.8.10), we may express the solution for the $d=1$ case as follows,

$$\boxed{h_\tau(k)=\frac{2(N-1)(2N-1-3k)+6(N-1-2k)\tau}{N(N^2-1)}}, \quad k=0,1,\dots,N-1 \tag{6.4.1}$$

A direct derivation of (6.4.1) is as follows. From the definition (6.3.4), we find:

$$\begin{aligned} S_N^TS_N&=\sum_{k=0}^{N-1}\mathbf{u}_{-k}\mathbf{u}_{-k}^T=\sum_{k=0}^{N-1}\begin{bmatrix} 1 & -k \\ -k & k^2 \end{bmatrix} \\ &=\begin{bmatrix} N & -N(N-1)/2 \\ -N(N-1)/2 & N(N-1)(2N-1)/6 \end{bmatrix} \\ (S_N^TS_N)^{-1}&=\frac{2}{N(N^2-1)}\begin{bmatrix} (N-1)(2N-1) & 3(N-1) \\ 3(N-1) & 6 \end{bmatrix} \end{aligned} \tag{6.4.2}$$

then, from Eq. (6.3.8), because the kth row of S_N is $\mathbf{u}_{-k}^T$, we obtain the kth impulse response coefficient:

$$h_\tau(k)=\mathbf{u}_{-k}^T(S_N^TS_N)^{-1}\mathbf{u}_\tau=\frac{2}{N(N^2-1)}\big[1,\,-k\big]\begin{bmatrix} (N-1)(2N-1) & 3(N-1) \\ 3(N-1) & 6 \end{bmatrix}\begin{bmatrix} 1 \\ \tau \end{bmatrix}$$

which leads to Eq. (6.4.1). Thus, we obtain,

$$h_\tau(k)=h_a(k)+h_b(k)\tau, \quad k=0,1,\dots,N-1 \tag{6.4.3}$$

with

$$\boxed{h_a(k)=\frac{2(2N-1-3k)}{N(N+1)}\,,\quad h_b(k)=\frac{6(N-1-2k)}{N(N^2-1)}} \tag{6.4.4}$$

These are the FIR filters that generate estimates of the *local level* and *local slope* of the input signal. Indeed, setting $\mathbf{c}(n)=[a_n,b_n]^T$, where a_n,b_n represent the local level and local slope[†] at time n, we obtain from (6.3.8),

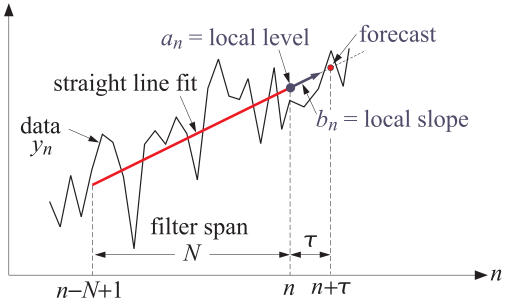
$$\begin{bmatrix} a_n \\ b_n \end{bmatrix} = (S_N^T S_N)^{-1}S_N^T\mathbf{y}(n) = (S_N^T S_N)^{-1}\sum_{k=0}^{N-1}\mathbf{u}_{-k}y_{n-k}$$

which is equivalent, component-wise, to the filtering equations:

$$\begin{aligned} a_n &= \sum_{k=0}^{N-1} h_a(k)y_{n-k} = \text{local level} \\ b_n &= \sum_{k=0}^{N-1} h_b(k)y_{n-k} = \text{local slope} \end{aligned} \tag{6.4.5}$$

Since, $\hat{y}_{n+\tau/n} = a_n + b_n\tau$, it is seen that the local level a_n is equal to $\hat{y}_{n/n}$. Similarly, the sum a_n+b_n is the one-step-ahead forecast $\hat{y}_{n+1/n}$ obtained by extrapolating to time instant $n+1$ by extending the local level a_n along the straight line with slope b_n. This is depicted in the figure below. The sum, a_n+b_n, can be generated directly by the predictive FIR filter, $h_1(k)=h_a(k)+h_b(k)$, obtained by setting $\tau=1$ in (6.4.1):

$$\boxed{h_1(k)=\frac{2(2N-2-3k)}{N(N-1)}\,,\quad k=0,1,\dots,N-1} \quad \text{(predictive FIR filter)} \tag{6.4.6}$$



The filters $h_a(k)$, $h_b(k)$, and $h_1(k)$ find application in the technical analysis of financial markets [280]. Indeed, the filter $h_a(k)$ is equivalent to the so-called *linear regression* indicator, $h_b(k)$ corresponds to the *linear regression slope* indicator, and $h_1(k)$, to the *time series forecast* indicator. We discuss these in more detail, as well as other indicators, in Sections 6.14–6.24.

[†] a,b are the same as the components c_0,c_1 of the vector $\mathbf{c}$.

More generally, for order d polynomials, it follows from the solution (6.3.8), that the FIR filters $\mathbf{h}_\tau$ satisfy the moment constraints $S_N^T\mathbf{h}_\tau = \mathbf{u}_\tau$, or, component-wise:

$$\sum_{k=0}^{N-1}(-k)^r h_\tau(k) = \tau^r\,,\quad r=0,1,\dots,d \tag{6.4.7}$$

In fact, the solution $\mathbf{h}_\tau = S_N(S_N^T S_N)^{-1}\mathbf{u}_\tau$ is recognized (from Chap. 15) to be the minimum-norm, pseudoinverse, solution of the under-determined system $S_N^T\mathbf{h}=\mathbf{u}_\tau$, that is, it has minimum norm, or, minimum noise-reduction ratio, $\mathcal{R}=\mathbf{h}^T\mathbf{h}=\min$. A direct derivation is as follows. Introduce a $(d+1)\times 1$ vector of Lagrange multipliers, $\boldsymbol{\lambda}=[\lambda_0,\lambda_1,\dots,\lambda_d]^T$, and incorporate the constraint into the performance index,

$$\mathcal{J} = \mathbf{h}^T\mathbf{h} + 2\boldsymbol{\lambda}^T(\mathbf{u}_\tau - S_N^T\mathbf{h}) = \min$$

Then, its minimization leads to,

$$\frac{\partial\mathcal{J}}{\partial\mathbf{h}} = 2\mathbf{h} - 2S_N\boldsymbol{\lambda} = 0 \quad\Rightarrow\quad \mathbf{h} = S_N\boldsymbol{\lambda}$$

and, imposing the constraint $S_N^T\mathbf{h}=\mathbf{u}_\tau$ leads to the solutions for $\boldsymbol{\lambda}$ and for $\mathbf{h}$,

$$\mathbf{u}_\tau = S_N^T\mathbf{h} = S_N^T S_N\boldsymbol{\lambda} \quad\Rightarrow\quad \boldsymbol{\lambda} = (S_N^T S_N)^{-1}\mathbf{u}_\tau \quad\Rightarrow\quad \mathbf{h} = S_N\boldsymbol{\lambda} = S_N(S_N^T S_N)^{-1}\mathbf{u}_\tau$$

Returning to Eq. (6.4.3) and setting $\tau=0$, we note that the $d=1$ local-level filter $h_a(k)$ satisfies the explicit constraints:

$$\sum_{k=0}^{N-1} h_a(k) = 1\,,\quad \sum_{k=0}^{N-1} k h_a(k) = 0 \tag{6.4.8}$$

The latter implies that its lag parameter $\bar{n}$ is zero, and therefore, straight-line inputs will appear at the output undelayed (see Example 6.5.1). It has certain limitations as a lowpass filter that we discuss in Sec. 6.10, but its NRR is decreasing with N:

$$\mathcal{R} = \frac{2(2N-1)}{N(N+1)} \tag{6.4.9}$$

A direct consequence of Eq. (6.4.7) is that the filter $h_\tau(k)$ generates the exact predicted value of any polynomial of degree d, that is, for any polynomial $P(x)$ with degree up to d in the variable x, we have the exact convolutional result,

$$\boxed{\sum_{k=0}^{N-1} P(n-k)h_\tau(k) = P(n+\tau)}\,,\quad \text{with } \deg(P)\le d \tag{6.4.10}$$

6.5 Higher-Order Exponential Smoothing

For any value of d, the FIR filters $\mathbf{h}_\tau$ have length N and act on the N-dimensional data vector $\mathbf{y}(n)=[y_n,y_{n-1},\dots,y_{n-N+1}]^T$. By contrast, the exponential smoother weights $\mathbf{h}_\tau(n)$ have an ever increasing length. Therefore, it proves convenient to recast them recursively in time. The resulting recursive algorithm bears a very close similarity to the so-called *exact recursive-least-squares* (RLS) adaptive filters that we discuss in Chap. 16. Let us define the quantities,

$$
\begin{aligned}
R_n &= S_n^T W_n S_n = \sum_{k=0}^{n} \lambda^k \mathbf{u}_{-k}\mathbf{u}_{-k}^T = (d+1)\times(d+1) \text{ matrix}\\
\mathbf{r}_n &= S_n^T W_n \mathbf{y}(n) = \sum_{k=0}^{n} \lambda^k \mathbf{u}_{-k} y_{n-k} = (d+1)\times 1 \text{ vector}
\end{aligned}
\tag{6.5.1}
$$

Then, the optimal polynomial coefficients (6.3.7) are:

$$
\mathbf{c}(n) = R_n^{-1}\mathbf{r}_n \tag{6.5.2}
$$

Clearly, the invertibility of R_n requires that $n \geq d$, which we will assume from now on. The sought recursions relate $\mathbf{c}(n)$ to the optimal coefficients $\mathbf{c}(n-1) = R_{n-1}^{-1}\mathbf{r}_{n-1}$ at the previous time instant $n-1$. Therefore, we must actually assume that $n > d$. To proceed, we note that the basis vector $\mathbf{u}_\tau = [1, \tau, \tau^2, \dots, \tau^d]^T$ satisfies the time-propagation property:

$$
\mathbf{u}_{\tau+1} = F\mathbf{u}_\tau \tag{6.5.3}
$$

where F is a $(d+1)\times(d+1)$ unit lower triangular matrix whose ith row consists of the binomial coefficients:

$$
F_{ij} = \binom{i}{j}, \quad 0 \leq i \leq d, \quad 0 \leq j \leq i \tag{6.5.4}
$$

This follows from the binomial expansion:

$$
(\tau+1)^i = \sum_{j=0}^{i}\binom{i}{j}\tau^j
$$

Some examples of the F matrices are for $d = 0, 1, 2$:

$$
F = [1], \quad F = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}, \quad F = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 2 & 1 \end{bmatrix} \tag{6.5.5}
$$

It follows from Eq. (6.5.3) that $\mathbf{u}_\tau = F\mathbf{u}_{\tau-1}$, and inverting $\mathbf{u}_{\tau-1} = F^{-1}\mathbf{u}_\tau$. The inverse matrix $G = F^{-1}$ will also be unit lower triangular with nonzero matrix elements obtained from the binomial expansion of $(\tau-1)^i$:

$$
G_{ij} = (-1)^{i-j}\binom{i}{j}, \quad 0 \leq i \leq d, \quad 0 \leq j \leq i \tag{6.5.6}
$$

For example, we have for $d = 0, 1, 2$,

$$
G = [1], \quad G = \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix}, \quad G = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 1 & -2 & 1 \end{bmatrix} \tag{6.5.7}
$$

It follows from $\mathbf{u}_{\tau-1} = G\mathbf{u}_\tau$ that $\mathbf{u}_{-k-1} = G\mathbf{u}_{-k}$. This implies the following recursion for R_n:

$$
\begin{aligned}
R_n &= \sum_{k=0}^{n}\lambda^k \mathbf{u}_{-k}\mathbf{u}_{-k}^T = \mathbf{u}_0\mathbf{u}_0^T + \sum_{k=1}^{n}\lambda^k \mathbf{u}_{-k}\mathbf{u}_{-k}^T\\
&= \mathbf{u}_0\mathbf{u}_0^T + \lambda\sum_{k=1}^{n}\lambda^{k-1}\mathbf{u}_{-k}\mathbf{u}_{-k}^T\\
&= \mathbf{u}_0\mathbf{u}_0^T + \lambda\sum_{k=0}^{n-1}\lambda^{k}\mathbf{u}_{-k-1}\mathbf{u}_{-k-1}^T\\
&= \mathbf{u}_0\mathbf{u}_0^T + \lambda G\left(\sum_{k=0}^{n-1}\lambda^k\mathbf{u}_{-k}\mathbf{u}_{-k}^T\right)G^T = \mathbf{u}_0\mathbf{u}_0^T + \lambda G R_{n-1}G^T
\end{aligned}
$$

where in the third line we changed summation variables from k to $k-1$, and in the fourth, we used $\mathbf{u}_{-k-1} = G\mathbf{u}_{-k}$. Similarly, we have for $\mathbf{r}_n$,

$$
\begin{aligned}
\mathbf{r}_n &= \sum_{k=0}^{n}\lambda^k\mathbf{u}_{-k}y_{n-k} = \mathbf{u}_0 y_n + \sum_{k=1}^{n}\lambda^k\mathbf{u}_{-k}y_{n-k}\\
&= \mathbf{u}_0 y_n + \lambda\sum_{k=0}^{n-1}\lambda^k\mathbf{u}_{-k-1}y_{n-k-1}\\
&= \mathbf{u}_0 y_n + \lambda G\left(\sum_{k=0}^{n-1}\lambda^k\mathbf{u}_{-k}y_{(n-1)-k}\right) = \mathbf{u}_0 y_n + \lambda G\mathbf{r}_{n-1}
\end{aligned}
$$

Thus, $R_n, \mathbf{r}_n$ satisfy the recursions:

$$
\begin{aligned}
R_n &= \mathbf{u}_0\mathbf{u}_0^T + \lambda G R_{n-1}G^T\\
\mathbf{r}_n &= \mathbf{u}_0 y_n + \lambda G\mathbf{r}_{n-1}
\end{aligned}
\tag{6.5.8}
$$

and they may be initialized to zero, $R_{-1} = 0$ and $\mathbf{r}_{-1} = 0$. Using $\hat{y}_{n+\tau/n} = \mathbf{u}_\tau^T\mathbf{c}(n)$, we may define the smoothed estimates, predictions, and the corresponding errors:

$$
\begin{aligned}
\hat{y}_{n/n} &= \mathbf{u}_0^T\mathbf{c}(n)\,, & e_{n/n} &= y_n - \hat{y}_{n/n}\\
\hat{y}_{n+1/n} &= \mathbf{u}_1^T\mathbf{c}(n) = \mathbf{u}_0^T F^T\mathbf{c}(n)\,, & e_{n+1/n} &= y_{n+1} - \hat{y}_{n+1/n}\\
\hat{y}_{n/n-1} &= \mathbf{u}_1^T\mathbf{c}(n-1) = \mathbf{u}_0^T F^T\mathbf{c}(n-1)\,, & e_{n/n-1} &= y_n - \hat{y}_{n/n-1}
\end{aligned}
\tag{6.5.9}
$$

where we used $\mathbf{u}_1 = F\mathbf{u}_0$. In the language of RLS adaptive filters, we may refer to $\hat{y}_{n/n-1}$ and $\hat{y}_{n/n}$ as the a priori and a posteriori estimates of y_n, respectively. Using the recursions (6.5.8), we may now obtain a recursion for $\mathbf{c}(n)$. Using $\mathbf{c}(n-1) = R_{n-1}^{-1}\mathbf{r}_{n-1}$ and the matrix relationship $GF = I$, we have,

$$
\begin{aligned}
R_n\mathbf{c}(n) &= \mathbf{r}_n = \mathbf{u}_0 y_n + \lambda G\mathbf{r}_{n-1} = \mathbf{u}_0 y_n + \lambda G R_{n-1}\mathbf{c}(n-1)\\
&= \mathbf{u}_0 y_n + \lambda G R_{n-1}G^T F^T\mathbf{c}(n-1) = \mathbf{u}_0 y_n + (R_n - \mathbf{u}_0\mathbf{u}_0^T)F^T\mathbf{c}(n-1)\\
&= R_n F^T\mathbf{c}(n-1) + \mathbf{u}_0\big(y_n - \mathbf{u}_0^T F^T\mathbf{c}(n-1)\big) = R_n F^T\mathbf{c}(n-1) + \mathbf{u}_0(y_n - \hat{y}_{n/n-1})\\
&= R_n F^T\mathbf{c}(n-1) + \mathbf{u}_0 e_{n/n-1}
\end{aligned}
$$

where in the second line we used $\lambda G R_{n-1} G^T = R_n - \mathbf{u}_0\mathbf{u}_0^T$. Multiplying both sides by R_n^{-1}, we obtain,

$$\mathbf{c}(n) = F^T\mathbf{c}(n-1) + R_n^{-1}\mathbf{u}_0 e_{n/n-1} \tag{6.5.10}$$

Again, in the language of RLS adaptive filters, we define the so-called a posteriori and a priori "Kalman gain" vectors $\mathbf{k}_n$ and $\mathbf{k}_{n/n-1}$,

$$\mathbf{k}_n = R_n^{-1}\mathbf{u}_0\,, \quad \mathbf{k}_{n/n-1} = \lambda^{-1}F^T R_{n-1}^{-1} F\mathbf{u}_0 \tag{6.5.11}$$

and the "likelihood" variables,

$$\nu_n = \mathbf{u}_0^T\mathbf{k}_{n/n-1} = \lambda^{-1}\mathbf{u}_0^T F^T R_{n-1}^{-1} F\mathbf{u}_0 = \lambda^{-1}\mathbf{u}_1^T R_{n-1}^{-1}\mathbf{u}_1\,, \quad \mu_n = \frac{1}{1+\nu_n} \tag{6.5.12}$$

Starting with the recursion $R_n = \mathbf{u}_0\mathbf{u}_0^T + \lambda G R_{n-1} G^T$ and multiplying both sides by R_n^{-1} from the left, then by F^T from the right, then by R_{n-1}^{-1} from the left, and then by F from the right, we may easily derive the equivalent relationship:

$$\lambda^{-1}F^T R_{n-1}^{-1} F = R_n^{-1}\mathbf{u}_0\,\lambda^{-1}\mathbf{u}_0^T F^T R_{n-1}^{-1} F + R_n^{-1} \tag{6.5.13}$$

Multiplying on the right by $\mathbf{u}_0$ and using the definitions (6.5.11), we find

$$\mathbf{k}_{n/n-1} = \mathbf{k}_n\nu_n + \mathbf{k}_n = (1+\nu_n)\mathbf{k}_n\,, \quad \text{or,}$$

$$\mathbf{k}_n = \mu_n\mathbf{k}_{n/n-1} \tag{6.5.14}$$

Substituting this into (6.5.13), we obtain a recursion for the inverse matrix R_n^{-1}, which is effectively a variant of the matrix inversion lemma:

$$R_n^{-1} = \lambda^{-1}F^T R_{n-1}^{-1} F - \mu_n\mathbf{k}_{n/n-1}\mathbf{k}_{n/n-1}^T \tag{6.5.15}$$

This also implies that the parameter μ_n can be expressed as

$$\mu_n = 1 - \mathbf{u}_0^T R_n^{-1}\mathbf{u}_0 = 1 - \mathbf{u}_0^T\mathbf{k}_n \tag{6.5.16}$$

The a priori and a posteriori errors are also proportional to each other. Using (6.5.16), we find,

$$\hat{y}_{n/n} = \mathbf{u}_0^T\mathbf{c}(n) = \mathbf{u}_0^T\big(F^T\mathbf{c}(n-1) + \mathbf{k}_n e_{n/n-1}\big) = \hat{y}_{n/n-1} + (1-\mu_n)e_{n/n-1} = y_n - \mu_n e_{n/n-1}$$

which implies that

$$e_{n/n} = \mu_n e_{n/n-1} \tag{6.5.17}$$

The coefficient updates (6.5.10) may now be expressed as:

$$\mathbf{c}(n) = F^T\mathbf{c}(n-1) + \mathbf{k}_n e_{n/n-1} \tag{6.5.18}$$

We summarize the complete set of computational steps for high-order exponential smoothing. We recall that the invertibility conditions require that we apply the recursions for $n > d$:

$$\boxed{\begin{array}{ll}
1. & \mathbf{k}_{n/n-1} = \lambda^{-1}F^T R_{n-1}^{-1} F\mathbf{u}_0 = \lambda^{-1}F^T R_{n-1}^{-1}\mathbf{u}_1 \\
2. & \nu_n = \mathbf{u}_0^T\mathbf{k}_{n/n-1}\,, \quad \mu_n = 1/(1+\nu_n) \\
3. & \mathbf{k}_n = \mu_n\mathbf{k}_{n/n-1} \\
4. & \hat{y}_{n/n-1} = \mathbf{u}_1^T\mathbf{c}(n-1)\,, \quad e_{n/n-1} = y_n - \hat{y}_{n/n-1} \\
5. & e_{n/n} = \mu_n e_{n/n-1}\,, \quad \hat{y}_n = y_n - e_{n/n} \\
6. & \mathbf{c}(n) = F^T\mathbf{c}(n-1) + \mathbf{k}_n e_{n/n-1} \\
7. & R_n^{-1} = \lambda^{-1}F^T R_{n-1}^{-1} F - \mu_n\mathbf{k}_{n/n-1}\mathbf{k}_{n/n-1}^T
\end{array}} \tag{6.5.19}$$

For $0 \le n \le d$, the fitting may be done with polynomials of varying degree $d_n = n$, and the coefficient estimate computed by explicit matrix inversion, $\mathbf{c}(n) = R_n^{-1}\mathbf{r}_n$. The above computational steps and initialization have been incorporated into the MATLAB function ema with usage:

```
C = ema(y,d,lambda);        % exponential moving average - exact version
```

The input y is an L-dimensional vector (row or column) of samples to be smoothed, with a total number $L > d$, and C is an $L\times(d+1)$ matrix whose nth row is the coefficient vector $\mathbf{c}(n)^T$. Thus, the first column, holds the smoothed estimate, the second column the estimated first derivative, and so on.

To understand the initialization process, consider an input sequence $\{y_0, y_1, y_2, \dots\}$ and the $d = 1$ smoother. At $n = 0$, we use a smoother of order $d_0 = 0$, constructing the quantities $R_0, \mathbf{r}_0$ using the definition (6.5.1):

$$R_0 = [1]\,, \quad \mathbf{r}_0 = [y_0] \quad \Rightarrow \quad \mathbf{c}(0) = R_0^{-1}\mathbf{r}_0 = y_0$$

Next, at $n = 1$ we use a $d_1 = 1$ smoother, and definition (6.5.1) now implies,

$$\begin{aligned}
R_1 &= \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} + \lambda\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} = \begin{bmatrix} 1+\lambda & -\lambda \\ -\lambda & \lambda \end{bmatrix} \\
\mathbf{r}_1 &= \begin{bmatrix} 1 \\ 0 \end{bmatrix} y_1 + \lambda\begin{bmatrix} 1 \\ -1 \end{bmatrix} y_0 = \begin{bmatrix} y_1 + \lambda y_0 \\ -\lambda y_0 \end{bmatrix}
\end{aligned} \quad \Rightarrow \quad \mathbf{c}(1) = R_1^{-1}\mathbf{r}_1 = \begin{bmatrix} y_1 \\ y_1 - y_0 \end{bmatrix}$$

Starting with $R_1, \mathbf{r}_1$, the recursion (6.5.8) may then be continued for $n \ge d + 1 = 2$. If we had instead $d = 2$, then there is one more initialization step, giving

$$R_2 = \begin{bmatrix} 1+\lambda+\lambda^2 & -\lambda-2\lambda^2 & \lambda+4\lambda^2 \\ -\lambda-2\lambda^2 & \lambda+4\lambda^2 & -\lambda-8\lambda^2 \\ \lambda+4\lambda^2 & -\lambda-8\lambda^2 & \lambda+16\lambda^2 \end{bmatrix}, \quad \mathbf{r}_2 = \begin{bmatrix} y_2 + \lambda y_1 + \lambda^2 y_0 \\ -\lambda y_1 - 2\lambda^2 y_0 \\ \lambda y_1 + 4\lambda^2 y_0 \end{bmatrix}$$

resulting in

$$\mathbf{c}(2) = R_2^{-1}\mathbf{r}_2 = \begin{bmatrix} y_2 \\ 1.5y_2 - 2y_1 + 0.5y_0 \\ 0.5y_2 - y_1 + 0.5y_0 \end{bmatrix} \tag{6.5.20}$$

We note that the first $d + 1$ smoothed values get initialized to the first $d + 1$ values of the input sequence.

Example 6.5.1: Fig. 6.5.1 shows the output of the exact exponential smoother with $d = 1$ and $\lambda = 0.9$ applied on the same noiseless input s_n of Example 6.1.3. In addition, it shows the $d = 1$ FIR filter $h_a(k)$ designed to have zero lag according to Eq. (6.4.4).

Because $d = 1$, both filters can follow a linear signal. The input s_n (dashed curve) is barely visible under the filter outputs (solid curves). The length of the FIR filter was chosen according to the rule $N = (1+\lambda)/(1-\lambda)$.

The following MATLAB code generates the two graphs; it uses the function `upulse` which is a part of the OSP toolbox that generates a unit-pulse of prescribed duration

```
n = 0:300;
s = (20 + 0.8*n) .* upulse(n,75) + ...          % upulse is in the OSP toolbox
    (80 - 0.3*(n-75)) .* upulse(n-75,150) + ...
    (35 + 0.4*(n-225)) .* upulse(n-225,76);

la = 0.9; al = 1-la; d = 1;

C = ema(s,d,la);                          % exact exponential smoother output
x = C(:,1);

N = round((1+la)/(1-la));                 % equivalent FIR length, N=19

k=0:N-1;
ha = 2*(2*N-1-3*k)/N/(N+1);               % zero-lag FIR filter

xh = filter(ha,1,s);                      % FIR filter output

figure; plot(n,s,'--', n,x,'-');          % left graph
figure; plot(n,s,'--', n,xh,'-');         % right graph
```

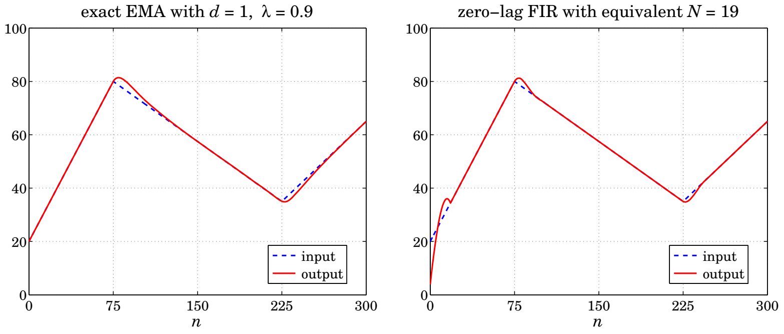


Fig. 6.5.1 Exact EMA with order $d = 1$, and zero-lag FIR filter with equivalent length.

Next, we add some noise $y_n = s_n + 4v_n$, where v_n is zero-mean, unit-variance, white noise. The top two graphs of Fig. 6.5.2 show the noisy signal y_n and the response of the exact EMA with $d = 0$ and $\lambda = 0.9$.

The bottom two graphs show the exact EMA with $d = 1$ as well as the response of the same zero-lag FIR filter to the noisy data. □

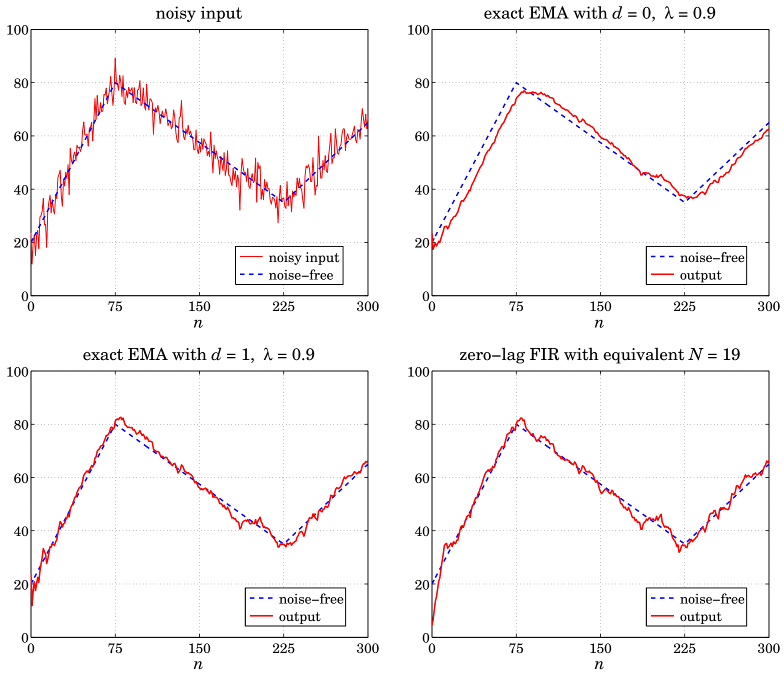


Fig. 6.5.2 EMA with order $d = 1$, and zero-lag FIR filter with equivalent length.

6.6 Steady-State Exponential Smoothing

Next, we look in more detail at the cases $d = 0, 1, 2$, which are the most commonly used in practice, with $d = 1$ providing the best performance and flexibility. We denote the polynomial coefficients by:

$$\mathbf{c}(n) = [a_n], \quad \mathbf{c}(n) = \begin{bmatrix} a_n \\ b_n \end{bmatrix}, \quad \mathbf{c}(n) = \begin{bmatrix} a_n \\ b_n \\ c_n \end{bmatrix} \tag{6.6.1}$$

Then, with $\mathbf{u}_\tau = [1]$, $\mathbf{u}_\tau = [1, \tau]^T$, and $\mathbf{u}_\tau = [1, \tau, \tau^2]^T$, the implied predicted estimates will be for arbitrary τ:

$$\begin{aligned} \hat{y}_{n+\tau/n} &= \mathbf{u}_\tau^T \mathbf{c}(n) = a_n \\ \hat{y}_{n+\tau/n} &= \mathbf{u}_\tau^T \mathbf{c}(n) = a_n + b_n\tau \\ \hat{y}_{n+\tau/n} &= \mathbf{u}_\tau^T \mathbf{c}(n) = a_n + b_n\tau + c_n\tau^2 \end{aligned} \tag{6.6.2}$$

Thus, a_n, b_n represent local estimates of the level and slope, respectively, and $2c_n$

represents the acceleration. The one-step-ahead predictions are,

$$\begin{aligned}
\hat{y}_{n/n-1} &= \mathbf{u}_1^T\mathbf{c}(n-1) = a_{n-1}\\
\hat{y}_{n/n-1} &= \mathbf{u}_1^T\mathbf{c}(n-1) = a_{n-1} + b_{n-1}\\
\hat{y}_{n/n-1} &= \mathbf{u}_1^T\mathbf{c}(n-1) = a_{n-1} + b_{n-1} + c_{n-1}
\end{aligned} \tag{6.6.3}$$

Denoting the a posteriori gains $\mathbf{k}_n$ by,

$$\mathbf{k}_n = [\alpha(n)]\,, \quad \mathbf{k}_n = \begin{bmatrix} \alpha_1(n) \\ \alpha_2(n) \end{bmatrix}, \quad \mathbf{k}_n = \begin{bmatrix} \alpha_1(n) \\ \alpha_2(n) \\ \alpha_3(n) \end{bmatrix} \tag{6.6.4}$$

then, the coefficient updates (6.5.18) take the forms, where $e_{n/n-1} = y_n - \hat{y}_{n/n-1}$,

$$\begin{aligned}
a_n &= a_{n-1} + \alpha(n)e_{n/n-1}\\
\begin{bmatrix} a_n \\ b_n \end{bmatrix} &= \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} a_{n-1} \\ b_{n-1} \end{bmatrix} + \begin{bmatrix} \alpha_1(n) \\ \alpha_2(n) \end{bmatrix} e_{n/n-1}\\
\begin{bmatrix} a_n \\ b_n \\ c_n \end{bmatrix} &= \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} a_{n-1} \\ b_{n-1} \\ c_{n-1} \end{bmatrix} + \begin{bmatrix} \alpha_1(n) \\ \alpha_2(n) \\ \alpha_3(n) \end{bmatrix} e_{n/n-1}
\end{aligned} \tag{6.6.5}$$

Since $\mathbf{k}_n = R_n^{-1}\mathbf{u}_0$, the gains depend only on λ and n and converge to steady-state values for large n. For example, for $d = 0$, we have,

$$R_n = \sum_{k=0}^{n} \lambda^k = \frac{1-\lambda^{n+1}}{1-\lambda} \quad \Rightarrow \quad k_n = R_n^{-1} = \frac{1-\lambda}{1-\lambda^{n+1}} \to 1-\lambda \equiv \alpha$$

Thus, the steady-state form of the $d = 0$ EMA smoother is as expected:

$$\boxed{\begin{aligned} e_{n/n-1} &= y_n - \hat{y}_{n/n-1} = y_n - a_{n-1}\\ a_n &= a_{n-1} + (1-\lambda)e_{n/n-1} \end{aligned}} \quad \text{(single EMA, } d = 0\text{)} \tag{6.6.6}$$

initialized as usual at $a_{-1} = y_0$. The corresponding likelihood variable $\mu_n = 1 - \mathbf{u}_0^T\mathbf{k}_n$ tends to $\mu = 1 - (1-\lambda) = \lambda$. Similarly, we find for $d = 1$,

$$R_n = \sum_{k=0}^{n} \lambda^k \begin{bmatrix} 1 \\ -k \end{bmatrix}[1, -k] = \sum_{k=0}^{n} \lambda^k \begin{bmatrix} 1 & -k \\ -k & k^2 \end{bmatrix} \equiv \begin{bmatrix} R_{00}(n) & R_{01}(n) \\ R_{10}(n) & R_{11}(n) \end{bmatrix}$$

where

$$\begin{aligned}
R_{00}(n) &= \frac{1-\lambda^{n+1}}{1-\lambda}\,, \quad R_{01}(n) = R_{10}(n) = \frac{-\lambda + \lambda^{n+1}[1 + n(1-\lambda)]}{(1-\lambda)^2}\\
R_{11}(n) &= \frac{\lambda(1+\lambda) - \lambda^{n+1}[1 + \lambda - 2n(1-\lambda) + n^2(1-\lambda)^2]}{(1-\lambda)^3}
\end{aligned}$$

which have the limit as $n \to \infty$,

$$\begin{aligned}
R_n \to R &= \frac{1}{(1-\lambda)^3}\begin{bmatrix} (1-\lambda)^2 & -\lambda(1-\lambda) \\ -\lambda(1-\lambda) & \lambda(1+\lambda) \end{bmatrix}\\
R^{-1} &= \begin{bmatrix} 1-\lambda^2 & (1-\lambda)^2 \\ (1-\lambda)^2 & \lambda^{-1}(1-\lambda)^3 \end{bmatrix}
\end{aligned} \tag{6.6.7}$$

It follows that the asymptotic gain vector $\mathbf{k} = R^{-1}\mathbf{u}_0$ will be the first column of R^{-1}:

$$\mathbf{k}_n \to \mathbf{k} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} = \begin{bmatrix} 1-\lambda^2 \\ (1-\lambda)^2 \end{bmatrix} \tag{6.6.8}$$

and the steady-state version of the $d = 1$ EMA smoother becomes:

$$\boxed{\begin{aligned} & e_{n/n-1} = y_n - \hat{y}_{n/n-1} = y_n - (a_{n-1} + b_{n-1})\\ & \begin{bmatrix} a_n \\ b_n \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} a_{n-1} \\ b_{n-1} \end{bmatrix} + \begin{bmatrix} 1-\lambda^2 \\ (1-\lambda)^2 \end{bmatrix} e_{n/n-1} \end{aligned}} \quad \text{(double EMA, } d = 1\text{)} \tag{6.6.9}$$

with estimated level $\hat{y}_{n/n} = a_n$ and one-step-ahead prediction $\hat{y}_{n+1/n} = a_n + b_n$. The corresponding limit of the likelihood parameter is $\mu = 1 - \mathbf{u}_0^T\mathbf{k} = 1 - (1-\lambda^2) = \lambda^2$. The difference equation may be initialized at $a_{-1} = 2y_0 - y_1$ and $b_{-1} = y_1 - y_0$ to agree with the first outputs of the exact smoother. Indeed, iterating up to $n = 1$, we find the same answer for $\mathbf{c}(1)$ as the exact smoother:

$$\begin{aligned}
\begin{bmatrix} a_0 \\ b_0 \end{bmatrix} &= \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} a_{-1} \\ b_{-1} \end{bmatrix} + \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} e_{0/-1} = \begin{bmatrix} y_0 \\ y_1 - y_0 \end{bmatrix}\\
\begin{bmatrix} a_1 \\ b_1 \end{bmatrix} &= \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} a_0 \\ b_0 \end{bmatrix} + \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} e_{1/0} = \begin{bmatrix} y_1 \\ y_1 - y_0 \end{bmatrix}
\end{aligned}$$

Of course, other initializations are possible, a common one being to fit a straight line to the first few input samples and choose the intercept and slope as the initial values. This is the default method used by the function `stema` (see below). For the $d = 2$ case, the asymptotic matrix R is

$$R = \sum_{k=0}^{\infty} \lambda^k \mathbf{u}_{-k}\mathbf{u}_{-k}^T = \sum_{k=0}^{\infty} \lambda^k \begin{bmatrix} 1 & -k & k^2 \\ -k & k^2 & -k^3 \\ k^2 & -k^3 & k^4 \end{bmatrix}$$

which may be summed to

$$R = \begin{bmatrix} \dfrac{1}{1-\lambda} & -\dfrac{\lambda}{(1-\lambda)^2} & \dfrac{\lambda(1+\lambda)}{(1-\lambda)^3} \\[2ex] -\dfrac{\lambda}{(1-\lambda)^2} & \dfrac{\lambda(1+\lambda)}{(1-\lambda)^3} & -\dfrac{\lambda(1+4\lambda+\lambda^2)}{(1-\lambda)^4} \\[2ex] \dfrac{\lambda(1+\lambda)}{(1-\lambda)^3} & -\dfrac{\lambda(1+4\lambda+\lambda^2)}{(1-\lambda)^4} & \dfrac{\lambda(1+\lambda)(1+10\lambda+\lambda^2)}{(1-\lambda)^5} \end{bmatrix}$$

with an inverse

$$R^{-1} = \begin{bmatrix} 1-\lambda^3 & \frac{3}{2}(1+\lambda)(1-\lambda)^2 & \frac{1}{2}(1-\lambda)^3 \\ \frac{3}{2}(1+\lambda)(1-\lambda)^2 & \frac{(1+\lambda)(1-\lambda)^3(1+9\lambda)}{4\lambda^2} & \frac{(1-\lambda)^4(1+3\lambda)}{4\lambda^2} \\ \frac{1}{2}(1-\lambda)^3 & \frac{(1-\lambda)^4(1+3\lambda)}{4\lambda^2} & \frac{(1-\lambda)^5}{4\lambda^2} \end{bmatrix}$$

The asymptotic gain vector $\mathbf{k} = R^{-1}\mathbf{u}_0$ and μ parameter are,

$$\mathbf{k} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} = \begin{bmatrix} 1-\lambda^3 \\ \frac{3}{2}(1+\lambda)(1-\lambda)^2 \\ \frac{1}{2}(1-\lambda)^3 \end{bmatrix}, \qquad \mu = 1-\alpha_1 = \lambda^3 \tag{6.6.10}$$

and the steady-state $d = 2$ EMA smoother becomes:

$$\boxed{\begin{aligned} & e_{n/n-1} = y_n - \hat{y}_{n/n-1} = y_n - (a_{n-1}+b_{n-1}+c_{n-1}) \\ & \begin{bmatrix} a_n \\ b_n \\ c_n \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_{n-1} \\ b_{n-1} \\ c_{n-1} \end{bmatrix} + \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} e_{n/n-1} \end{aligned}} \quad \text{(triple EMA, } d = 2\text{)} \tag{6.6.11}$$

They may be initialized to reach the same values at $n = 2$ as the exact smoother, that is, Eq. (6.5.20). This requirement gives:

$$\begin{bmatrix} a_{-1} \\ b_{-1} \\ c_{-1} \end{bmatrix} = \begin{bmatrix} y_2 - 3y_1 + 3y_0 \\ -1.5y_2 + 4y_1 - 2.5y_0 \\ 0.5y_2 - y_1 + 0.5y_0 \end{bmatrix} \quad \Rightarrow \quad \begin{bmatrix} a_2 \\ b_2 \\ c_2 \end{bmatrix} = \begin{bmatrix} y_2 \\ 1.5y_2 - 2y_1 + 0.5y_0 \\ 0.5y_2 - y_1 + 0.5y_0 \end{bmatrix}$$

Alternatively, they may be initialized by fitting a second degree polynomial to the first few input samples, as is done by default in the function `stema`, or they may be initialized to a zero vector, or to any other values, for example, $[a_{-1}, b_{-1}, c_{-1}] = [y_0, 0, 0]$.

For arbitrary polynomial order d, the matrix R_n converges to a $(d+1)\times(d+1)$ matrix R that must satisfy the Lyapunov-type equation:

$$R = \mathbf{u}_0\mathbf{u}_0^T + \lambda G R G^T \tag{6.6.12}$$

where G is the backward boost matrix, $G = F^{-1}$. This follows by considering the limit of Eq. (6.5.1) as $n \to \infty$ and using the property $\mathbf{u}_{-k-1} = G\mathbf{u}_{-k}$. Multiplying from the left by F, and noting that $F\mathbf{u}_0 = \mathbf{u}_1$, we have

$$FR = \mathbf{u}_1\mathbf{u}_0^T + \lambda R G^T \tag{6.6.13}$$

Taking advantage of the unit-lower-triangular nature of F and G, this equation can be written component-wise as follows:

$$\sum_{k=0}^{i} F_{ik}R_{kj} = u_1(i)u_0(j) + \lambda \sum_{k=0}^{j} R_{ik}G_{jk}\,, \quad 0 \le i,j \le d \tag{6.6.14}$$

Noting that $u_1(i) = 1$ and $u_0(j) = \delta(j)$, and setting first $i = j = 0$, we find

$$R_{00} = 1 + \lambda R_{00} \quad \Rightarrow \quad R_{00} = \frac{1}{1-\lambda} \tag{6.6.15}$$

Then, setting $i = 0$ and $1 \le j \le d$,

$$R_{0j} = \lambda \sum_{k=0}^{j} R_{ik}G_{jk} = \lambda R_{0j} + \lambda \sum_{k=0}^{j-1} R_{0k}G_{jk}$$

which can be solved recursively for R_{0j}:

$$R_{0j} = R_{j0} = \frac{\lambda}{1-\lambda} \sum_{k=0}^{j-1} R_{0k}G_{jk}\,, \quad j = 1, 2, \dots, d \tag{6.6.16}$$

Next, take $i \ge 1$ and $j \ge i$, and use the symmetry of R:

$$R_{ij} + \sum_{k=0}^{i-1} F_{ik}R_{kj} = \lambda R_{ij} + \lambda \sum_{k=0}^{j-1} R_{ik}G_{jk}$$

or, for $\quad i = 1, 2, \dots, d, \quad j = i, i+1, \dots, d$,

$$R_{ij} = R_{ji} = \frac{1}{1-\lambda}\left[\lambda \sum_{k=0}^{j-1} R_{ik}G_{jk} - \sum_{k=0}^{i-1} F_{ik}R_{kj}\right] \tag{6.6.17}$$

To clarify the computations, we give the MATLAB code below:

```
R(1,1) = 1/(1-lambda);
for j=2:d+1,
  R(1,j) = lambda * R(1,1:j-1) * G(j,1:j-1)' / (1-lambda);
  R(j,1) = R(1,j);
end
for i=2:d+1,
  for j=i:d+1,
    R(i,j) = (lambda*R(i,1:j-1)*G(j,1:j-1)' - F(i,1:i-1)*R(1:i-1,j))/(1-lambda);
    R(j,i) = R(i,j);
  end
end
```

Once R is determined, one may calculate the gain vector $\mathbf{k} = R^{-1}\mathbf{u}_0$. Then, the overall filtering algorithm can be stated as follows, for $n \ge 0$,

$$\boxed{\begin{aligned} & \hat{y}_{n/n-1} = \mathbf{u}_1^T\mathbf{c}(n-1) \\ & e_{n/n-1} = y_n - \hat{y}_{n/n-1} \\ & \mathbf{c}(n) = F^T\mathbf{c}(n-1) + \mathbf{k}\,e_{n/n-1} \end{aligned}} \quad \text{(steady-state EMA)} \tag{6.6.18}$$

which requires specification of the initial vector $\mathbf{c}(-1)$. The transfer function from the input y_n to the signals $\mathbf{c}(n)$ can be determined by taking z-transforms of Eq. (6.6.18):

$$\mathbf{C}(z) = z^{-1}F^T\mathbf{C}(z) + \mathbf{k}\big(Y(z) - z^{-1}\mathbf{u}_1^T\mathbf{C}(z)\big)\,, \quad \text{or,}$$

$$\boxed{\mathbf{H}(z)=\frac{\mathbf{C}(z)}{Y(z)}=\big[I-(F^T-\mathbf{k}\mathbf{u}_1^T)z^{-1}\big]^{-1}\mathbf{k}} \tag{6.6.19}$$

The computational steps (6.6.18) have been incorporated in the MATLAB function `stema`, with usage,

```
C = stema(y,d,lambda,cinit);        % steady-state exponential moving average
```

where `C,y,d,lambda` have the same meaning as in the function `ema`. The parameter `cinit` is a $(d+1)\times 1$ column vector that represents the initial vector $\mathbf{c}(-1)$. If omitted, it defaults to fitting a polynomial of order d to the first L input samples, where L is the effective length corresponding to λ, that is, $L=(1+\lambda)/(1-\lambda)$. The fitting is carried out with the help of the function `lpbasis` from Chap. 3, and is given in MATLAB notation by:

```
cinit = lpbasis(L,d,-1)\y(1:L);     % fit order-d polynomial to first L inputs
```

where the fit is carried out with respect to the time origin $n=-1$. The length L must be less than the length of the input vector $\mathbf{y}$. If not, another, shorter L can be used. Other initialization possibilities for $\mathbf{c}_{\text{init}}$ are summarized in the help file for `stema`.

To clarify the fitting operation, we note that fitting the first L samples y_n, $n=0,1,\dots,L-1$, to a polynomial of degree d centered at $n=-1$ amounts to the minimization of the performance index:

$$\mathcal{J}=\sum_{n=0}^{L-1}(y_n-p_n)^2=\min\,,\quad p_n=\sum_{i=0}^{d}(n+1)^i c_i=\mathbf{u}_{n+1}^T\mathbf{c}$$

which can be written compactly as

$$\mathcal{J}=\|\mathbf{y}-S\mathbf{c}\|^2=\min\,,\quad S=[\mathbf{u}_1,\mathbf{u}_2,\dots,\mathbf{u}_{n+1},\dots,\mathbf{u}_L]^T$$

with solution $\mathbf{c}=(S^TS)^{-1}S^T\mathbf{y}=S\backslash\mathbf{y}$ in MATLAB notation.[†] The actual fitted values $\mathbf{p}=[p_0,p_1,\dots,p_{L-1}]^T$ are then computed by $\mathbf{p}=S\mathbf{c}$.

Selecting $n=-1$ as the centering time, assumes that the filtering operation will start at $n=0$ requiring therefore the value $\mathbf{c}(-1)$. The centering can be done at any other reference time $n=n_0$, for example, one would choose $n_0=L-1$ if the filtering operation were to start at $n=L$. The performance index would be then,

$$\mathcal{J}=\sum_{n=0}^{L-1}(y_n-p_n)^2=\min\,,\quad p_n=\sum_{i=0}^{d}(n-n_0)^i c_i=\mathbf{u}_{n-n_0}^T\bar{\mathbf{c}}$$

with another set of coefficients $\bar{\mathbf{c}}$. The MATLAB implementation is in this case,

```
cinit = lpbasis(L,d,n0)\y(1:L);     % fit order-d polynomial to first L inputs
```

From $\mathbf{u}_{n+1}=F\mathbf{u}_n$, we obtain $\mathbf{u}_{n+1}=F^{n_0+1}\mathbf{u}_{n-n_0}$. By requiring that the fitted polynomials be the same, $p_n=\mathbf{u}_{n+1}^T\mathbf{c}=\mathbf{u}_{n-n_0}^T\bar{\mathbf{c}}$, it follows that,

$$\bar{\mathbf{c}}=(F^T)^{n_0+1}\mathbf{c} \tag{6.6.20}$$

[†] assuming that S has full rank, which requires $L>d$.

In Sec. 6.8, we discuss the connection to conventional multiple exponential smoothing obtained by filtering in cascade through $d+1$ copies of a single exponential smoothing filter $H(z)=\alpha/(1-\lambda z^{-1})$, that is, through $\big[H(z)\big]^{d+1}$. Example 6.11.1 illustrates the above initialization methods, as well as how to map the initial values of $\mathbf{c}(n)$ to the initial values of the cascaded filter outputs.

6.7 Smoothing Parameter Selection

The performance of the steady-state EMA may be judged by computing the covariance of the estimates $\mathbf{c}(n)$, much like the case of the $d=0$ smoother. Starting with $\mathbf{c}(n)=R_n^{-1}\mathbf{r}_n$ and $\mathbf{r}_n=S_n^TW_n\mathbf{y}(n)$, we obtain for the correlation matrix,

$$E[\mathbf{c}(n)\mathbf{c}^T(n)]=R_n^{-1}S_n^TW_nE[\mathbf{y}(n)\mathbf{y}^T(n)]W_nS_nR_n^{-1}$$

and for the corresponding covariance matrix,

$$\Sigma_{cc}=R_n^{-1}S_n^TW_n\Sigma_{yy}W_nS_nR_n^{-1} \tag{6.7.1}$$

Under the typical assumption that y_n is white noise, we have $\Sigma_{yy}=\sigma_y^2I_{n+1}$, where I_{n+1} is the $(n+1)$-dimensional unit matrix. Then,

$$\Sigma_{cc}=\sigma_y^2R_n^{-1}Q_nR_n^{-1}\,,\quad Q_n=S_n^TW_n^2S_n \tag{6.7.2}$$

In the limit $n\to\infty$, the matrices R_n,Q_n tend to steady-state values, so that

$$\Sigma_{cc}=\sigma_y^2R^{-1}QR^{-1} \tag{6.7.3}$$

where the limit matrices R,Q are given by

$$R=\sum_{k=0}^{\infty}\lambda^k\mathbf{u}_{-k}\mathbf{u}_{-k}\,,\quad Q=\sum_{k=0}^{\infty}\lambda^{2k}\mathbf{u}_{-k}\mathbf{u}_{-k} \tag{6.7.4}$$

Since $\hat{y}_{n/n}=\mathbf{u}_0^T\mathbf{c}(n)$ and $\hat{y}_{n+1/n}=\mathbf{u}_1^T\mathbf{c}(n)$, the corresponding variances will be:

$$\sigma_{\hat{y}_{n/n}}^2=\mathbf{u}_0^T\Sigma_{cc}\mathbf{u}_0\,,\quad \sigma_{\hat{y}_{n+1/n}}^2=\mathbf{u}_1^T\Sigma_{cc}\mathbf{u}_1\equiv\sigma_{\hat{y}}^2\,, \tag{6.7.5}$$

Because y_n was assumed to be an uncorrelated sequence, the two terms in the prediction error $e_{n+1/n}=y_{n+1}-\hat{y}_{n+1/n}$ will be uncorrelated since $\hat{y}_{n+1/n}$ depends only on data up to n. Therefore, the variance of the prediction error $e_{n+1/n}$ will be:

$$\sigma_e^2=\sigma_y^2+\sigma_{\hat{y}}^2=\sigma_y^2\big[1+\mathbf{u}_1^TR^{-1}QR^{-1}\mathbf{u}_1\big] \tag{6.7.6}$$

For the case $d=0$, we have

$$R=\sum_{k=0}^{\infty}\lambda^k=\frac{1}{1-\lambda}\,,\quad Q=\sum_{k=0}^{\infty}\lambda^{2k}=\frac{1}{1-\lambda^2}$$

which gives the usual results:

$$\sigma_{\hat{y}}^2=\Sigma_{cc}=\frac{1-\lambda}{1+\lambda}\sigma_y^2\,,\quad \sigma_e^2=\sigma_y^2+\sigma_{\hat{y}}^2=\frac{2}{1+\lambda}\sigma_y^2$$

For $d = 1$, we have as in Eq. (6.6.7),

$$R = \frac{1}{(1-\lambda)^3}\begin{bmatrix}(1-\lambda)^2 & -\lambda(1-\lambda)\\ -\lambda(1-\lambda) & \lambda(1+\lambda)\end{bmatrix},$$

with the Q matrix being obtained from R by replacing $\lambda \to \lambda^2$,

$$Q = \frac{1}{(1-\lambda^2)^3}\begin{bmatrix}(1-\lambda^2)^2 & -\lambda^2(1-\lambda^2)\\ -\lambda^2(1-\lambda^2) & \lambda^2(1+\lambda^2)\end{bmatrix}$$

It follows then that

$$\Sigma_{cc} = \sigma_y^2 R^{-1}QR^{-1} = \frac{1-\lambda}{(1+\lambda)^3}\begin{bmatrix}1+4\lambda+5\lambda^2 & (1-\lambda)(1+3\lambda)\\ (1-\lambda)(1+3\lambda) & 2(1-\lambda)^2\end{bmatrix} \tag{6.7.7}$$

The diagonal entries are the variances of the level and slope signals a_n, b_n:

$$\sigma_a^2 = \frac{(1-\lambda)(1+4\lambda+5\lambda^2)}{(1+\lambda)^3}\sigma_y^2\,, \qquad \sigma_b^2 = \frac{2(1-\lambda)^3}{(1+\lambda)^3}\sigma_y^2 \tag{6.7.8}$$

For the prediction variance, we find

$$\sigma_{\hat{y}}^2 = \sigma_y^2\,\mathbf{u}_1^T(R^{-1}QR^{-1})\mathbf{u}_1 = \frac{(1-\lambda)(\lambda^2+4\lambda+5)}{(1+\lambda)^3}\sigma_y^2 \tag{6.7.9}$$

which gives for the prediction error:

$$\sigma_e^2 = \sigma_y^2 + \sigma_{\hat{y}}^2 = \left[1 + \frac{(1-\lambda)(\lambda^2+4\lambda+5)}{(1+\lambda)^3}\right]\sigma_y^2 = \frac{2(3+\lambda)}{(1+\lambda)^3}\sigma_y^2 \tag{6.7.10}$$

In order to achieve an equivalent smoothing performance with a $d = 0$ EMA, one must equate the corresponding prediction variances, or mean-square errors. If λ_0, λ_1 denote the equivalent $d = 0$ and $d = 1$ parameters, the condition reads:

$$\frac{2}{1+\lambda_0} = \frac{2(3+\lambda_1)}{(1+\lambda_1)^3} \quad\Rightarrow\quad \lambda_0 = \frac{(1+\lambda_1)^3}{3+\lambda_1} - 1 \tag{6.7.11}$$

Eq. (6.7.11) may also be solved for λ_1 in terms of λ_0,

$$\lambda_1 = \frac{1}{3}D_0 + \frac{1+\lambda_0}{D_0} - 1\,, \quad D_0 = \left[27(1+\lambda_0) + \sqrt{27(1+\lambda_0)^2(26-\lambda_0)}\right]^{1/3} \tag{6.7.12}$$

Setting $\lambda_0 = 0$ gives $D_0 = (27+3\sqrt{78})^{1/3}$ and $\lambda_1 = 0.5214$. For all $\lambda_1 \geq 0.5214$, the equivalent λ_0 is non-negative and the NRR $\sigma_{\hat{y}}^2/\sigma_y^2$ of the prediction filter remains less than unity.

The corresponding FIR averager would have length $N_0 = (1+\lambda_0)/(1-\lambda_0)$, whereas an equivalent zero-lag FIR filter should have length N_1 that matches the corresponding NRRs. We have from Eq. (6.4.9):

$$\frac{2(2N_1-1)}{N_1(N_1+1)} = \frac{1-\lambda_0}{1+\lambda_0}$$

which gives,

$$\lambda_0 = \frac{N_1^2 - 3N_1 + 2}{N_1^2 + 5N_1 - 2} \quad\Leftrightarrow\quad N_1 = \frac{3 + 5\lambda_0 + \sqrt{33\lambda_0^2 + 30\lambda_0 + 1}}{2(1-\lambda_0)} \tag{6.7.13}$$

The MATLAB function `emap` implements Eq. (6.7.12),

```
la1 = emap(la0);          % mapping equivalent λ's between d = 0 and d = 1 EMAs
```

The computed λ_1 is an increasing function of λ_0 and varies over $0.5214 \leq \lambda_1 \leq 1$ as λ_0 varies over $0 \leq \lambda_0 \leq 1$.

Example 6.7.1: The lower-right graph of Fig. 6.7.1 shows a zero-lag FIR filter defined by Eq. (6.4.4) with length $N_1 = 100$ and applied to the noisy signal shown on the upper-left graph. The noisy signal was $y_n = 20 + 0.2n + 4v_n$, for $0 \leq n \leq 300$, with zero-mean, unit-variance, white noise v_n.

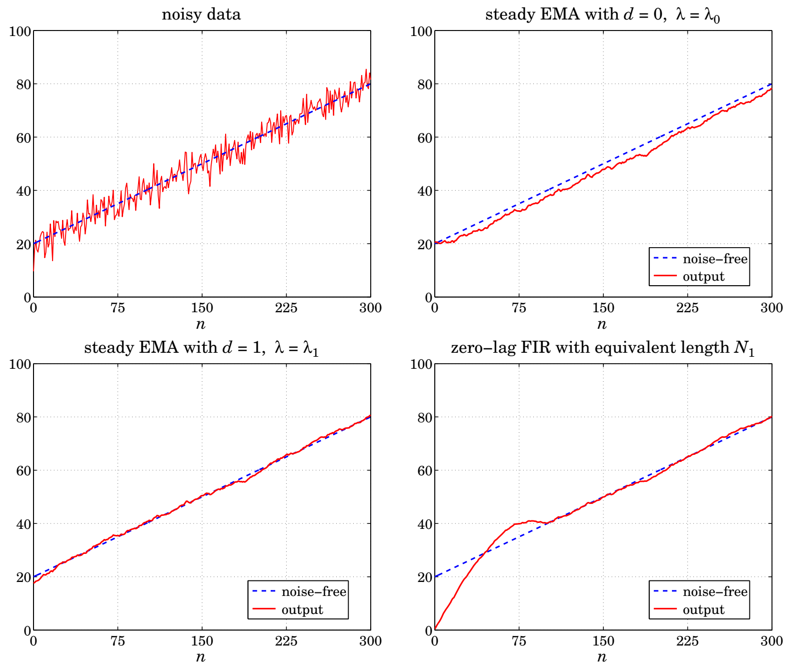


Fig. 6.7.1 Comparison of two equivalent steady-state EMAs with equivalent zero-lag FIR.

The equivalent EMA parameter for $d = 0$ was found from (6.7.13) to be $\lambda_0 = 0.9242$, which was then mapped to $\lambda_1 = 0.9693$ of an equivalent $d = 1$ EMA using Eq. (6.7.12). The upper-right graph shows the $d = 0$ EMA output, and the lower-left graph, the $d = 1$ EMA. The steady-state version was used for both EMAs with default initializations. The following MATLAB code illustrates the calculations:

```
t = 0:300; s = 20 + 0.2*t;
randn('state', 1000);
y = s + 4 * randn(size(t));                         % noisy input

N1 = 100;
la0 = (N1^2-3*N1+2)/(N1^2+5*N1-2);                  % equivalent λ0
la1 = emap(la0);                                    % equivalent λ1

C = stema(y,0,la0); x0 = C(:,1);                    % steady EMA with d = 0, λ = λ0
C = stema(y,1,la1); x1 = C(:,1);                    % steady EMA with d = 1, λ = λ1

k=0:N1-1; h = 2*(2*N1-1-3*k)/N1/(N1+1);             % zero-lag FIR of length N1
% h = lpinterp(N1,1,-(N1-1)/2)';                    % alternative calculation
xh = filter(h,1,y);

figure; plot(t,y,'-', t,s,'-');   figure; plot(t,s,'--', t,x0,'-');
figure; plot(t,s,'--', t,x1,'-'); figure; plot(t,s,'--', t,xh,'-');
```

We observe that all three outputs achieve comparable noise reduction. The $d = 0$ EMA suffers from the expected delay. Both the $d = 1$ EMA and the zero-lag FIR filter follow the straight-line input with no delay, after the initial transients have subsided. □

The choice of the parameter λ is more of an art than science. There do exist, however, criteria that determine an "optimum" value. Given the prediction $\hat{y}_{n/n-1} = \mathbf{u}_1^T\mathbf{c}(n-1)$ of y_n, and prediction error $e_{n/n-1} = y_n - \hat{y}_{n/n-1}$, the following criteria, to be minimized with respect to λ, are widely used:

$$\boxed{\begin{array}{ll} \text{MSE} = \text{mean}\big(e_{n/n-1}^2\big)\,, & \text{(mean square error)} \\ \text{MAE} = \text{mean}\big(|e_{n/n-1}|\big)\,, & \text{(mean absolute error)} \\ \text{MAPE} = \text{mean}\big(100|e_{n/n-1}/y_n|\big)\,, & \text{(mean absolute percentage error)} \end{array}} \tag{6.7.14}$$

where the mean may be taken over the entire data set or over a portion of it. Usually, the criteria are evaluated over a range of λ's and the minimum is selected. Typically, the criteria tend to underestimate the value of λ, that is, they produce too small a λ to be useful for smoothing purposes. Even so, the optimum λ has been used successfully for forecasting applications. The MATLAB function `emaerr` calculates these criteria for any vector of λ's and determines the optimum λ in that range:

```
[err,lopt] = emaerr(y,d,lambda,type);      % mean error criteria
```

where `type` takes one of the string values `'mse'`, `'mae'`, `'mape'` and `err` is the criterion evaluated at the vector `lambda`, and `lopt` is the corresponding optimum λ.

Example 6.7.2: Fig. 6.7.2 shows the same Dow-Jones data of Example 6.1.2. The MSE criterion was searched over the range $0.1 \le \lambda \le 0.9$. The upper-left graph shows the MSE versus λ. The minimum occurs at $\lambda_{\text{opt}} = 0.61$.

The upper-right graph shows the $d = 1$ exact EMA run with $\lambda = \lambda_{\text{opt}}$. The EMA output is too rough to provide adequate smoothing. The other criteria are even worse. The MAE and MAPE optima both occur at $\lambda_{\text{opt}} = 0.56$. For comparison, the bottom two graphs show the $d = 1$ exact EMA run with the two higher values $\lambda = 0.90$ and $\lambda = 0.95$. The MATLAB code generating these graphs was as follows:

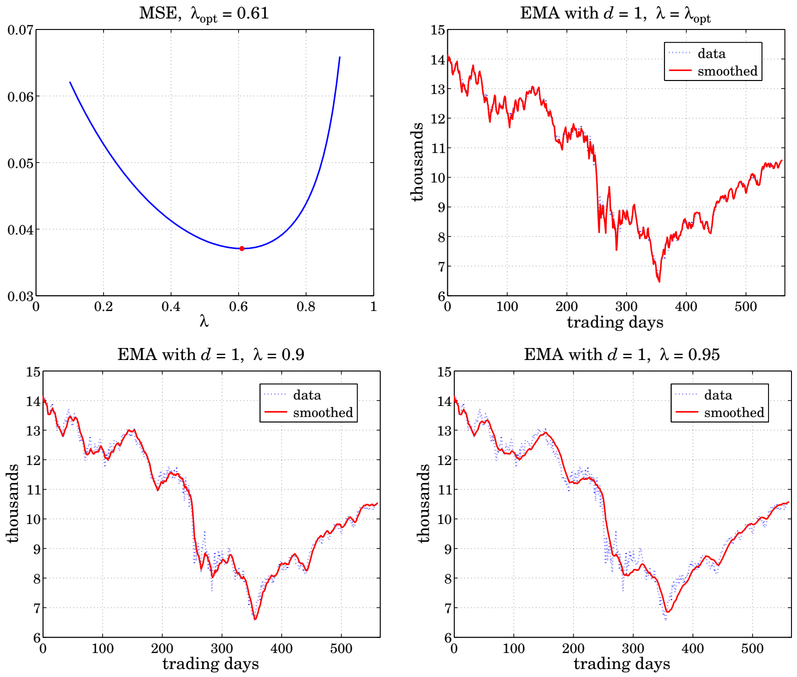


Fig. 6.7.2 MSE criterion for the DJIA data.

```
Y = loadfile('dow-oct07-dec09.dat');             % read data
y = Y(:,1)/1000;   n = (0:length(y)-1)';

d = 1; u1 = ones(d+1,1);                         % polynomial order for EMA

la = linspace(0.1, 0.9, 81);                     % range of λ's to search
[err,lopt] = emaerr(y,d,la,'mse');               % evaluate MSE at this range of λ's

figure; plot(la,err, lopt,min(err),'.');         % upper-left graph

C = ema(y,d,lopt); yhat = C*u1;
figure; plot(n,y,':', n,yhat,'-');               % upper-right graph

la=0.90; C = ema(y,d,la); yhat = C*u1;           % bottom-left graph
figure; plot(n,y,':', n,yhat,'-');               % use la=0.95 for bottom-right
```

We note that the $d = 1$ smoother is more capable in following the signal than the $d = 0$ one. We plotted the forecasted value $\hat{y}_{n+1/n} = \mathbf{c}^T(n)\mathbf{u}_1$ versus n. Because the output matrix `C` from the `ema` function has the $\mathbf{c}^T(n)$ as its rows, the entire vector of forecasted values can be calculated by acting by `C` on the unit vector $\mathbf{u}_1$, that is, `yhat = C*u1`. □

6.8 Single, Double, and Triple Exponential Smoothing

Single exponential smoothing is the same as first-order, $d = 0$, steady-state exponential smoothing. We recall its filtering equation and corresponding transfer function:

$$a_n^{[1]} = \lambda a_{n-1}^{[1]} + \alpha y_n\,, \quad H^{[1]}(z) = H(z) = \frac{\alpha}{1-\lambda z^{-1}} \tag{6.8.1}$$

where $\alpha = 1-\lambda$. Double smoothing refers to filtering $a_n^{[1]}$ one more time through the same filter $H(z)$; and triple smoothing, two more times. The resulting filtering equations and transfer functions (from the overall input y_n to the final outputs) are:

$$\begin{aligned} a_n^{[2]} &= \lambda a_{n-1}^{[2]} + \alpha a_n^{[1]}\,, \quad H^{[2]}(z) = \left(\frac{\alpha}{1-\lambda z^{-1}}\right)^2 \\ a_n^{[3]} &= \lambda a_{n-1}^{[3]} + \alpha a_n^{[2]}\,, \quad H^{[3]}(z) = \left(\frac{\alpha}{1-\lambda z^{-1}}\right)^3 \end{aligned} \tag{6.8.2}$$

$$y_n \longrightarrow \boxed{H} \longrightarrow a_n^{[1]} \longrightarrow \boxed{H} \longrightarrow a_n^{[2]} \longrightarrow \boxed{H} \longrightarrow a_n^{[3]}$$

Thus, the filter $H(z)$ acts once, twice, three times, or in general $d{+}1$ times, in cascade, producing the outputs,

$$y_n \longrightarrow \boxed{H} \longrightarrow a_n^{[1]} \longrightarrow \boxed{H} \longrightarrow a_n^{[2]} \longrightarrow \boxed{H} \longrightarrow a_n^{[3]} \longrightarrow \cdots \longrightarrow a_n^{[d]} \longrightarrow \boxed{H} \longrightarrow a_n^{[d+1]} \tag{6.8.3}$$

The transfer function and the corresponding causal impulse response from y_n to the r-th output $a_n^{[r]}$ are, for $r = 1, 2, \dots, d{+}1$ with $u(n)$ denoting the unit-step function:

$$H^{[r]}(z) = \big[H(z)\big]^r = \left(\frac{\alpha}{1-\lambda z^{-1}}\right)^r \quad \Leftrightarrow \quad h^{[r]}(n) = \alpha^r \lambda^n \frac{(n+r-1)!}{n!(r-1)!}\,u(n) \tag{6.8.4}$$

Double and triple exponential smoothing are in a sense equivalent to the $d = 1$ and $d = 2$ steady-state EMA filters of Eq. (6.6.9) and (6.6.11). From Eq. (6.6.19), which in this case reads $\mathbf{H}(z) = [H_a(z), H_b(z)]^T$, we may obtain the transfer functions from y_n to the outputs a_n and b_n:

$$H_a(z) = \frac{(1-\lambda)(1+\lambda-2\lambda z^{-1})}{(1-\lambda z^{-1})^2}\,, \quad H_b(z) = \frac{(1-\lambda)^2(1-z^{-1})}{(1-\lambda z^{-1})^2} \tag{6.8.5}$$

It is straightforward to verify that H_a and H_b are related to H and H^2 by

$$\boxed{\begin{aligned} H_a &= 2H - H^2 = 1-(1-H)^2 \\ H_b &= \frac{\alpha}{\lambda}(H - H^2) \end{aligned}} \quad \begin{aligned} &\text{(local level filter)} \\ &\text{(local slope filter)} \end{aligned} \tag{6.8.6}$$

In the time domain this implies the following relationships between the a_n, b_n signals and the cascaded outputs $a_n^{[1]}, a_n^{[2]}$:

$$\boxed{\begin{aligned} a_n &= 2a_n^{[1]} - a_n^{[2]} = \text{local level} \\ b_n &= \frac{\alpha}{\lambda}\big(a_n^{[1]} - a_n^{[2]}\big) = \text{local slope} \end{aligned}} \tag{6.8.7}$$

which can be written in a 2×2 matrix form:

$$\begin{bmatrix} a_n \\ b_n \end{bmatrix} = \begin{bmatrix} 2 & -1 \\ \alpha/\lambda & -\alpha/\lambda \end{bmatrix} \begin{bmatrix} a_n^{[1]} \\ a_n^{[2]} \end{bmatrix} \quad \Rightarrow \quad \begin{bmatrix} a_n^{[1]} \\ a_n^{[2]} \end{bmatrix} = \begin{bmatrix} 1 & -\lambda/\alpha \\ 1 & -2\lambda/\alpha \end{bmatrix} \begin{bmatrix} a_n \\ b_n \end{bmatrix} \tag{6.8.8}$$

Similarly, for the $d = 2$ case, the transfer functions from y_n to a_n, b_n, c_n are:

$$\begin{aligned} H_a(z) &= \frac{\alpha\big[1+\lambda+\lambda^2 - 3\lambda(1+\lambda)z^{-1} + 3\lambda^2 z^{-2}\big]}{(1-\lambda z^{-1})^3} \\ H_b(z) &= \frac{1}{2}\,\frac{\alpha^2(1-z^{-1})\big[3(1+\lambda) - (5\lambda+1)z^{-1}\big]}{(1-\lambda z^{-1})^3} \\ H_c(z) &= \frac{1}{2}\,\frac{\alpha^3(1-z^{-1})^2}{(1-\lambda z^{-1})^3} \end{aligned} \tag{6.8.9}$$

which are related to H, H^2, H^3 by the matrix transformation:

$$\begin{bmatrix} H \\ H^2 \\ H^3 \end{bmatrix} = \begin{bmatrix} 1 & -\lambda/\alpha & \lambda(\lambda+1)/\alpha^2 \\ 1 & -2\lambda/\alpha & 2\lambda(2\lambda+1)/\alpha^2 \\ 1 & -3\lambda/\alpha & 3\lambda(3\lambda+1)/\alpha^2 \end{bmatrix} \begin{bmatrix} H_a \\ H_b \\ H_c \end{bmatrix} \tag{6.8.10}$$

implying the transformation between the outputs:

$$\begin{bmatrix} a_n^{[1]} \\ a_n^{[2]} \\ a_n^{[3]} \end{bmatrix} = \begin{bmatrix} 1 & -\lambda/\alpha & \lambda(\lambda+1)/\alpha^2 \\ 1 & -2\lambda/\alpha & 2\lambda(2\lambda+1)/\alpha^2 \\ 1 & -3\lambda/\alpha & 3\lambda(3\lambda+1)/\alpha^2 \end{bmatrix} \begin{bmatrix} a_n \\ b_n \\ c_n \end{bmatrix} \tag{6.8.11}$$

with corresponding inverse relationships,

$$\begin{bmatrix} H_a \\ H_b \\ H_c \end{bmatrix} = \frac{1}{2\lambda^2} \begin{bmatrix} 6\lambda^2 & -6\lambda^2 & 2\lambda^2 \\ \alpha(1+5\lambda) & -2\alpha(1+4\lambda) & \alpha(1+3\lambda) \\ \alpha^2 & -2\alpha^2 & \alpha^2 \end{bmatrix} \begin{bmatrix} H \\ H^2 \\ H^3 \end{bmatrix} \tag{6.8.12}$$

$$\begin{bmatrix} a_n \\ b_n \\ c_n \end{bmatrix} = \frac{1}{2\lambda^2} \begin{bmatrix} 6\lambda^2 & -6\lambda^2 & 2\lambda^2 \\ \alpha(1+5\lambda) & -2\alpha(1+4\lambda) & \alpha(1+3\lambda) \\ \alpha^2 & -2\alpha^2 & \alpha^2 \end{bmatrix} \begin{bmatrix} a_n^{[1]} \\ a_n^{[2]} \\ a_n^{[3]} \end{bmatrix} \tag{6.8.13}$$

In particular, we have:

$$H_a = 3H - 3H^2 + H^3 = 1-(1-H)^3 \tag{6.8.14}$$

and

$$\hat{y}_{n/n} = a_n = 3a_n^{[1]} - 3a_n^{[2]} + a_n^{[3]} \tag{6.8.15}$$

More generally, for an order-d polynomial EMA, we have [243],

$$H_a = 1-(1-H)^{d+1} \tag{6.8.16}$$

$$\hat{y}_{n/n} = a_n = -\sum_{r=1}^{d+1} (-1)^r \binom{d+1}{r} a_n^{[r]} \tag{6.8.17}$$

6.9 Exponential Smoothing and Tukey's Twicing Operation

There is an interesting interpretation [255] of these results in terms of Tukey's *twicing* operation [257] and its generalizations to *thricing*, and so on. To explain twicing, consider a smoothing operation, which for simplicity we may assume that it can be represented by the matrix operation $\hat{\mathbf{y}} = H\mathbf{y}$, or if so preferred, in the z-domain as the multiplication of z-transforms $\hat{Y}(z) = H(z)Y(z)$.

The resulting residual error is $\mathbf{e} = \mathbf{y} - \hat{\mathbf{y}} = (I - H)\mathbf{y}$. In the twicing procedure, the residuals are filtered through the same smoothing operation, which will smooth them further, $\hat{\mathbf{e}} = H\mathbf{e} = H(I - H)\mathbf{y}$, and the result is added to the original estimate to get an improved estimate:

$$\hat{\mathbf{y}}_{\text{impr}} = \hat{\mathbf{y}} + \hat{\mathbf{e}} = \big[H + H(I - H)\big]\mathbf{y} = \big[2H - H^2\big]\mathbf{y} \tag{6.9.1}$$

which is recognized as the operation (6.8.6). The process can be continued by repeating it on the residuals of the residuals, and so on. For example, at the next step, one would compute the new residual $\mathbf{r} = \mathbf{e} - \hat{\mathbf{e}} = (I - H)\mathbf{e} = (I - H)^2\mathbf{y}$, then filter it through H, $\hat{\mathbf{r}} = H\mathbf{r} = H(I - H)^2\mathbf{y}$, and add it to get the "thriced" improved estimate:

$$\hat{\mathbf{y}}_{\text{impr}} = \hat{\mathbf{y}} + \hat{\mathbf{e}} + \hat{\mathbf{r}} = \big[H + H(I - H) + H(I - H)^2\big]\mathbf{y} = \big[3H - 3H^2 + H^3\big]\mathbf{y} \tag{6.9.2}$$

which is Eq. (6.8.14). The process can be continued d times, resulting in,

$$\hat{\mathbf{y}}_{\text{impr}} = H\big[I + (I - H) + (I - H)^2 + \cdots + (I - H)^d\big]\mathbf{y} = \big[I - (I - H)^{d+1}\big]\mathbf{y} \tag{6.9.3}$$

Twicing and its generalizations can be applied with benefit to any smoothing operation, for example, if we used an LPRS filter designed by $B = \text{lprs}(N, d, s)$, the computational steps for twicing would be:

$$\hat{\mathbf{y}} = \text{lpfilt}(B, \mathbf{y}) \;\Rightarrow\; \mathbf{e} = \mathbf{y} - \hat{\mathbf{y}} \;\Rightarrow\; \hat{\mathbf{e}} = \text{lpfilt}(B, \mathbf{e}) \;\Rightarrow\; \hat{\mathbf{y}}_{\text{impr}} = \hat{\mathbf{y}} + \hat{\mathbf{e}}$$

A limitation of twicing is that, while it drastically improves the passband of a lowpass smoothing filter, it worsens its stopband. To see this, we write for the improved transfer function, $H_{\text{impr}}(z) = 1 - \big(1 - H(z)\big)^{d+1}$. In the passband, H is near unity, say $H \approx 1 - \epsilon$, with $|\epsilon| \ll 1$, then,

$$H_{\text{impr}} = 1 - \big(1 - (1 - \epsilon)\big)^{d+1} = 1 - \epsilon^{d+1}$$

thus, making the passband ripple $(d+1)$ orders of magnitude smaller. On the other hand, in the stopband, H is near zero, say $H \approx \pm\epsilon$, resulting in a worsened stopband,

$$H_{\text{impr}} = 1 - (1 \mp \epsilon)^{d+1} \approx 1 - \big(1 \mp (d + 1)\epsilon\big) = \pm(d + 1)\epsilon$$

The twicing procedure has been generalized by Kaiser and Hamming [258] to the so-called "filter sharpening" that improves both the passband and the stopband. For example, the lowest-order filter combination that achieves this is,

$$H_{\text{impr}} = H^2(3 - 2H) = 3H^2 - 2H^3 \tag{6.9.4}$$

where now both the passband and stopband ripples are replaced by $\epsilon \to \epsilon^2$. More generally, it can be shown [258] that the filter that achieves pth order tangency at $H = 0$ and qth order tangency at $H = 1$ is given by

$$H_{\text{impr}} = H^{p+1} \sum_{k=0}^{q} \frac{(p + k)!}{p!\,k!}(1 - H)^k \tag{6.9.5}$$

The multiple exponential moving average case corresponds to $p = 0$ and $q = d$, resulting in $H_{\text{impr}} = 1 - (1 - H)^{d+1}$, whereas Eq. (6.9.4) corresponds to $p = q = 1$.

6.10 Twicing and Zero-Lag Filters

Another advantage of twicing and, more generally, filter sharpening is that the resulting improved smoothing filter always has zero lag, that is, $\bar{n} = 0$.

Indeed, assuming unity DC gain for the original filter, $H(z)\big|_{z=1} = 1$, it is straightforward to show that the general formula (6.9.5) gives for the first derivative:

$$H'_{\text{impr}}(z)\big|_{z=1} = 0 \tag{6.10.1}$$

which implies that its lag is zero, $\bar{n} = 0$, by virtue of Eq. (6.1.18). The twicing procedure, or its generalizations, for getting zero-lag filters is not limited to the exponential moving average. It can be applied to any other lowpass filter. For example, if we apply it to an ordinary length-N FIR averager, we would obtain:

$$H(z) = \frac{1}{N}\sum_{n=0}^{N-1} z^{-n} = \frac{1}{N}\,\frac{1 - z^{-N}}{1 - z^{-1}} \quad\Rightarrow\quad H_a(z) = 2H(z) - H^2(z) \tag{6.10.2}$$

The impulse response of $H_a(z)$ can be shown to be, where $0 \le n \le 2(N - 1)$,

$$h_a(n) = \left(\frac{2N - 1 - n}{N^2}\right)\big[u(n) - 2u(n - N) + u(n - 2N + 1)\big] \tag{6.10.3}$$

It is straightforward to show that $\bar{n}_a = 0$ and that its noise-reduction ratio is

$$\mathcal{R} = \frac{8N^2 - 6N + 1}{3N^3} \tag{6.10.4}$$

Because of their zero-lag property, double and triple EMA filters are used as *trend indicators* in the financial markets [297,298]. The application of twicing to the modified exponential smoother of Eq. (2.3.5) gives rise to a similar indicator called the *instantaneous trendline* [285], and further discussed in Problem 6.8. We discuss such market indicators in Sections 6.14–6.24.

The zero-lag property for a causal lowpass filter comes at a price, namely, that although its magnitude response is normalized to unity at $\omega = 0$ and has a flat derivative there, it typically bends upwards developing a bandpass peak near DC before it attenuates to smaller values at higher frequencies. See, for example, Fig. 6.10.1.

This behavior might be deemed to be objectionable since it tends to unevenly amplify the low-frequency passband of the filter.

To clarify these remarks, consider a lowpass filter $H(\omega)$ (with real-valued impulse response h_n) satisfying the gain and flatness conditions $H(0)=1$ and $H'(0)=0$ at $\omega=0$. The flatness condition implies the zero-lag property $\bar{n}=0$. Using these two conditions, it is straightforward to verify that the second derivative of the magnitude response at DC is given by:

$$\frac{d^2}{d\omega^2}\big|H(\omega)\big|^2_{\omega=0} = 2\,\mathrm{Re}\big[H''(0)\big] + 2|H'(0)|^2 = 2\,\mathrm{Re}\big[H''(0)\big] = -2\sum_{n=0}^{\infty} n^2 h_n \tag{6.10.5}$$

Because $\bar{n}=\sum_n nh_n = 0$, it follows that some of the coefficients h_n must be negative, which can cause (6.10.5) to become positive, implying that $\omega=0$ is a local minimum and hence the response will rise for ωs immediately beyond DC. This is demonstrated for example in Problem 6.7 by Eq. (6.25.1), so that,

$$\frac{d^2}{d\omega^2}\big|H(\omega)\big|^2_{\omega=0} = -2\sum_{n=0}^{\infty} n^2 h_n = \frac{4\lambda^2}{(1-\lambda)^2}$$

A similar calculation yields the result,

$$\frac{d^2}{d\omega^2}\big|H(\omega)\big|^2_{\omega=0} = -2\sum_{n=0}^{\infty} n^2 h_n = \frac{1}{3}(N-1)(N-2)$$

for the optimum zero-lag FIR filter of Eq. (6.4.4),

$$h_a(k) = \frac{2(2N-1-3k)}{N(N+1)}\,, \quad k = 0,1,\dots,N-1 \tag{6.10.6}$$

We note that the first derivative of the magnitude response $|H(\omega)|^2$ is always zero at DC, regardless of whether the filter has zero lag or not. Indeed, it follows from $H(0)=1$ and the reality of h_n that,

$$\frac{d}{d\omega}\big|H(\omega)\big|^2_{\omega=0} = 2\,\mathrm{Re}\big[H'(0)\big] = 0 \tag{6.10.7}$$

Example 6.10.1: *Zero-Lag Filters.* In Problem 6.7, we saw that the double EMA filter has transfer function and magnitude response:

$$H_a(z) = \frac{(1-\lambda)(1+\lambda-2\lambda z^{-1})}{(1-\lambda z^{-1})^2}$$

$$|H_a(\omega)|^2 = \frac{(1-\lambda)^2\big[1+2\lambda+5\lambda^2-4\lambda(1+\lambda)\cos\omega\big]}{\big[1-2\lambda\cos\omega+\lambda^2\big]^2}$$

and that a secondary peak develops at,

$$\cos\omega_{\max} = \frac{1+4\lambda-\lambda^2}{2(1+\lambda)}\,, \quad |H_a(\omega_{\max})|^2 = \frac{(1+\lambda)^2}{1+2\lambda}$$

The left graph of Fig. 6.10.1 shows the magnitude responses for the two cases of $\lambda=0.8$ and $\lambda=0.9$. The calculated peak frequencies are $\omega_{\max}=0.1492$ and $\omega_{\max}=0.0726$ rads/sample, corresponding to periods of $2\pi/\omega_{\max}=42$ and 86 samples/cycle. The peak points are indicated by black dots on the graph.

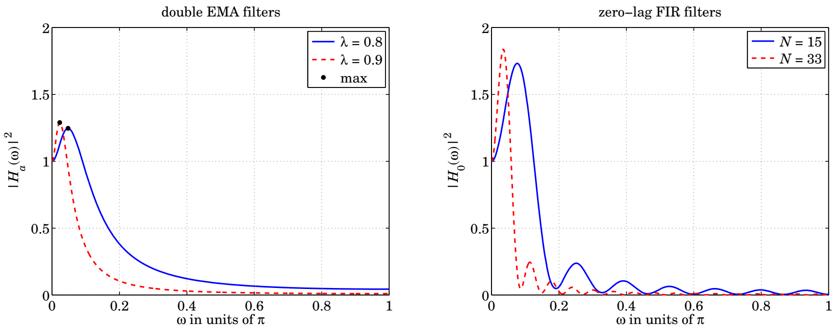


Fig. 6.10.1 Double EMA and zero-lag FIR filter responses.

The right graph shows the magnitude responses of the optimum zero-lag FIR filter $h_a(k)$ of Eq. (6.10.6) for the two lengths $N=15$ and $N=33$. The lengths N were calculated to achieve equivalent behavior in the vicinity of DC, i.e., equal second derivatives at $\omega=0$,

$$\frac{4\lambda^2}{(1-\lambda)^2} = \frac{1}{3}(N-1)(N-2) \quad\Rightarrow\quad N = \frac{3}{2} + \sqrt{\frac{12\lambda^2}{(1-\lambda)^2} + \frac{1}{4}}$$

The magnitude responses were computed from the above formulas for the double EMA cases, and by the following MATLAB code in the FIR cases:

```
w = linspace(0,1,1001);              % ω in units of π
N = 15; k = (0:N-1);
h = 2*(2*N-1-3*k)/N/(N+1);
H2 = abs(freqz(h,1,pi*w)).^2;        % magnitude response squared
```

We observe from these examples that the zero-lag filters have less than desirable passbands. However, they do act as lowpass filters, attenuating the high frequencies and thereby smoothing the input signal. □

Local Level, Local Slope, and Local Acceleration Filters

Since the twicing operation can be applied to any lowpass filter $H(z)$ resulting in the zero-lag local-level filter, $H_a(z)=2H(z)-H^2(z)$, it raises the question as to what would be the corresponding local-slope filter $H_b(z)$, in other words, what is the generalization of Eqs. (6.8.6) for an arbitrary filter $H(z)$, and similarly, what is the generalization of Eq. (6.8.12) for the thricing operations.

Starting with an arbitrary causal lowpass filter $H(z)$, with impulse response $h(n)$, and assuming that it has unity gain at DC, the local level, slope, and acceleration filters depend on the following two filter moments:

$$\mu_1 = \bar{n} = \sum_{n=0}^{\infty} nh(n)\,, \quad \mu_2 = \sum_{n=0}^{\infty} n^2 h(n) \tag{6.10.8}$$

In terms of these parameters, the generalization of Eq. (6.8.6) is then,

$$\boxed{\begin{aligned} H_a(z) &= 2H(z) - H^2(z) \\ H_b(z) &= \frac{1}{\mu_1}\big[H(z) - H^2(z)\big] \end{aligned}} \quad \begin{aligned} &\text{(local level filter)} \\ &\text{(local slope filter)} \end{aligned} \tag{6.10.9}$$

while the generalization of (6.8.12) is,

$$\begin{bmatrix} H_a(z) \\ H_b(z) \\ H_c(z) \end{bmatrix} = \frac{1}{2\mu_1^3}\begin{bmatrix} 6\mu_1^3 & -6\mu_1^3 & 2\mu_1^3 \\ \mu_2 + 4\mu_1^2 & -2(\mu_2 + 3\mu_1^2) & \mu_2 + 2\mu_1^2 \\ \mu_1 & -2\mu_1 & \mu_1 \end{bmatrix}\begin{bmatrix} H(z) \\ H^2(z) \\ H^3(z) \end{bmatrix} \tag{6.10.10}$$

and in particular,

$$H_a = 3H - 3H^2 + H^3 = 1 - (1-H)^3 \tag{6.10.11}$$

For an EMA filter, $h(n) = (1-\lambda)\lambda^n u(n)$, we have,

$$\mu_1 = \frac{\lambda}{1-\lambda}\,, \quad \mu_2 = \frac{\lambda(1+\lambda)}{(1-\lambda)^2}$$

and Eqs. (6.10.9) and (6.10.10) reduce to (6.8.6) and (6.8.12), respectively. To justify (6.10.9), consider a noiseless straight-line input signal, $y_n = a + bn$. Since it is linearly rising and noiseless, the local-level will be itself, and the local-slope will be constant, that is, we may define,

$$a_n \equiv a + bn\,, \quad b_n \equiv b$$

Following the calculation leading to Eq. (6.1.24), we can easily verify that the two successive outputs, $a_n^{[1]}, a_n^{[2]}$, from the twice-cascaded filter $h(n)$, will be,

$$\begin{aligned} a_n^{[1]} &= a + b(n - \mu_1) = (a - \mu_1 b) + bn = a + bn - \mu_1 b = a_n - \mu_1 b_n \\ a_n^{[2]} &= (a - \mu_1 b - \mu_1 b) + bn = (a - 2\mu_1 b) + bn = a_n - 2\mu_1 b_n \end{aligned}$$

These may be solved for a_n, b_n in terms of $a_n^{[1]}, a_n^{[2]}$, leading to the following time-domain relationships, which are equivalent to Eqs. (6.10.9),

$$\begin{aligned} a_n &= 2a_n^{[1]} - a_n^{[2]} \\ b_n &= \frac{1}{\mu_1}\big(a_n^{[1]} - a_n^{[2]}\big) \end{aligned}$$

For the thricing case, consider a noiseless input that is quadratically varying with time, $y_n = a + bn + cn^2$, so that its local level, local slope, and local acceleration may be defined as,†

$$a_n \equiv a + bn + cn^2\,, \quad b_n \equiv b + 2cn\,, \quad c_n \equiv c$$

† true acceleration would be represented by $2c$.

Upon passing through the first stage of $h(n)$, the output will be,

$$\begin{aligned} a_n^{[1]} &= \sum_k \big[a + b(n-k) + c(n-k)^2\big]h(k) \\ &= \sum_k \big[a + b(n-k) + c(n^2 - 2nk + k^2)\big]h(k) \\ &= a + b(n - \mu_1) + c(n^2 - 2n\mu_1 + \mu_2) \\ &= (a - b\mu_1 + c\mu_2) + (b - 2c\mu_1)n + cn^2 \end{aligned}$$

and applying the same formula to the second and third stages of $h(n)$, we find the outputs,

$$\begin{aligned} a_n^{[1]} &= (a - b\mu_1 + c\mu_2) + (b - 2c\mu_1)n + cn^2 \\ a_n^{[2]} &= (a - 2b\mu_1 + 2c\mu_2 + 2c\mu_1^2) + (b - 4c\mu_1)n + cn^2 \\ a_n^{[3]} &= (a - 3b\mu_1 + 3c\mu_2 + 6c\mu_1^2) + (b - 6c\mu_1)n + cn^2 \end{aligned}$$

which can be re-expressed in terms of the a_n, b_n, c_n signals,

$$\begin{aligned} a_n^{[1]} &= a_n - \mu_1 b_n + \mu_2 c_n \\ a_n^{[2]} &= a_n - 2\mu_1 b_n + 2(\mu_2 + \mu_1^2)c_n \\ a_n^{[3]} &= a_n - 3\mu_1 b_n + 3(\mu_2 + 2\mu_1^2)c_n \end{aligned}$$

Solving these for a_n, b_n, c_n leads to the time-domain equivalent of Eq. (6.10.10),

$$\begin{bmatrix} a_n \\ b_n \\ c_n \end{bmatrix} = \frac{1}{2\mu_1^3}\begin{bmatrix} 6\mu_1^3 & -6\mu_1^3 & 2\mu_1^3 \\ \mu_2 + 4\mu_1^2 & -2(\mu_2 + 3\mu_1^2) & \mu_2 + 2\mu_1^2 \\ \mu_1 & -2\mu_1 & \mu_1 \end{bmatrix}\begin{bmatrix} a_n^{[1]} \\ a_n^{[2]} \\ a_n^{[3]} \end{bmatrix}$$

and in particular,

$$a_n = 3a_n^{[1]} - 3a_n^{[2]} + a_n^{[3]}$$

6.11 Basis Transformations and EMA Initialization

The transformation matrix between the $\mathbf{c}(n) = [c_0(n), c_1(n), \dots, c_d(n)]^T$ basis and the cascaded basis $\mathbf{a}(n) = [a_n^{[1]}, a_n^{[2]}, \dots, a_n^{[d+1]}]^T$ can be written in the general form:

$$\mathbf{a}(n) = M\mathbf{c}(n) \quad \Rightarrow \quad a_n^{[r]} = \sum_{i=0}^{d} M_{ri}c_i(n)\,, \quad r = 1, 2, \dots, d+1 \tag{6.11.1}$$

The matrix elements M_{ri} can be found by looking at the special case when the input is a polynomial of degree d,

$$x_{n+\tau} = \sum_{i=0}^{d} \tau^i c_i(n)$$

The convolutional output of the filter $H^{[r]}(z)$ is then,

$$a_n^{[r]} = \sum_{k=0}^{\infty} h^{[r]}(k)x_{n-k} = \sum_{k=0}^{\infty} h^{[r]}(k)\sum_{i=0}^{d}(-k)^i c_i(n)$$

It follows that,

$$M_{ri} = \sum_{k=0}^{\infty} h^{[r]}(k)(-k)^i = \sum_{k=0}^{\infty} \alpha^r \lambda^k \frac{(k+r-1)!}{k!(r-1)!}(-k)^i \tag{6.11.2}$$

with $1 \le r \le d+1$ and $0 \le i \le d$. The matrices for $d = 1$ and $d = 2$ in Eqs. (6.8.8) and (6.8.11) are special cases of (6.11.2). The function emat calculates M numerically,

```
M = emat(d,lambda);        % polynomial to cascaded basis transformation matrix
```

One of the uses of this matrix is to determine the initial condition vector in the $a_n^{[r]}$ basis, $\mathbf{a}_{\text{init}} = M\mathbf{c}_{\text{init}}$, where $\mathbf{c}_{\text{init}}$ is more easily determined, for example, using the default method of the function stema.

The function mema implements the multiple exponential moving average in cascade form, generating the individual outputs $a_n^{[r]}$:

```
[a,A] = mema(y,d,la,ainit);        % multiple exponential moving average
```

where A is an $N\times(d+1)$ matrix whose nth row is $\mathbf{a}(n)^T = [a_n^{[1]}, a_n^{[2]}, \dots, a_n^{[d+1]}]$, and a is the a_n output of Eq. (6.8.17). The $(d+1)\times 1$ vector ainit represents the initial values of $\mathbf{a}(n)$, that is, $\mathbf{a}_{\text{init}} = [a_{\text{init}}^{[1]}, a_{\text{init}}^{[2]}, \dots, a_{\text{init}}^{[d+1]}]^T$. If the argument ainit is omitted, it defaults to the following least-squares fitting procedure:

```
L = round((1+la)/(1-la));              % effective length of single EMA
cinit = lpbasis(L,d,-1)\y(1:L);        % fit order-d polynomial to first L inputs
M = emat(d,la);                        % transformation matrix to cascaded basis
ainit = M*cinit;                       % (d+1)×1 initial vector
```

The function mema calculates the filter outputs using the built-in function filter to implement filtering by the single EMA filter $H(z) = \alpha/(1-\lambda z^{-1})$ and passing each output to the next stage, for example, with $a_n^{[0]} = y_n$,

$$a^{[r]} = \texttt{filter}(\alpha, [1,-\lambda], a^{[r-1]}, \lambda a_{\text{init}}^{[r]}), \quad r = 1,2,\dots,d+1 \tag{6.11.3}$$

The last argument of filter imposes the proper initial state on the filtering operation (the particular form is dictated from the fact that filter uses the transposed realization.) Eq. (6.11.3) is equivalent to the operations:

$$a_n^{[r]} = \lambda a_{n-1}^{[r]} + \alpha a_n^{[r-1]}, \quad r = 1,2,\dots,d+1 \tag{6.11.4}$$

Example 6.11.1: *EMA Initialization.* To clarify these operations and the initializations, we consider a small example using $d = 1$ (double EMA) and $\lambda = 0.9$, $\alpha = 1-\lambda = 0.1$. The data vector $\mathbf{y}$ has length 41 and is given in the code segment below.

The top two graphs in Fig. 6.11.1 show the default initialization method in which a linear fit (because $d = 1$) is performed to the first $L = (1+\lambda)/(1-\lambda) = 19$ data samples. In the bottom two graphs, the initialization is based on performing the linear fit to just the first $L = 5$ samples.

In all cases, the linearly-fitted segments are shown on the graphs (short dashed lines). In the left graphs, the initialization parameters $\mathbf{c}_{\text{init}}, \mathbf{a}_{\text{init}}$ were determined at time $n = -1$ and processing began at $n = 0$. In the right graphs, the $\mathbf{c}_{\text{init}}, \mathbf{a}_{\text{init}}$ were recalculated to correspond to time $n = L-1$ (i.e., $n = 18$ and $n = 4$ for the top and bottom graphs), and processing was started at $n = L$. The table below displays the computations for the left and right bottom graphs.

For both the left and right tables, the same five data samples $\{y_n, 0 \le n \le 4\}$ were used to determine the initialization vectors $\mathbf{c}_{\text{init}}$, which were then mapped into $\mathbf{a}_{\text{init}} = M\mathbf{c}_{\text{init}}$. The transformation matrix M is in this example (cf. Eq. (6.8.8)):

$$M = \begin{bmatrix} 1 & -\lambda/\alpha \\ 1 & -2\lambda/\alpha \end{bmatrix} = \begin{bmatrix} 1 & -9 \\ 1 & -18 \end{bmatrix}$$

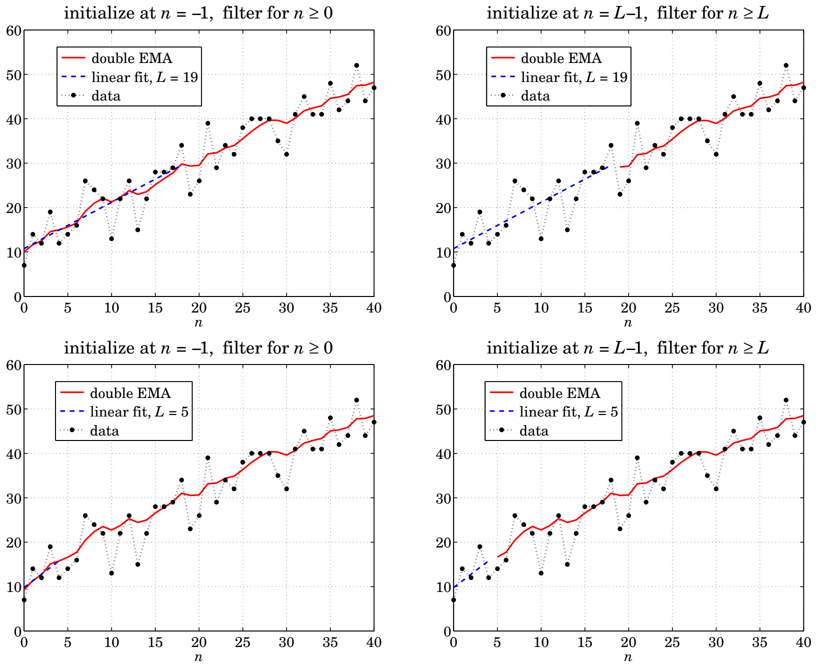


Fig. 6.11.1 Double-EMA initialization examples.

n	y_n	$a_n^{[1]}$	$a_n^{[2]}$
−1		−5.2000	−18.7000
0	7	−3.9800	−17.2280
1	14	−2.1820	−15.7234
2	12	−0.7638	−14.2274
3	19	1.2126	−12.6834
4	12	2.2913	−11.1860
5	14	3.4622	−9.7211
6	16	4.7160	−8.2774
⋮	⋮	⋮	⋮
39	44	39.2235	30.5592
40	47	40.0011	31.5034

n	y_n	$a_n^{[1]}$	$a_n^{[2]}$
−1			
0	7		
1	14		
2	12		
3	19		
4	12	2.3000	−11.2000
5	14	3.4700	−9.7330
6	16	4.7230	−8.2874
⋮	⋮	⋮	⋮
39	44	39.2237	30.5596
40	47	40.0013	31.5037

For the left table, the data fitting relative to $n = -1$ gives:

$$\mathbf{c}_{\text{init}} = \begin{bmatrix} 8.3 \\ 1.5 \end{bmatrix} \quad\Rightarrow\quad \mathbf{a}_{\text{init}} = M\mathbf{c}_{\text{init}} = \begin{bmatrix} 1 & -9 \\ 1 & -18 \end{bmatrix}\begin{bmatrix} 8.3 \\ 1.5 \end{bmatrix} = \begin{bmatrix} -5.2 \\ -18.7 \end{bmatrix}$$

obtained from `cinit = S\y(1:L)`, indeed, with $S = \text{lpbasis}(L, d, -1)$, we find

$$S = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \\ 1 & 5 \end{bmatrix} \;\Rightarrow\; \mathbf{c}_{\text{init}} = (S^T S)^{-1} S^T \mathbf{y}_{1:L} = \begin{bmatrix} 0.8 & 0.5 & 0.2 & -0.1 & -0.4 \\ -0.2 & -0.1 & 0.0 & 0.1 & 0.2 \end{bmatrix}\begin{bmatrix} 7 \\ 14 \\ 12 \\ 19 \\ 12 \end{bmatrix} = \begin{bmatrix} 8.3 \\ 1.5 \end{bmatrix}$$

These initial values are shown at the top of the left table. The rest of the table entries are computed by cranking the difference equations for $n \ge 0$,

$$a_n^{[1]} = \lambda a_{n-1}^{[1]} + \alpha y_n$$
$$a_n^{[2]} = \lambda a_{n-1}^{[2]} + \alpha a_n^{[1]}$$

for example,

$$a_0^{[1]} = \lambda a_{-1}^{[1]} + \alpha y_0 = (0.9)(-5.2) + (0.1)(7) = -3.980$$
$$a_0^{[2]} = \lambda a_{-1}^{[2]} + \alpha a_0^{[1]} = (0.9)(-18.7) + (0.1)(-3.98) = -17.228$$

For the right table, the initialization coefficients relative to $n = L-1 = 4$ may be determined by boosting those for $n = -1$ by $L = 5$ time units:

$$\bar{\mathbf{c}}_{\text{init}} = (F^T)^L \mathbf{c}_{\text{init}} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}^5 \begin{bmatrix} 8.3 \\ 1.5 \end{bmatrix} = \begin{bmatrix} 1 & 5 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 8.3 \\ 1.5 \end{bmatrix} = \begin{bmatrix} 15.8 \\ 1.5 \end{bmatrix}$$

$$\bar{\mathbf{a}}_{\text{init}} = M\bar{\mathbf{c}}_{\text{init}} = \begin{bmatrix} 1 & -9 \\ 1 & -18 \end{bmatrix}\begin{bmatrix} 15.8 \\ 1.5 \end{bmatrix} = \begin{bmatrix} 2.3 \\ -11.2 \end{bmatrix}$$

Alternatively, $\bar{\mathbf{c}}_{\text{init}}$ can be computed from `cinit = lpbasis(L,d,L-1)\y(1:L)`, i.e.,

$$S = \begin{bmatrix} 1 & -4 \\ 1 & -3 \\ 1 & -2 \\ 1 & -1 \\ 1 & 0 \end{bmatrix} \;\Rightarrow\; \bar{\mathbf{c}}_{\text{init}} = (S^T S)^{-1} S^T \mathbf{y}_{1:L} = \begin{bmatrix} -0.2 & 0.0 & 0.2 & 0.4 & 0.6 \\ -0.2 & -0.1 & 0.0 & 0.1 & 0.2 \end{bmatrix}\begin{bmatrix} 7 \\ 14 \\ 12 \\ 19 \\ 12 \end{bmatrix} = \begin{bmatrix} 15.8 \\ 1.5 \end{bmatrix}$$

The $\bar{\mathbf{a}}_{\text{init}}$ values are shown on the right table at the $n = 4$ line. The rest of the table is computed by cranking the above difference equations starting at $n = 5$. For example,

$$a_5^{[1]} = \lambda a_4^{[1]} + \alpha y_5 = (0.9)(2.3) + (0.1)(14) = 3.47$$
$$a_5^{[2]} = \lambda a_4^{[2]} + \alpha a_5^{[1]} = (0.9)(-11.2) + (0.1)(3.47) = -9.733$$

We note that the filtered values at $n = L - 1 = 4$ on the left table and the initial values on the right table are very close to each other, and therefore, the two initialization methods produce very comparable results for the output segments $n \ge L$. The following MATLAB code illustrates the generation of the bottom graphs in Fig. 6.11.1:

```
y = [ 7 14 12 19 12 14 16 26 24 22 13 22 26 15 22 28 28 29 34 23 26 ...
     39 29 34 32 38 40 40 40 35 32 41 45 41 41 48 42 44 52 44 47 ]';
n = (0:length(y)-1)';

d=1; F=binmat(d,1); L=5;                        % F = boost matrix - not needed
la = 0.9; al = 1-la;
% L = round((1+la)/(1-la));                     % use this L for the top two graphs

cinit = lpbasis(L,d,-1)\y(1:L);                 % fit relative to n = −1
M = emat(d,la);                                 % transformation matrix
ainit = M*cinit;                                % initial values for cascade realization

C = stema(y,d,la,cinit);                        % needed for testing purposes only
[a,A] = mema(y,d,la,ainit);                     % filter for n ≥ 0

N1 = norm(A-C*M') + norm(a-C(:,1));             % compare stema and mema outputs

t = (0:L-1)'; p = lpbasis(L,d,-1)*cinit;        % initial L fitted values

figure; plot(n,y,'.', n,a,'-', t,p,'--', n,y,':');   % bottom left graph

cinit = lpbasis(L,d,L-1)\y(1:L);                % fit relative to n = L − 1
                                                % or, multiply previous cinit by (F')^L
ainit = M*cinit;                                % initial values for cascade realization

nL = n(L+1:end); yL = y(L+1:end);               % restrict input to n ≥ L

C = stema(yL,d,la,cinit);                       % needed for testing purposes only
[a,A] = mema(yL,d,la,ainit);                    % filter for n ≥ L

N2 = norm(A-C*M') + norm(a-C(:,1));             % compare stema and mema outputs

t = (0:L-1)'; p = lpbasis(L,d,L-1)*cinit;       % initial L fitted values

figure; plot(n,y,'.', nL,a,'-', t,p,'--', n,y,':');  % bottom right graph

Ntot = N1 + N2                                  % overall test – should be zero
```

The first initialization scheme (finding $\mathbf{c}_{\text{init}}, \mathbf{a}_{\text{init}}$ at $n = -1$ and starting filtering at $n = 0$) is generally preferable because it produces an output of the same length as the input. □

An alternative initialization scheme that is more common in financial market trading applications of EMA is discussed in Sec. 6.17.

6.12 Holt's Exponential Smoothing

We recall that the $d = 1$ steady-state EMA was given by

$$\begin{bmatrix} a_n \\ b_n \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} a_{n-1} \\ b_{n-1} \end{bmatrix} + \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} (y_n - a_{n-1} - b_{n-1}) \tag{6.12.1}$$

with asymptotic gain factors $\alpha_1 = 1-\lambda^2$ and $\alpha_2 = (1-\lambda)^2$, which are both computable from a single λ parameter.

Holt [240] has generalized (6.12.1) to allow arbitrary values for α_1, α_2. The additional flexibility has been found to be effective in applications. There is an alternative way of writing (6.12.1). From the first equation, we have

$$a_n = a_{n-1}+b_{n-1}+\alpha_1(y_n-a_{n-1}-b_{n-1}) \quad\Rightarrow\quad y_n-a_{n-1}-b_{n-1} = \frac{1}{\alpha_1}(a_n-a_{n-1}-b_{n-1})$$

and substituting into the second,

$$b_n = b_{n-1} + \alpha_2(y_n - a_{n-1} - b_{n-1}) = b_{n-1} + \frac{\alpha_2}{\alpha_1}(a_n - a_{n-1} - b_{n-1})$$

Defining $\bar{\alpha}_2 = \alpha_2/\alpha_1$, we may write the new system as follows:

$$\begin{aligned} a_n &= a_{n-1} + b_{n-1} + \alpha_1(y_n - a_{n-1} - b_{n-1}) \\ b_n &= b_{n-1} + \bar{\alpha}_2(a_n - a_{n-1} - b_{n-1}) \end{aligned} \tag{6.12.2}$$

and defining the effective λ-parameters $\lambda_1 = 1 - \alpha_1$ and $\bar{\lambda}_2 = 1 - \bar{\alpha}_2$,

$$\boxed{\begin{aligned} a_n &= \lambda_1(a_{n-1} + b_{n-1}) + \alpha_1 y_n \\ b_n &= \bar{\lambda}_2 b_{n-1} + \bar{\alpha}_2(a_n - a_{n-1}) \end{aligned}} \quad \text{(Holt's exponential smoothing)} \tag{6.12.3}$$

Eq. (6.12.1) is referred to a exponential smoothing with "local trend". The first equation tracks the local level a_n, and the second, the local slope b_n, which is being determined by smoothing the first-order difference of the local level $a_n - a_{n-1}$.

The predicted value is as usual $\hat{y}_{n/n-1} = a_{n-1} + b_{n-1}$, and for the next time instant, $\hat{y}_{n+1/n} = a_n + b_n$, and τ steps ahead, $\hat{y}_{n+\tau/n} = a_n + b_n\tau$. The MATLAB function `holt` implements Eq. (6.12.1):

```
C = holt(y,a1,a2,cinit);        % Holt's exponential smoothing
```

where C has the same meaning as `stema`, its nth row $\mathbf{c}^T(n) = [a_n, b_n]$ holding the local level and slope at time n. The initialization vector `cinit` can be chosen as in `stema` by a linear fit to the first L samples of $\mathbf{y}$, where $L = (1+\lambda)/(1-\lambda)$, with λ determined from α_1 from the relationship $\alpha_1 = 1 - \lambda^2$ or $\lambda = \sqrt{1-\alpha_1}$. Another possibility is to choose $\mathbf{c}_{\text{init}} = [y_0, 0]^T$, or, $[y_0, y_1 - y_0]^T$.

Like `emaerr`, the MATLAB function **holterr** evaluates the MSE/MAE/MAPE errors over any set of parameter pairs (α_1, α_2) and produces the corresponding optimum pair $(\alpha_{1,\text{opt}}, \alpha_{2,\text{opt}})$:

```
[err,a1opt,a2opt] = holterr(y,a1,a2,type,cinit);        % mean error measures
```

By taking z-transforms of Eq. (6.12.1), we obtain the transfer functions from y_n to the two outputs a_n, b_n:

$$\begin{aligned} H_a(z) &= \frac{\alpha_1 + (\alpha_2 - \alpha_1)z^{-1}}{1 + (\alpha_1 + \alpha_2 - 2)z^{-1} + (1 - \alpha_1)z^{-2}} \\ H_b(z) &= \frac{\alpha_2(1 - z^{-1})}{1 + (\alpha_1 + \alpha_2 - 2)z^{-1} + (1 - \alpha_1)z^{-2}} \end{aligned} \tag{6.12.4}$$

The transfer function from y_n to the predicted output $\hat{y}_{n+1/n}$ is $H(z) = H_a(z) + H_b(z)$. Making the usual assumption that y_n is a white noise sequence, the variance of $\hat{y}_{n+1/n}$ will be $\sigma_{\hat{y}}^2 = \mathcal{R}\sigma_y^2$, where $\mathcal{R}$ is the NRR of $H(z)$:

$$\mathcal{R} = \sum_{n=0}^{\infty} h^2(n) = \frac{2\alpha_1^2 + \alpha_1\alpha_2 + 2\alpha_2}{\alpha_1(4 - 2\alpha_1 - \alpha_2)} \tag{6.12.5}$$

This reduces to Eq. (6.7.9) in the EMA case of $\alpha_1 = 1 - \lambda^2$ and $\alpha_2 = (1-\lambda)^2$, while Eq. (6.12.4) reduces to (6.8.5). It can be shown that $\mathcal{R}$ remains less than unity for $0 \le \alpha_1 \le 1$ and $0 \le \alpha_2 \le 2\alpha_1(1-\alpha_1)/(1+\alpha_1)$, with R reaching unity at $\alpha_1 = \sqrt{2} - 1 = 0.4142$ and $\alpha_2 = 2\alpha_1(1-\alpha_1)/(1+\alpha_1) = 2(3 - 2\sqrt{2}) = 0.3431$.

6.13 State-Space Models for Holt's Method

Just like the $d = 0$ local level case could be represented by an innovations model, so can the linear trend model. We may define the model by assuming that the prediction is perfect and thus, the prediction error $e_{n/n-1} = y_n - \hat{y}_{n/n-1} \equiv \varepsilon_n$ is a white noise signal. The state vector may be defined as $\mathbf{x}_n = [a_n, b_n]^T$, leading to the innovations state-space model,

$$\begin{aligned} y_n &= [1, 1]\mathbf{x}_{n-1} + \varepsilon_n \\ \mathbf{x}_n &= \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \mathbf{x}_{n-1} + \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} \varepsilon_n \end{aligned} \tag{6.13.1}$$

Eliminating the state vector, we may obtain the transfer function from the innovations sequence ε_n to the observation y_n,

$$\frac{Y(z)}{\mathcal{E}(z)} = \frac{1 + (\alpha_1 + \alpha_2 - 2)z^{-1} + (1 - \alpha_1)z^{-2}}{(1 - z^{-1})^2}$$

which leads to an ARIMA(0,2,2) or IMA(2,2) equivalent signal model:

$$\nabla^2 y_n = y_n - 2y_{n-2} + y_{n-2} = \varepsilon_n + (\alpha_1 + \alpha_2 - 2)\varepsilon_{n-1} + (1-\alpha_1)\varepsilon_{n-2} \qquad (6.13.2)$$

For $\alpha_1 = 1 - \lambda^2$ and $\alpha_2 = (1-\lambda)^2$, we obtain the special case [253],

$$\frac{Y(z)}{\mathcal{E}(z)} = \frac{(1-\lambda z^{-1})^2}{(1-z^{-1})^2}\,, \quad \nabla^2 y_n = \varepsilon_n - 2\lambda\varepsilon_{n-1} + \lambda^2\varepsilon_{n-2} \qquad (6.13.3)$$

The following more conventional formulation of the linear trend state-space model has steady-state Kalman filtering equations that are equivalent to Holt's method:

$$\begin{bmatrix} a_{n+1} \\ b_{n+1} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} a_n \\ b_n \end{bmatrix} + \begin{bmatrix} w_n \\ u_n \end{bmatrix}, \quad y_n = [1,0]\begin{bmatrix} a_n \\ b_n \end{bmatrix} + v_n \qquad (6.13.4)$$

where a_n, b_n represent the local level and local slope, respectively, and w_n, u_n, v_n are zero-mean, mutually uncorrelated, white-noise signals of variances $\sigma_w^2, \sigma_u^2, \sigma_v^2$. Denoting the state vector and its filtered and predicted estimates by,

$$\mathbf{x}_n = \begin{bmatrix} a_n \\ b_n \end{bmatrix}, \quad \hat{\mathbf{x}}_{n/n} = \begin{bmatrix} \hat{a}_n \\ \hat{b}_n \end{bmatrix}, \quad \hat{\mathbf{x}}_{n+1/n} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} \hat{a}_n \\ \hat{b}_n \end{bmatrix}$$

it turns out that the steady-state Kalman filtering equations take exactly the innovations form of Eq. (6.13.1):

$$\varepsilon_n = y_n - (\hat{a}_{n-1} + \hat{b}_{n-1})\,, \quad \begin{bmatrix} \hat{a}_n \\ \hat{b}_n \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} \hat{a}_{n-1} \\ \hat{b}_{n-1} \end{bmatrix} + \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix}\varepsilon_n \qquad (6.13.5)$$

where α_1, α_2 are related to the noise variances by:

$$\frac{\sigma_w^2}{\sigma_v^2} = \frac{\alpha_1^2 + \alpha_1\alpha_2 - 2\alpha_2}{1-\alpha_1}\,, \quad \frac{\sigma_u^2}{\sigma_v^2} = \frac{\alpha_2^2}{1-\alpha_1} \qquad (6.13.6)$$

State-space models provide a modern way of thinking about exponential smoothing and will be explored further in Chap. 13.

There is an extensive literature on exponential smoothing, a small subset of which is [232–279]. There are many other variants (no less than 15), such as multiplicative, seasonal, adaptive versions. A recent review of all cases that emphasizes the state-space point of view is found in [239].

We finish by mentioning the Holt-Winters generalization [241] of Holt's method to seasonal data. In addition to tracking the level and slope signals a_n, b_n the method also tracks the local seasonal component, say s_n. For the additive version, we have:

$$\boxed{\begin{aligned} a_n &= \lambda_1(a_{n-1} + b_{n-1}) + \alpha_1(y_n - s_{n-D}) \\ b_n &= \bar{\lambda}_2 b_{n-1} + \bar{\alpha}_2(a_n - a_{n-1}) \\ s_n &= \lambda_3 s_{n-D} + \alpha_3(y_n - a_{n-1} - b_{n-1}) \end{aligned}} \quad \text{(Holt-Winters)} \qquad (6.13.7)$$

where D is the assumed periodicity of the seasonal data, and α_3 and $\lambda_3 = 1 - \alpha_3$ are the smoothing parameters associated with the seasonal component. The predicted estimate is obtained by $\hat{y}_{n+1/n} = a_n + b_n + s_{n-D}$.

6.14 Filtering Methods in Financial Market Trading

Technical analysis of financial markets refers to a family of signal processing methods and indicators used by stock market traders to make sense of the constantly fluctuating market data and arrive at successful "buy" or "sell" decisions.

Both linear and nonlinear filtering methods are used. A comprehensive reference on such methods is the Achelis book [280]. Some additional references are [281–347].

Here, we look briefly at some widely used indicators arising from FIR or EMA filters, and summarize their properties and their MATLAB implementation. In order to keep the discussion self-contained, some material from the previous sections is repeated.

6.15 Moving Average Filters – SMA, WMA, TMA, EMA

Among the linear filtering methods are smoothing filters that are used to smooth out the daily fluctuations and bring out the trends in the data. The most common filters are the *simple moving average* (SMA) and the *exponentially weighted moving average* (EMA), and variations, such as the *weighted or linear moving average* (WMA) and the *triangular moving average* (TMA). The impulse responses of these filters are:

$$\begin{aligned} &\text{(SMA)} \quad h(n) = \frac{1}{N}\,, \quad 0 \le n \le N-1 \\ &\text{(WMA)} \quad h(n) = \frac{2(N-n)}{N(N+1)}\,, \quad 0 \le n \le N-1 \\ &\text{(TMA)} \quad h(n) = \frac{N - |n-N+1|}{N^2}\,, \quad 0 \le n \le 2N-2 \\ &\text{(EMA)} \quad h(n) = (1-\lambda)\lambda^n\,, \quad 0 \le n < \infty \end{aligned} \qquad (6.15.1)$$

with transfer functions,

$$\begin{aligned} &\text{(SMA)} \quad H(z) = \frac{1 + z^{-1} + z^{-2} + \cdots + z^{-N+1}}{N} = \frac{1}{N}\,\frac{1-z^{-N}}{1-z^{-1}} \\ &\text{(WMA)} \quad H(z) = \frac{2}{N(N+1)}\,\frac{N - (N+1)z^{-1} + z^{-N-1}}{(1-z^{-1})^2} \\ &\text{(TMA)} \quad H(z) = \left[\frac{1}{N}\,\frac{1-z^{-N}}{1-z^{-1}}\right]^2 \\ &\text{(EMA)} \quad H(z) = \frac{\alpha}{1-\lambda z^{-1}}\,, \quad \alpha = 1-\lambda \end{aligned} \qquad (6.15.2)$$

where N denotes the filter span for the SMA and WMA cases, while for the EMA case, λ is a forgetting factor such that $0 < \lambda < 1$, which is usually specified in terms an equivalent FIR length N given by the following condition, which implies that the SMA and the EMA filters have the same lag and the same noise reduction ratio, as discussed in Sec. 6.1,

$$N = \frac{1+\lambda}{1-\lambda} \quad \Rightarrow \quad \lambda = \frac{N-1}{N+1} \quad \Rightarrow \quad \alpha = 1 - \lambda = \frac{2}{N+1} \qquad (6.15.3)$$

The TMA filter has length $2N-1$ and is evidently the convolution of two length-N SMAs. Denoting by y_n the raw data, where n represents the nth trading day (or, weekly, monthly, or quarterly sample periods), we will denote the output of the moving average filters by a_n representing the smoothed *local level* of y_n. The corresponding I/O filtering equations are then,

$$
\begin{aligned}
&\text{(SMA)} \qquad a_n = \frac{y_n + y_{n-1} + y_{n-2} + \cdots + y_{n-N+1}}{N}\\
&\text{(WMA)} \qquad a_n = \frac{2}{N(N+1)}\sum_{k=0}^{N-1}(N-k)\,y_{n-k}\\
&\text{(TMA)} \qquad a_n = \frac{1}{N^2}\sum_{k=0}^{2N-2}\big(N-|k-N+1|\big)\,y_{n-k}\\
&\text{(EMA)} \qquad a_n = \lambda\,a_{n-1} + (1-\lambda)\,y_n
\end{aligned}
\tag{6.15.4}
$$

The typical *trading rule* used by traders is to "buy" when a_n is rising and y_n lies above a_n, and to "sell" when a_n is falling and y_n lies below a_n.

Unfortunately, these widely used filters have an inherent lag, which can often result in false buy/sell signals. The basic tradeoff is that longer lengths N result in longer lags, but at the same time, the filters become more effective in smoothing and reducing noise in the data. The noise-reduction capability of any filter is quantified by its "noise-reduction ratio" defined by,

$$\mathcal{R} = \sum_{n=0}^{\infty} h^2(n) \tag{6.15.5}$$

with smaller $\mathcal{R}$ corresponding to more effective noise reduction. By construction, the above filters are lowpass filters with unity gain at DC, therefore, satisfying the constraint,

$$\sum_{n=0}^{\infty} h(n) = 1 \tag{6.15.6}$$

The "lag" is defined as the group delay at DC which, after using Eq. (6.15.6), is given by,

$$\bar{n} = \sum_{n=0}^{\infty} n\,h(n) \tag{6.15.7}$$

One can easily verify that the noise-reduction ratios and lags of the above filters are:

$$
\begin{aligned}
&\text{(SMA)} \qquad \mathcal{R} = \frac{1}{N}, && \bar{n} = \frac{N-1}{2}\\
&\text{(WMA)} \qquad \mathcal{R} = \frac{4N+2}{3N(N+1)}, && \bar{n} = \frac{N-1}{3}\\
&\text{(TMA)} \qquad \mathcal{R} = \frac{2N^2+1}{3N^3}, && \bar{n} = N-1\\
&\text{(EMA)} \qquad \mathcal{R} = \frac{1-\lambda}{1+\lambda} = \frac{1}{N}, && \bar{n} = \frac{\lambda}{1-\lambda} = \frac{N-1}{2}, \quad \text{for equivalent } N
\end{aligned}
\tag{6.15.8}
$$

The tradeoff is evident, with $\mathcal{R}$ decreasing and $\bar{n}$ increasing with N.

We include one more lowpass smoothing filter, the *integrated linear regression slope* (ILRS) filter [307] which is developed in Sec. 6.16. It has unity DC gain and its impulse response, transfer function, lag, and NRR are given by,

$$
\begin{aligned}
&\text{(IRLS)} \qquad h(n) = \frac{6(n+1)(N-1-n)}{N(N^2-1)}, \qquad n = 0,1,\dots,N-1\\
&\qquad\qquad H(z) = \frac{6}{N(N^2-1)}\,\frac{N(1-z^{-1})(1+z^{-N}) - (1-z^{-N})(1+z^{-1})}{(1-z^{-1})^3}\\
&\qquad\qquad \bar{n} = \frac{N-2}{2}, \quad \mathcal{R} = \frac{6(N^2+1)}{5N(N^2-1)}
\end{aligned}
\tag{6.15.9}
$$

Fig. 6.15.1 compares the frequency responses of the above filters. We note that the ILRS has similar bandwidth as the WMA, but it also has a smaller NRR and more suppressed high-frequency range, thus, resulting in smoother output.

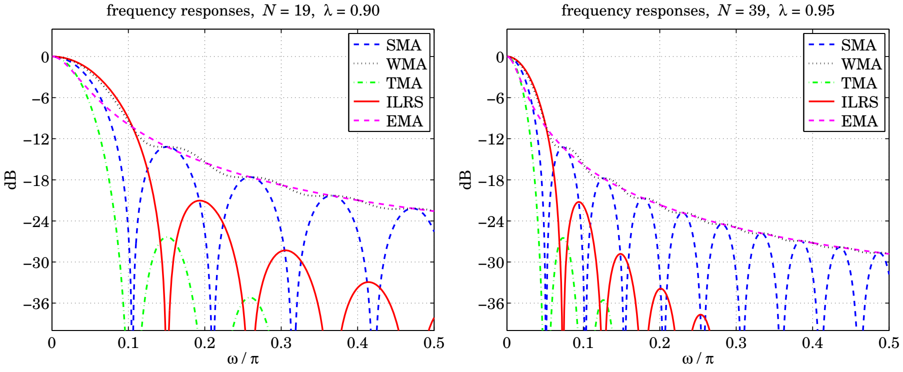


Fig. 6.15.1 Frequency responses of SMA, WMA, TMA, ILRS, and EMA filters.

As a small example, we also give for comparison the impulse responses, $\mathbf{h} = [h_0, h_1, \dots]$, of the SMA, WMA, TMA, and ILRS filters for the case $N = 5$,

$$
\begin{aligned}
&\text{(SMA)} \qquad \mathbf{h} = \frac{1}{5}[1,\ 1,\ 1,\ 1,\ 1]\\
&\text{(WMA)} \qquad \mathbf{h} = \frac{1}{15}[5,\ 4,\ 3,\ 2,\ 1]\\
&\text{(TMA)} \qquad \mathbf{h} = \frac{1}{25}[1,\ 2,\ 3,\ 4,\ 5,\ 4,\ 3,\ 2,\ 1]\\
&\text{(ILRS)} \qquad \mathbf{h} = \frac{1}{10}[2,\ 3,\ 3,\ 2,\ 0]
\end{aligned}
$$

with the SMA having constant weights, the WMA having linearly decreasing weights, the TMA has triangular weights, and the last coefficient h_{N-1} of the ILRS always being zero.

The following MATLAB functions implement the SMA, WMA, TMA, and ILRS moving averages. The input array **y** represents the financial data to be filtered.

```
a = sma(y,N,yin);    % simple moving average
a = wma(y,N,yin);    % weighted moving average
a = tma(y,N,yin);    % triangular moving average
a = ilrs(y,N,yin);   % integrated linear regression slope
```

The string variable `yin` specifies the way the filters are initialized and can take on the following values as explained further in Sec. 6.19,

```
yin = 'f',     % progressive filtering (default method)
yin = 'n',     % initial transients are NaNs (facilitates plotting)
yin = 'c',     % standard convolutional transients
```

Some comparisons of these and other moving average indicators are presented in Figures 6.18.2 and 6.21.1.

6.16 Predictive Moving Average Filters

The *predictive FIR* and *double EMA* filters discussed in Sects. 6.4 and 6.8 find application in stock market trading. Their main property is the elimination or shortening of the lag, which is accomplished by tracking both the local level and the local slope of the data. More discussion of these filters and their application in the trading context may be found in Refs. [297–308].

The local-level and local-slope FIR filters $h_a(k)$ and $h_b(k)$ were given in Eq. (6.4.4), and their filtering equations by (6.4.5). They define the following market indicators:

$$\boxed{\begin{aligned} a_n &= \sum_{k=0}^{N-1} h_a(k)\,y_{n-k} = \text{linear regression indicator}\\ b_n &= \sum_{k=0}^{N-1} h_b(k)\,y_{n-k} = \text{linear regression slope indicator}\\ a_n+b_n &= \sum_{k=0}^{N-1} h_1(k)\,y_{n-k} = \text{time-series forecast indicator}\end{aligned}} \tag{6.16.1}$$

where $h_1(k)= h_a(k)+h_b(k)$. The quantity a_n+b_n, denoted by $\hat{y}_{n+1/n}$, represents the one-step ahead forecast or prediction to time $n+1$ based on the data up to time n. More generally, the prediction τ steps ahead from time n is given by the following indicator, which we will refer to as the *predictive moving average* (PMA),

$$\boxed{\hat{y}_{n+\tau/n} = a_n + \tau b_n = \sum_{k=0}^{N-1} h_\tau(k)\,y_{n-k}} \quad \text{(PMA)} \tag{6.16.2}$$

where, as follows from Eq. (6.4.4), we have for $n = 0,1,\dots,N-1$,

$$\boxed{h_\tau(n)= h_a(n)+\tau h_b(n)= \frac{2(2N-1-3n)}{N(N+1)} + \tau\,\frac{6(N-1-2n)}{N(N^2-1)}} \tag{6.16.3}$$

The time "advance" τ can be non-integer, positive, or negative. Positive τs correspond to forecasting, negative τs to delay or lag. In fact, the SMA and WMA are special cases of Eq. (6.16.3) for the particular choices of $\tau = -(N-1)/2$ and $\tau = -(N-1)/3$, respectively.

The phrase "linear regression indicators" is justified in Sec. 6.18. The filters $h_\tau(n)$ are very flexible and useful in the trading context, and are actually the *optimal filters* that have *minimum noise-reduction ratio* subject to the two constraints of having unity DC gain and lag equal to $-\tau$, that is, for fixed N, $h_\tau(n)$ is the solution of the optimization problem (for $N=1$, we ignore the lag constraint to get, $h_\tau(n)=1$, for $n=0$, and all τ):

$$\mathcal{R} = \sum_{n=0}^{N-1} h^2(n) = \min, \qquad \text{subject to} \quad \sum_{n=0}^{N-1} h(n) = 1\,, \quad \sum_{n=0}^{N-1} n\,h_\tau(n) = -\tau \tag{6.16.4}$$

This was solved in Sec. 6.4. The noise-reduction-ratio of these filters is,

$$\mathcal{R}_\tau = \sum_{n=0}^{N-1} h_\tau^2(n) = \frac{1}{N} + \frac{3(N-1+2\tau)^2}{N(N^2-1)} \tag{6.16.5}$$

We note the two special cases, first for the SMA filter having $\tau = -(N-1)/2$, and second, for the zero-lag filter $h_a(n)$ having $\tau = 0$,

$$\mathcal{R}_{\text{SMA}} = \frac{1}{N}\,, \quad \mathcal{R}_a = \frac{4N-2}{N(N+1)}$$

The transfer functions of the FIR filters $h_a(n), h_b(n)$ are not particularly illuminating, however, they are given below in rational form,

$$\begin{aligned} H_a(z) &= \sum_{n=0}^{N-1} h_a(n) z^{-n} = \frac{2}{N(N+1)}\;\frac{N(1-z^{-1})(2+z^{-N}) - (1+2z^{-1})(1-z^{-N})}{(1-z^{-1})^2}\\ H_b(z) &= \sum_{n=0}^{N-1} h_b(n) z^{-n} = \frac{6}{N(N^2-1)}\;\frac{N(1-z^{-1})(1+z^{-N}) - (1+z^{-1})(1-z^{-N})}{(1-z^{-1})^2}\end{aligned}$$

By a proper limiting procedure, one can easily verify the unity-gain and zero-lag properties, $H_a(z)\big|_{z=1} = 1$, and, $\bar{n} = -H_a'(z)\big|_{z=1} = 0$.

The ILRS filter mentioned in the previous section is defined as the *integration*, or cumulative sum, of the slope filter, which can be evaluated explicitly resulting in (6.15.9),

$$h(n) = \sum_{k=0}^{n} h_b(k) = \sum_{k=0}^{n} \frac{6(N-1-2k)}{N(N^2-1)} = \frac{6(n+1)(N-1-n)}{N(N^2-1)} \tag{6.16.6}$$

where $0 \le n \le N-1$. For $n > N$, since $h_b(k)$ has duration N, the above sum remains constant and equal to zero, i.e., equal to its final value,

$$\sum_{k=0}^{N-1} h_b(k) = 0$$

The corresponding transfer function is the integrated (accumulated) form of $H_b(z)$ and is easily verified to be as in Eq. (6.15.9),

$$H(z)=\frac{H_b(z)}{1-z^{-1}}$$

The following MATLAB function, **pma**, implements Eq. (6.16.2) and the related indicators, where the input array **y** represents the financial data to be filtered. The function, **pmaimp**, implements the impulse response of Eq. (6.16.3).

```
at = pma(y,N,tau,yin);                  % at = a + tau*b, prediction distance tau

 a = pma(y,N,0,yin);                    % local-level indicator
 b = pma(y,N,1,yin)-pma(y,N,0,yin);     % local-slope indicator

af = pma(y,N,1,yin);                    % time-series forecast indicator, af = a + b

ht = pmaimp(N,tau);                     % impulse response of predictive filter
ha = pmaimp(N,0);                       % impulse response of local level filter
hb = pmaimp(N,1)-pmaimp(N,0);           % impulse response of local slope filter
```

and again, the string variable `yin` specifies the way the filters are initialized and can take on the following values,

```
yin = 'f',     % progressive filtering (default method)
yin = 'n',     % initial transients are NaNs (facilitates plotting)
yin = 'c',     % standard convolutional transients
```

A few examples of impulse responses are as follows, for $N=5,8,11$,

$$
\begin{aligned}
N=5\,,\quad & \mathbf{h}_a=\frac{1}{5}[3,\,2,\,1,\,0,\,-1] && \text{(local level)}\\
& \mathbf{h}_b=\frac{1}{10}[2,\,1,\,0,\,-1,\,-2] && \text{(local slope)}\\
& \mathbf{h}_1=\frac{1}{10}[8,\,5,\,2,\,-1,\,-4] && \text{(time-series forecast)}\\
N=8\,,\quad & \mathbf{h}_a=\frac{1}{12}[5,\,4,\,3,\,2,\,1,\,0,\,-1,\,-2]\\
& \mathbf{h}_b=\frac{1}{84}[7,\,5,\,3,\,1,\,-1,\,-3,\,-5,\,-7]\\
& \mathbf{h}_1=\frac{1}{28}[14,\,11,\,8,\,5,\,2,\,-1,\,-4,\,-7]\\
N=11\,,\quad & \mathbf{h}_a=\frac{1}{22}[7,\,6,\,5,\,4,\,3,\,2,\,1,\,0,\,-1,\,-2,\,-3]\\
& \mathbf{h}_b=\frac{1}{110}[5,\,4,\,3,\,2,\,1,\,0,\,-1,\,-2,\,-3,\,-4,\,-5]\\
& \mathbf{h}_1=\frac{1}{55}[20,\,17,\,14,\,11,\,8,\,5,\,2,\,-1,\,-4,\,-7,\,-10]
\end{aligned}
$$

Some comparisons of PMA with other moving average indicators are shown in Figures 6.18.2 and 6.21.1.

6.17 Single, Double, and Triple EMA Indicators

As discussed in Sec. 6.6, the single EMA (SEMA), double EMA (DEMA), and triple EMA (TEMA) steady-state exponential smoothing recursions are as follows,

$$\boxed{\begin{aligned} e_{n/n-1}&=y_n-\hat{y}_{n/n-1}=y_n-a_{n-1}\\ a_n&=a_{n-1}+(1-\lambda)e_{n/n-1}\end{aligned}}\qquad\text{(SEMA)}\qquad(6.17.1)$$

$$\boxed{\begin{aligned} &e_{n/n-1}=y_n-\hat{y}_{n/n-1}=y_n-(a_{n-1}+b_{n-1})\\ &\begin{bmatrix}a_n\\b_n\end{bmatrix}=\begin{bmatrix}1&1\\0&1\end{bmatrix}\begin{bmatrix}a_{n-1}\\b_{n-1}\end{bmatrix}+\begin{bmatrix}1-\lambda^2\\(1-\lambda)^2\end{bmatrix}e_{n/n-1}\end{aligned}}\qquad\text{(DEMA)}\qquad(6.17.2)$$

$$\boxed{\begin{aligned} &e_{n/n-1}=y_n-\hat{y}_{n/n-1}=y_n-(a_{n-1}+b_{n-1}+c_{n-1})\\ &\begin{bmatrix}a_n\\b_n\\c_n\end{bmatrix}=\begin{bmatrix}1&1&1\\0&1&2\\0&0&1\end{bmatrix}\begin{bmatrix}a_{n-1}\\b_{n-1}\\c_{n-1}\end{bmatrix}+\begin{bmatrix}\alpha_1\\\alpha_2\\\alpha_3\end{bmatrix}e_{n/n-1}\end{aligned}}\qquad\text{(TEMA)}\qquad(6.17.3)$$

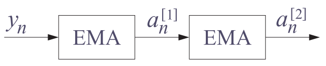
where

$$\alpha_1=1-\lambda^3\,,\quad \alpha_2=\frac{3}{2}(1-\lambda)(1-\lambda^2)\,,\quad \alpha_3=\frac{1}{2}(1-\lambda)^3$$

and $\hat{y}_{n/n-1}$ represents the forecast of y_n based on data up to time $n-1$. More generally, the forecast ahead by a distance τ is given by,

$$
\begin{aligned}
&\text{(SEMA)}\quad \hat{y}_{n+\tau/n}=a_n && & \hat{y}_{n/n-1}=a_{n-1}\\
&\text{(DEMA)}\quad \hat{y}_{n+\tau/n}=a_n+b_n\tau && \Rightarrow & \hat{y}_{n/n-1}=a_{n-1}+b_{n-1}\\
&\text{(TEMA)}\quad \hat{y}_{n+\tau/n}=a_n+b_n\tau+c_n\tau^2 && & \hat{y}_{n/n-1}=a_{n-1}+b_{n-1}+c_{n-1}
\end{aligned}
\qquad(6.17.4)
$$

We saw in Sec. 6.8 that an alternative way of computing the local level and local slope signals a_n,b_n in the DEMA case is in terms of the outputs of the cascade of two single EMAs, that is, with $\alpha=1-\lambda$,



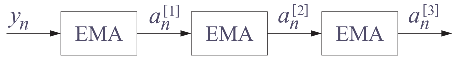
$$\boxed{\begin{aligned} a_n^{[1]}&=\lambda a_{n-1}^{[1]}+\alpha y_n\\ a_n^{[2]}&=\lambda a_{n-1}^{[2]}+\alpha a_n^{[1]}\end{aligned}}\qquad(6.17.5)$$

$$\boxed{\begin{aligned} a_n&=2a_n^{[1]}-a_n^{[2]}=\text{local level DEMA indicator}\\ b_n&=\frac{\alpha}{\lambda}\,(a_n^{[1]}-a_n^{[2]})=\text{local slope DEMA indicator}\end{aligned}}\qquad(6.17.6)$$

The transfer functions from y_n to the signals a_n,b_n were given in Eq. (6.8.5), and are expressible as follows in terms of the transfer function of a single EMA, $H(z)=\alpha/(1-\lambda z^{-1})$,

$$
\begin{aligned}
H_a(z) &= \frac{\alpha(1+\lambda-2\lambda z^{-1})}{(1-\lambda z^{-1})^2} = 2H(z) - H^2(z) = 1-\big[1-H(z)\big]^2 \\
H_b(z) &= \frac{\alpha^2(1-z^{-1})}{(1-\lambda z^{-1})^2} = \frac{\alpha}{\lambda}\big[H(z)-H^2(z)\big]
\end{aligned}
\tag{6.17.7}
$$

Similarly, in the TEMA case, the signals a_n, b_n, c_n can be computed from the outputs of three successive single EMAs via the following relationships,



$$
\boxed{
\begin{aligned}
a_n^{[1]} &= \lambda a_{n-1}^{[1]} + \alpha y_n \\
a_n^{[2]} &= \lambda a_{n-1}^{[2]} + \alpha a_n^{[1]} \\
a_n^{[3]} &= \lambda a_{n-1}^{[3]} + \alpha a_n^{[2]}
\end{aligned}
}
\tag{6.17.8}
$$

$$
\begin{bmatrix} a_n \\ b_n \\ c_n \end{bmatrix} = \frac{1}{2\lambda^2}
\begin{bmatrix}
6\lambda^2 & -6\lambda^2 & 2\lambda^2 \\
\alpha(1+5\lambda) & -2\alpha(1+4\lambda) & \alpha(1+3\lambda) \\
\alpha^2 & -2\alpha^2 & \alpha^2
\end{bmatrix}
\begin{bmatrix} a_n^{[1]} \\ a_n^{[2]} \\ a_n^{[3]} \end{bmatrix}
\tag{6.17.9}
$$

where $\alpha = 1-\lambda$. See also Eqs. (6.8.9)–(6.8.13). In particular, we have,

$$
\boxed{a_n = 3a_n^{[1]} - 3a_n^{[2]} + a_n^{[3]}} \quad \text{(local level TEMA indicator)}
\tag{6.17.10}
$$

Initialization issues for the single EMA, DEMA, and TEMA recursions are discussed in Sec. 6.19. The following MATLAB functions implement the corresponding filtering operations, where the input array $\mathbf{y}$ represents the financial data to be filtered.

```
             a = sema(y,N,yin);   % single exponential moving average
     [a,b,a1,a2] = dema(y,N,yin);   % double exponential moving average
[a,b,c,a1,a2,a3] = tema(y,N,yin);   % triple exponential moving average
```

The variable `yin` specifies the way the filters are initialized and can take on the following possible values,

```
yin = y(1)   % default for SEMA
yin = 'f'    % fits polynomial to first N samples, default for DEMA, TEMA
yin = 'c'    % cascaded initialization for DEMA, TEMA, described in Sect. 6.19
yin = any vector of initial values of [a], [a;b], or [a;b;c] at n=-1
yin = [0], [0;0], or [0;0;0] for standard convolutional output
```

Even though the EMA filters are IIR filters, traders prefer to specify the parameter λ of the EMA recursions through the SMA-equivalent length N defined as in Eq. (6.1.16),

$$
\lambda = \frac{N-1}{N+1} \quad \Leftrightarrow \quad N = \frac{1+\lambda}{1-\lambda}
\tag{6.17.11}
$$

The use of DEMA and TEMA as market indicators with less lag was first advocated by Mulloy [297,298]. Some comparisons of these with other moving average indicators are shown in Fig. 6.18.2.

6.18 Linear Regression and R-Square Indicators

In the literature of technical analysis, the PMA indicators of Eq. (6.16.1) are usually not implemented as FIR filters, but rather as successive fits of straight lines to the past N data from the current data point, that is, over the time span, $[n-N+1, n]$, for each n. This is depicted Fig. 6.18.1 below.

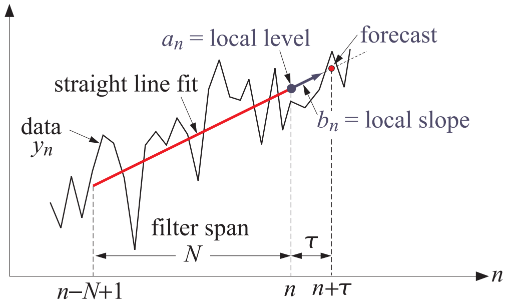


Fig. 6.18.1 Local linear regression and prediction.

They have been rediscovered many times in the past and different names given to them. For example, Lafferty [300] calls them "end-point moving averages", while Rafter [303] refers to them as "moving trends." Their application as a forecasting tool was discussed first by Chande [299].

Because of the successive fitting of straight lines, the signals a_n, b_n are known as the "linear regression" indicator and the "linear regression slope" indicator, respectively. The a_n indicator is also known as "least-squares moving average" (LSMA).

For each $n \geq N-1$, the signals a_n, b_n can be obtained as the *least-squares solution* of the following $N{\times}2$ *overdetermined* system of linear equations in two unknowns:

$$
a_n - kb_n = y_{n-k}\,, \quad k = 0, 1, \dots, N-1
\tag{6.18.1}
$$

which express the fitting of a straight line, $a + b\tau$, to the data $[y_{n-N+1}, \dots, y_{n-1}, y_n]$, that is, over the time window, $[n-N+1, n]$, where a is the intercept at the end of the line. The overdetermined system (6.18.1) can be written compactly in matrix form by defining the length-N column vectors,

$$
\mathbf{u} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ \vdots \\ 1 \\ \vdots \\ 1 \end{bmatrix}, \quad
\mathbf{k} = \begin{bmatrix} 0 \\ 1 \\ 2 \\ \vdots \\ k \\ \vdots \\ N-1 \end{bmatrix}, \quad
\mathbf{y}_n = \begin{bmatrix} y_n \\ y_{n-1} \\ y_{n-2} \\ \vdots \\ y_{n-k} \\ \vdots \\ y_{n-N+1} \end{bmatrix}
\quad \Rightarrow \quad
[\mathbf{u}, -\mathbf{k}] \begin{bmatrix} a_n \\ b_n \end{bmatrix} = \mathbf{y}_n
\tag{6.18.2}
$$

with the least-squares solution expressed in the following MATLAB-like vectorial notation using the backslash operator,

$$\begin{bmatrix} a_n \\ b_n \end{bmatrix} = [\mathbf{u}, -\mathbf{k}] \setminus \mathbf{y}_n \tag{6.18.3}$$

Indeed, this is the solution for the local level and local slope parameters a, b that minimize the following least-squares performance index, defined for each n,

$$\mathcal{J}_n = \sum_{k=0}^{N-1} \big(a - bk - y_{n-k}\big)^2 = \min \tag{6.18.4}$$

In order to account also for the initial transient period, $0 \le n \le N-1$, we may change the upper limit of summation in Eq. (6.18.4) to,

$$\mathcal{J}_n = \sum_{k=0}^{\min(n,N-1)} \big(a - bk - y_{n-k}\big)^2 = \min \tag{6.18.5}$$

which amounts to fitting a straight line to a progressively longer and longer data vector until its length becomes equal to N, that is, starting with $a_0 = y_0$ and $b_0 = 0$,[†] we fit a line to $[y_0, y_1]$ to get a_1, b_1, then fit a line to $[y_0, y_1, y_2]$ to get a_2, b_2, and so on until $n = N-1$, and beyond that, we continue with a length-N data window.

Thus, we may state the complete solution for all $0 \le n \le L-1$, where L is the length of the data vector y_n, using the backslash notation,[‡]

$$\boxed{\begin{array}{l} \text{for each, } n = 0, 1, 2, \dots, L-1, \text{ do:} \\ \quad K_n = \min(n, N-1) + 1 = \text{fitting length, } K_n = N \text{ when } n \ge N-1 \\ \quad \mathbf{k} = \big[0 : K_n - 1\big]' = \text{column vector, length } K_n \\ \quad \mathbf{y}_n = y(n - \mathbf{k}) = \text{column vector, } [y_n, y_{n-1}, \dots, y_{n-K_n+1}]^T \\ \quad \mathbf{u} = \text{ones}(K_n, 1) = \text{column vector} \\ \quad \begin{bmatrix} a_n \\ b_n \end{bmatrix} = [\mathbf{u}, -\mathbf{k}] \setminus \mathbf{y}_n = \text{linear regression indicators} \\ \quad R^2(n) = \big(\text{corr}(-\mathbf{k}, \mathbf{y}_n)\big)^2 = 1 - \det\big(\text{corrcoef}(-\mathbf{k}, \mathbf{y}_n)\big) = R^2 \text{ indicator} \end{array}} \tag{6.18.6}$$

where we also included the so-called R*-square indicator*,[*] which is the *coefficient of determination* for the linear fit, and quantifies the strength of the linear relationship, that is, higher values of R^2 suggest that the linear fit is statistically significant with a certain degree of confidence (usually taken to be at the 95% confidence level).

The MATLAB function, **r2crit** in the OSP toolbox, calculates the critical values R_c^2 of R^2 for a given N and given confidence level p, such that if $R^2(n) > R_c^2$, then the linear fit is considered to be statistically significant for the nth segment. Some typical critical values of R_c^2 at the $p = 0.95$ and $p = 0.99$ levels are listed below in Eq. (6.18.7), and were computed with the following MATLAB commands (see also [280]),

[†] $b_0 = 0$ is an arbitrary choice since b_0 is indeterminate for $N = 1$.

[‡] the backslash solution also correctly generates the case $n = 0$, i.e., $a_0 = y_0$ and $b_0 = 0$.

[*] where, **corr**, **det**, and **corrcoef**, are built-in MATLAB functions.

```
N = [5, 10, 14, 20, 25, 30, 50, 60, 120];
R2c = r2crit(N,0.95);
R2c = r2crit(N,0.99);
```

N	$p = 0.95$	$p = 0.99$
5	0.7711	0.9180
10	0.3993	0.5846
14	0.2835	0.4374
20	0.1969	0.3152
25	0.1569	0.2552
30	0.1303	0.2143
50	0.0777	0.1303
60	0.0646	0.1090
120	0.0322	0.0549

(6.18.7)

The *standard errors* for the successive linear fits, as well as the standard errors for the quantities a_n, b_n, can be computed by including the following lines within the for-loop in Eq. (6.18.6),

$$\boxed{\begin{array}{l} \mathbf{e}_n = \mathbf{y}_n - [\mathbf{u}, -\mathbf{k}] \begin{bmatrix} a_n \\ b_n \end{bmatrix} = \text{fitting error, column vector} \\ \sigma_e(n) = \sqrt{\dfrac{\mathbf{e}_n^T \mathbf{e}_n}{K_n - 2}} = \text{standard error} \\ \sigma_a(n) = \sqrt{\dfrac{2(2K_n - 1)}{K_n(K_n + 1)}}\, \sigma_e(n) = \text{standard error for } a_n \\ \sigma_b(n) = \sqrt{\dfrac{12}{K_n(K_n^2 - 1)}}\, \sigma_e(n) = \text{standard error for } b_n \end{array}} \tag{6.18.8}$$

The derivation of the expressions for $\sigma_e, \sigma_a, \sigma_b$ follows from the standard theory of least-squares linear regression. For example, linear regression based on the K pairs, (x_k, y_k), $k = 0, 1, \dots, K-1$, results in the estimates, $\hat{y}_k = a + b x_k$, and error residuals, $e_k = y_k - \hat{y}_k$, from which the standard errors can be calculated from the following expressions [349],

$$\sigma_e^2 = \frac{1}{K-2} \sum_{k=0}^{N-1} e_k^2\,, \quad \sigma_a^2 = \sigma_e^2 \frac{\overline{x^2}}{K\sigma_x^2}\,, \quad \sigma_b^2 = \frac{\sigma_e^2}{K\sigma_x^2} \tag{6.18.9}$$

For our special case of equally-spaced data, $x_k = -k$, we easily find,

$$\begin{aligned} \bar{x} &= -\bar{k} = -\frac{1}{K} \sum_{k=0}^{K-1} k = -\frac{K-1}{2} \\ \overline{x^2} &= \overline{k^2} = \frac{1}{K} \sum_{k=0}^{K-1} k^2 = \frac{(K-1)(2K-1)}{6} \\ \sigma_x^2 &= \sigma_k^2 = \overline{k^2} - \bar{k}^2 = \frac{K^2 - 1}{12} \end{aligned} \quad \Rightarrow \quad \begin{aligned} \sigma_a^2 &= \frac{2(2K-1)}{K(K+1)}\, \sigma_e^2 \\ \sigma_b^2 &= \frac{12}{K(K^2-1)}\, \sigma_e^2 \end{aligned}$$

Standard error bands [329], as well as other types of bands and envelopes, and their use as market indicators, are discussed further in Sec. 6.22. The MATLAB function, **lreg**, implements Eqs. (6.18.6) and (6.18.8) with usage,

```
[a,b,R2,se,sa,sb] = lreg(y,N,init);    % linear regression indicators
```

```
y = data                  a = local level     se = standard error
N = window length         b = local slope     sa = standard error for a
init = initialization     R2 = R-square       sb = standard error for b
```

where `init` specifies the initialization scheme and takes on the following values,

```
init = 'f', progressive linear fitting of initial N-1 samples, default
init = 'n', replacing initial N-1 samples of a,b,R2,se,sa,sb by NaNs
```

The local level and local slope outputs a_n, b_n are identical to those produced by the function **pma** of Sec. 6.16.

Generally, these indicators behave similarly to the DEMA indicators, but both indicator types should be used with some caution since they are too quick to respond to price changes and sometimes tend to produce false buy/sell signals. In other words, some delay may miss the onset of a trend but provides more safety.

Example 6.18.1: Fig. 6.18.2 compares the SMA, EMA, WMA, PMA/linear regression, DEMA, TEMA indicators. The data are from [305] and represent daily prices for Nicor-Gas over 130 trading days starting on Sept. 1, 2006. The included excel file, **nicor.xls**, contains the open-high-low-close prices in its first four columns. The left graphs were produced by the following MATLAB code, in which the function, **ohlc**, from the OSP toolbox, makes an OHLC[†] bar chart,

```
Y = xlsread('nicor.xls');   % read Nicor-Gas data
Y = Y(1:130,:);             % keep only 130 trading days
y = Y(:,4);                 % closing prices
t = 0:length(y)-1;          % trading days

N = 20;                     % filter length

figure;                     % SMA, EMA, WMA
plot(t,sma(y,N),'r-', t,sema(y,N),'k--', t,wma(y,N),'g-.');  hold on;
ohlc(t,Y(:,1:4));           % add OHLC bar chart


figure;                     % PMA/lreg, DEMA, TEMA
plot(t,pma(y,N,0),'r-', t,dema(y,N),'k--',  t,tema(y,N),'g-.'); hold on;
ohlc(t,Y(:,1:4));           % add OHLC bar chart
```

The filter length was $N = 20$. The right graphs are an expanded view of the range $[45, 90]$ days and show more clearly the reduced lag of the PMA, DEMA, and TEMA indicators. At about the 57th trading day, these indicators turn downwards but still lie above the data, therefore, they would correctly issue a “sell” signal. By contrast, the SMA, EMA, and WMA indicators are rising and lie below the data, and they would issue a “buy” signal.

[†] Open-High-Low-Close

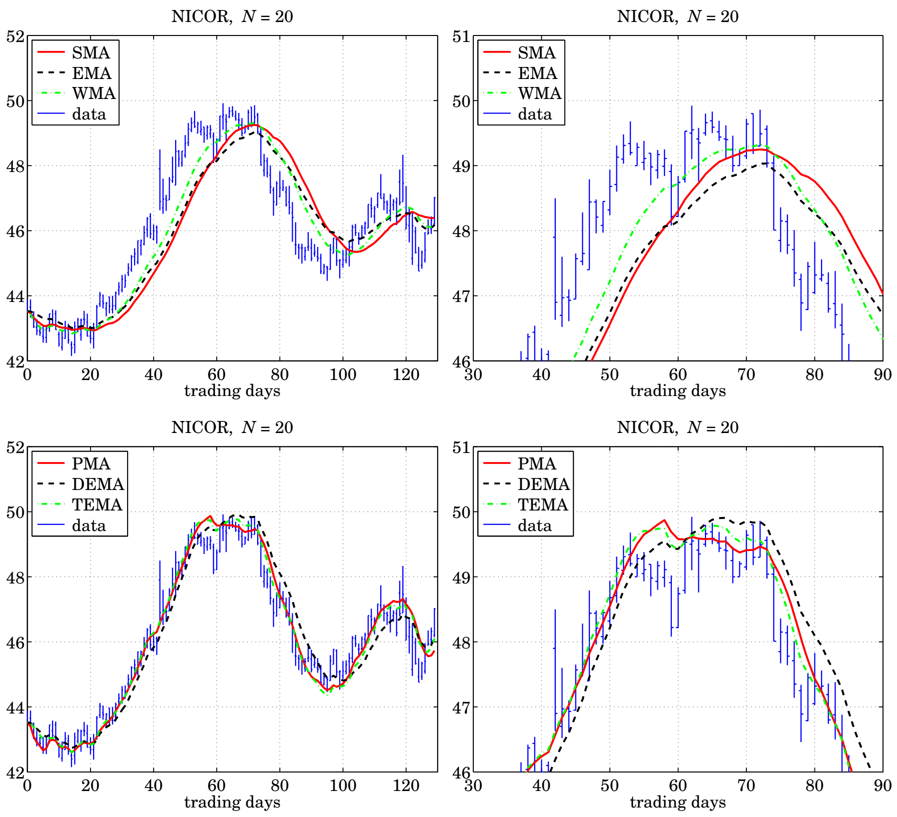


Fig. 6.18.2 Comparison of SMA, EMA, WMA with PMA/LREG, DEMA, TEMA indicators.

Fig. 6.21.1 in Sec. 6.21 compares the PMA with two other indicators of reduced lag, namely, the Hull moving average (HMA), and the exponential Hull moving average (EHMA).

The R-squared and slope indicators are also useful in determining the direction of trend. Fig. 6.18.3 shows the PMA/linear regression indicator, a_n, for the same Nicor data, together with the corresponding $R^2(n)$ signal, and the slope signal b_n, using again a filter length of $N = 20$. They were computed with the MATLAB code:

```
[a,b,R2] = lreg(y,N);          % local level, local slope, and R-squared

% equivalent calculation:
% a = pma(y,N,0);
% b = pma(y,N,1)-pma(y,N,0);
```

For $N = 20$, the critical value of R^2 at the 95% confidence level is $R_c^2 = 0.1969$, determined in Eq. (6.18.7), and is displayed as the horizontal dashed line on the R^2 graph.

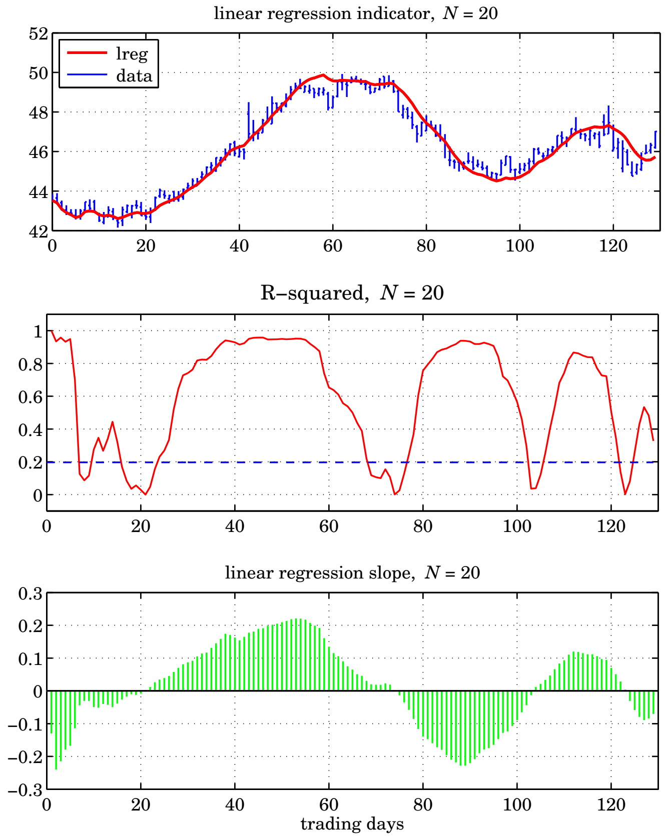


Fig. 6.18.3 PMA/linear regression, R-squared, and linear regression slope indicators.

When $R^2(n)$ is small, below R_c^2, it indicates lack of a trend with the data moving sideways, and corresponds to slope b_n near zero.

When $R^2(n)$ rises near unity, it indicates a strong trend, but it does not indicate the direction, upwards or downwards. This is indicated by the slope indicator b_n, which is positive when the signal is rising, and negative, when it is falling. More discussion on using these three indicators in conjunction may be found in [305]. □

6.19 Initialization Schemes

In Eq. (6.18.6), one solves a shorter linear fitting problem of progressively increasing length during the transient period, $0 \le n < N-1$, and then switches to fixed length N for $n \ge N-1$.

The same idea can be applied to all FIR filters, such as the SMA, WMA, TMA, and the PMA filter, $h_\tau(n)$, that is, to use the same type of filter, but of progressively increasing length, during the period $0 \le n < N-1$, and then switch to using the filters of fixed length N for $n \ge N-1$. The first $N-1$ outputs computed in this manner are not the same as the standard convolutional outputs obtained from the built-in function **filter**, because the latter uses the same length-N filter and assumes zero initial internal states.

To clarify this, consider the SMA case with $N = 5$, then the above procedure and the standard convolutional one compute the outputs in the following manner, agreeing only after $n \ge N-1 = 4$,

progressive	convolutional
$a_0 = y_0$	$a_0 = \frac{1}{5}y_0$
$a_1 = \frac{1}{2}(y_1 + y_0)$	$a_1 = \frac{1}{5}(y_1 + y_0)$
$a_2 = \frac{1}{3}(y_2 + y_1 + y_0)$	$a_2 = \frac{1}{5}(y_2 + y_1 + y_0)$
$a_3 = \frac{1}{4}(y_3 + y_2 + y_1 + y_0)$	$a_3 = \frac{1}{5}(y_3 + y_2 + y_1 + y_0)$
$a_4 = \frac{1}{5}(y_4 + y_3 + y_2 + y_1 + y_0)$	$a_4 = \frac{1}{5}(y_4 + y_3 + y_2 + y_1 + y_0)$
$a_5 = \frac{1}{5}(y_5 + y_4 + y_3 + y_2 + y_1)$	$a_5 = \frac{1}{5}(y_5 + y_4 + y_3 + y_2 + y_1)$
$\cdots$	$\cdots$

Similarly, the local level PMA filters, $\mathbf{h}_a$, of lengths up to $N = 5$ can be determined from Eq. (6.16.3), leading to the following progressive initializations,

$$\begin{aligned}
&N=1\,, \quad \mathbf{h}_a = [1]\,, && a_0 = y_0\\
&N=2\,, \quad \mathbf{h}_a = [1,0]\,, && a_1 = y_1\\
&N=3\,, \quad \mathbf{h}_a = \frac{1}{6}[5,2,-1]\,, && a_2 = \frac{1}{6}(5y_2 + 2y_1 - y_0)\\
&N=4\,, \quad \mathbf{h}_a = \frac{1}{10}[7,4,1,-2]\,, && a_3 = \frac{1}{10}(7y_3 + 4y_2 + y_1 - 2y_0)\\
&N=5\,, \quad \mathbf{h}_a = \frac{1}{5}[3,2,1,0,-1]\,, && a_4 = \frac{1}{5}(3y_4 + 2y_3 + y_2 - y_0)
\end{aligned}$$

and for the local slope filters $\mathbf{h}_b$,

$$\begin{aligned}
&N=1\,, \quad \mathbf{h}_b = [0]\,, && b_0 = 0\\
&N=2\,, \quad \mathbf{h}_b = [1,-1]\,, && b_1 = y_1 - y_0\\
&N=3\,, \quad \mathbf{h}_b = \frac{1}{2}[1,0,-1]\,, && b_2 = \frac{1}{2}(y_2 - y_0)\\
&N=4\,, \quad \mathbf{h}_b = \frac{1}{10}[3,1,-1,-3]\,, && b_3 = \frac{1}{10}(3y_3 + y_2 - y_1 - 3y_0)\\
&N=5\,, \quad \mathbf{h}_b = \frac{1}{10}[2,1,0,-1,-2]\,, && b_4 = \frac{1}{10}(2y_4 + y_3 - y_1 - 2y_0)
\end{aligned}$$

where, we arbitrarily set $\mathbf{h}_b = [0]$ for the case $N = 1$, since the slope is meaningless for a single data point. To see the equivalence of these with the least-square criterion of Eq. (6.18.5) consider, for example, the case $N = 5$ and $n = 2$,

$$\mathcal{J}_2 = (a-y_2)^2 + (a-b-y_1)^2 + (a-2b-y_0)^2 = \min$$

with minimization conditions,

$$\left.\begin{aligned} \frac{\partial J_2}{\partial a} &= 2(a-y_2)+2(a-b-y_1)+2(a-2b-y_0)=0 \\ \frac{\partial J_2}{\partial b} &= -2(a-b-y_1)-4(a-2b-y_0)=0 \end{aligned}\right. \quad\Rightarrow\quad \begin{aligned} 3a-3b &= y_2+y_1+y_0 \\ 3a-5b &= y1+2y_0 \end{aligned}$$

resulting in the solution,

$$a = \frac{1}{6}(5y_2+2y_1-y_0)\,,\quad b = \frac{1}{2}(y_2-y_0)$$

Similarly we have for the cases $n = 0$ and $n = 1$,

$$\begin{aligned} \mathcal{J}_0 &= (a-y_0)^2 = \min & &\Rightarrow & a &= y_0\,, & b &= \text{indeterminate} \\ \mathcal{J}_1 &= (a-y_1)^2+(a-b-y_0)^2 = \min & &\Rightarrow & a &= y_1\,, & b &= y_1-y_0 \end{aligned}$$

EMA Initializations

The single, double, and triple EMA difference equations (6.17.1)–(6.17.3), also need to be properly initialized at $n = -1$. For the single EMA case, a good choice is $a_{-1} = y_0$, which leads to the same value at $n = 0$, that is,

$$a_0 = \lambda a_{-1} + \alpha y_0 = \lambda y_0 + \alpha y_0 = y_0 \tag{6.19.1}$$

This is the default initialization for our function, **sema**. Another possibility is to choose the mean of the first N data samples, $a_{-1} = \text{mean}([y_0, y_1, \dots, y_{N-1}])$.

For DEMA, if we initialize both the first and the second EMAs as in Eq. (6.19.1), then we must choose, $a_{-1}^{[1]} = y_0$, which leads to $a_0^{[1]} = y_0$, which then would require that, $a_{-1}^{[2]} = a_0^{[1]} = y_0$, thus, in this scheme, we would choose,

$$\begin{bmatrix} a_{-1}^{[1]} \\ a_{-1}^{[2]} \end{bmatrix} = \begin{bmatrix} y_0 \\ y_0 \end{bmatrix} \quad\Rightarrow\quad \begin{bmatrix} a_{-1} \\ b_{-1} \end{bmatrix} = \begin{bmatrix} 2 & -1 \\ \alpha/\lambda & -\alpha/\lambda \end{bmatrix} \begin{bmatrix} a_{-1}^{[1]} \\ a_{-1}^{[2]} \end{bmatrix} = \begin{bmatrix} y_0 \\ 0 \end{bmatrix} \tag{6.19.2}$$

This is the default initialization method for our function, **dema**. Another possibility is to fit a straight line to a few initial data [297,348], such as the first N data, where N is the equivalent SMA length, $N = (1+\lambda)/(1-\lambda)$, and then extrapolate the line backwards to $n = -1$. This can be accomplished in MATLAB-like notation as follows,

$$\begin{aligned} \mathbf{n} &= [1:N]' = \text{column vector} \\ \mathbf{y} &= [y_0, y_1, \dots, y_{N-1}]' = \text{column vector} \\ \mathbf{u} &= \text{ones}(\text{size}(\mathbf{n})) \\ \begin{bmatrix} a_{-1} \\ b_{-1} \end{bmatrix} &= [\mathbf{u}, \mathbf{n}] \setminus \mathbf{y} \end{aligned} \tag{6.19.3}$$

If one wishes to use the cascade of two EMAs, then the EMA signals, $a_n^{[1]}, a_n^{[2]}$, must be initialized by first applying Eq. (6.19.3), and then using the inverse matrix relationship of Eq. (6.17.6), i.e.,

$$\begin{bmatrix} a_{-1}^{[1]} \\ a_{-1}^{[2]} \end{bmatrix} = \begin{bmatrix} 1 & -\lambda/\alpha \\ 1 & -2\lambda/\alpha \end{bmatrix} \begin{bmatrix} a_{-1} \\ b_{-1} \end{bmatrix} \tag{6.19.4}$$

A third possibility [280] is to initialize the first EMA with $a_{-1}^{[1]} = y_0$, then calculate the output at the time instant $n = N-1$ and use it to initialize the second EMA at $n = N$, that is, define $a_{N-1}^{[2]} = a_{N-1}^{[1]}$. This value can be iterated backwards to $n = -1$ to determine the proper initial value $a_{-1}^{[2]}$ such that, if iterated forward, it would arrive at the chosen value $a_{N-1}^{[2]} = a_{N-1}^{[1]}$. Thus, the steps in this scheme are as follows,

$$\begin{aligned} & a_{-1}^{[1]} = y_0 \\ & \text{for } n = 0, 1, \dots, N-1, \\ & \qquad a_n^{[1]} = \lambda\, a_{n-1}^{[1]} + \alpha\, y_n \\ & \text{end} \end{aligned} \quad\Rightarrow\quad \begin{aligned} & a_{N-1}^{[2]} = a_{N-1}^{[1]} \\ & \text{for } n = N-1, \dots, 1, 0, \\ & \qquad a_{n-1}^{[2]} = \frac{1}{\lambda}\big(a_n^{[2]} - \alpha\, a_n^{[1]}\big) \\ & \text{end} \end{aligned} \tag{6.19.5}$$

Upon exit from the second loop, one has $a_{-1}^{[2]}$, then, one can transform the calculated $a_{-1}^{[1]}, a_{-1}^{[2]}$ to the a_n, b_n basis in order to get the DEMA recursion started,

$$\begin{bmatrix} a_{-1} \\ b_{-1} \end{bmatrix} = \begin{bmatrix} 2 & -1 \\ \alpha/\lambda & -\alpha/\lambda \end{bmatrix} \begin{bmatrix} a_{-1}^{[1]} \\ a_{-1}^{[2]} \end{bmatrix}$$

Such cascaded initialization scheme for DEMA (and TEMA below) is somewhat ad hoc since the EMA filters are IIR and there is nothing special about the time $n = N$; one, could just as well wait until about $n = 6N$ when typically all transient effects have disappeared. We have found that the schemes described in Eqs. (6.19.2) and (6.19.3) work the best.

Finally, we note that for ordinary convolutional output, one would choose zero initial values,

$$\begin{bmatrix} a_{-1} \\ b_{-1} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad\Rightarrow\quad \begin{bmatrix} a_{-1}^{[1]} \\ a_{-1}^{[2]} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

All of the above initialization choices are incorporated in the function, **dema**. For TEMA, the default initialization is similar to that of Eq. (6.19.2), that is,

$$\begin{bmatrix} a_{-1}^{[1]} \\ a_{-1}^{[2]} \\ a_{-1}^{[3]} \end{bmatrix} = \begin{bmatrix} y_0 \\ y_0 \\ y_0 \end{bmatrix} \quad\Rightarrow\quad \begin{bmatrix} a_{-1} \\ b_{-1} \\ c_{-1} \end{bmatrix} = \begin{bmatrix} y_0 \\ 0 \\ 0 \end{bmatrix} \tag{6.19.6}$$

Alternatively, one can fit a second-order polynomial to the first few data samples, such as the first $2N$ samples [297], and extrapolate them back to $n = -1$. The fitting can be done with the following MATLAB-like code,

$$\begin{aligned}
&\mathbf{n} = [1 : 2N-1]' = \text{column vector}\\
&\mathbf{y} = [y_0, y_1, \dots, y_{2N-1}]' = \text{column vector}\\
&\mathbf{u} = \text{ones}(\text{size}(\mathbf{n}))\\
&\begin{bmatrix} a_{-1} \\ b_{-1} \\ c_{-1} \end{bmatrix} = [\mathbf{u}, \mathbf{n}, \mathbf{n}^2] \setminus \mathbf{y}
\end{aligned}$$

The cascaded initialization scheme is also possible in which the output of the first EMA at time $n = N-1$ serves to initialize the second EMA at $n = N$, and the output of the second EMA at $n = 2N-1$ serves to initialize the third EMA at $n = 2N$. This, as well as the second-order polynomial fitting initialization schemes are incorporated in the function, **tema**.

A special case of the EMA indicator is "Wilder's Exponential Moving Average" [281], known as WEMA. It is used widely and appears in several other indicators, such as the "Relative Strength Index" (RSI), the "Average True Range" (ATR), and the "Directional Movement System" (±DMI and ADX), discussed in Sec. 6.23. An N-point WEMA is defined to be an ordinary EMA with λ, α parameters,

$$\boxed{\alpha = \frac{1}{N}\,, \quad \lambda = 1 - \alpha = 1 - \frac{1}{N}} \quad \text{(WEMA parameters)} \tag{6.19.7}$$

It is equivalent to an EMA with effective length, N_e, determined as follows,

$$\lambda = \frac{N_e - 1}{N_e + 1} = 1 - \frac{1}{N} \quad \Rightarrow \quad \boxed{N_e = 2N - 1} \tag{6.19.8}$$

The corresponding filtering equation for calculating the smoothed local-level signal a_n from the input data y_n, will be,

$$a_n = \lambda a_{n-1} + \alpha y_n = a_{n-1} + \alpha(y_n - a_{n-1})$$

or, for $n \ge 0$,

$$\boxed{a_n = a_{n-1} + \frac{1}{N}(y_n - a_{n-1})} \quad \text{(WEMA)} \tag{6.19.9}$$

The required initial value a_{-1} can be chosen in a variety of ways, just as in EMA. However, by convention [281], the default way of fixing it is similar to that in Eq. (6.19.5). It is defined by choosing the value of a_n at time $n = N-1$ to be the mean of first N input values, then, a_{N-1} is back-tracked to time $n = -1$, thus, we have,

$$\begin{array}{l}
a_{N-1} = \dfrac{1}{N}(y_0 + y_1 + \cdots + y_{N-1})\\
\text{for } n = N-1, \dots, 1, 0,\\
\qquad a_{n-1} = \dfrac{1}{\lambda}(a_n - \alpha y_n)\\
\text{end}
\end{array} \tag{6.19.10}$$

Upon exit from the loop, one has the proper starting value of a_{-1}. The following MATLAB function, **wema**, implements WEMA with such default initialization scheme,

```
a = wema(y,N,ain);     % Wilder's EMA

y   = signal to be smoothed
N   = effective length, (EMA alpha = 1/N, lambda = 1-1/N)
ain = any initial value
    = 'm', default, as in Eq.(6.19.10)
    = 0, for standard convolutional output

a = smoothed version of y
```

6.20 Butterworth Moving Average Filters

Butterworth moving average (BMA) lowpass filters, are useful alternatives [285] to the first-order EMA filters, and have comparable smoothing properties and shorter lag. Here, we summarize their properties and filtering implementation, give explicit design equations for orders $M = 1, 2, 3$, and derive a general expression for their lag.

Digital Butterworth filters are characterized by two parameters, the filter order M, and the 3-dB cutoff frequency f_0 in Hz, or, the corresponding digital frequency in units of radians per sample, $\omega_0 = 2\pi f_0/f_s$, where f_s is the sampling rate in Hz. We may also define the period of f_0 in units of samples/cycle, $N = f_s/f_0$, so that, $\omega_0 = 2\pi/N$.

We follow the design method of Ref. [30] based on the bilinear transformation, although the matched z-transform method has also been used [285]. If the filter order is even, say, $M = 2K$, then, there are K second-order sections, and if it is odd, $M = 2K+1$, there is an additional first-order section. Both cases can be combined into one by writing,

$$M = 2K + r\,, \quad r = 0, 1 \tag{6.20.1}$$

Then, the transfer function can be expressed in the following cascaded and direct forms,

$$\begin{aligned}
H(z) &= \left[\frac{G_0(1+z^{-1})}{1 + a_{01}z^{-1}}\right]^r \prod_{i=1}^{K}\left[\frac{G_i(1+z^{-1})^2}{1 + a_{i1}z^{-1} + a_{i2}z^{-2}}\right]\\
&= \frac{G(1+z^{-1})^M}{1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_M z^{-M}}
\end{aligned} \tag{6.20.2}$$

where the notation $[\;]^r$ means that the first-order factor is absent if $r = 0$ and present if $r = 1$. The corresponding first-order coefficients are,

$$G_0 = \frac{\Omega_0}{\Omega_0 + 1}\,, \qquad a_{01} = \frac{\Omega_0 - 1}{\Omega_0 + 1} \tag{6.20.3}$$

The second-order coefficients are , for $i = 1, 2, \dots, K$,

$$
\begin{aligned}
G_i &= \frac{\Omega_0^2}{1 - 2\Omega_0\cos\theta_i + {\Omega_0}^2} \\
a_{i1} &= \frac{2\,(\Omega_0^2 - 1)}{1 - 2\Omega_0\cos\theta_i + {\Omega_0}^2}\,, \quad a_{i2} = \frac{1 + 2\Omega_0\cos\theta_i + \Omega_0^2}{1 - 2\Omega_0\cos\theta_i + {\Omega_0}^2}
\end{aligned}
\tag{6.20.4}
$$

where the angles θ_i are defined by,

$$
\boxed{\theta_i = \frac{\pi}{2M}\,(M - 1 + 2i)\,, \quad i = 1, 2, \dots, K}
\tag{6.20.5}
$$

and the quantity Ω_0 is the equivalent analog 3-dB frequency defined as,

$$
\boxed{\Omega_0 = \tan\left(\frac{\omega_0}{2}\right) = \tan\left(\frac{\pi f_0}{f_s}\right) = \tan\left(\frac{\pi}{N}\right)}
\tag{6.20.6}
$$

We note that the filter sections have zeros at $z = -1$, that is, at the Nyquist frequency, $f = f_s/2$, or, $\omega = \pi$. Setting $\Omega = \tan(\omega/2)$, the magnitude response of the designed digital filter can be expressed simply as follows:

$$
|H(\omega)|^2 = \frac{1}{1 + \big(\Omega/\Omega_0\big)^{2M}} = \frac{1}{1 + \big(\tan(\omega/2)/\Omega_0\big)^{2M}}
\tag{6.20.7}
$$

Each section has unity gain at DC. Indeed, setting $z = 1$ in Eq. (6.20.2), we obtain the following condition, which can be verified from the given definitions,

$$
\frac{4G_i}{1 + a_{i1} + a_{i2}} = 1 \qquad \text{and} \qquad \frac{2G_0}{1 + a_{01}} = 1
$$

Moreover, the filter lag can be shown to be (cf. Problem 6.12), for any $M \ge 1$ and $N > 2$,

$$
\boxed{\bar{n} = \frac{1}{2\Omega_0\,\sin\left(\dfrac{\pi}{2M}\right)} = \frac{1}{2\tan\left(\dfrac{\pi}{N}\right)\sin\left(\dfrac{\pi}{2M}\right)}} \quad \text{(lag)}
\tag{6.20.8}
$$

For $M \gtrsim 2$ and $N \gtrsim 5$, it can be approximated well by [284],

$$
\bar{n} = \frac{MN}{\pi^2}
$$

The overall numerator gain in the direct form is the product of gains,

$$
G = G_0^r G_1 G_2 \cdots G_K
$$

and the direct-form numerator coefficients are the coefficients of the binomial expansion of $(1 + z^{-1})^M$ times the overall gain G. The direct-form denominator coefficients are obtained by convolving the coefficients of the individual sections, that is, setting, $\mathbf{a} = [1]$ if M is even, and, $\mathbf{a} = [1, a_{01}]$ if M is odd, then the vector, $\mathbf{a} = [1, a_1, a_2, \dots, a_M]$, can be constructed recursively by,

$$
\begin{aligned}
&\text{for } i = 1, 2, \dots, K \\
&\qquad \mathbf{a} = \text{conv}\big(\mathbf{a},\ [1, a_{i1}, a_{i2}]\big)
\end{aligned}
\tag{6.20.9}
$$

For example, we have,

$$
\begin{aligned}
&M = 2\,, \quad \mathbf{a} = [1, a_{11}, a_{12}] \\
&M = 3\,, \quad \mathbf{a} = \text{conv}\big([1, a_{01}],\ [1, a_{11}, a_{12}]\big) = [1,\ a_{01} + a_{11},\ a_{12} + a_{01}a_{11},\ a_{01}a_{12}]
\end{aligned}
$$

From these, we obtain the following explicit expressions, for $M = 2$,

$$
\begin{aligned}
G &= \frac{\Omega_0^2}{\Omega_0^2 + \sqrt{2}\Omega_0 + 1}\,, \quad a_1 = \frac{2\,(\Omega_0^2 - 1)}{\Omega_0^2 + \sqrt{2}\Omega_0 + 1}\,, \quad a_2 = \frac{\Omega_0^2 - \sqrt{2}\Omega_0 + 1}{\Omega_0^2 + \sqrt{2}\Omega_0 + 1} \\
H(z) &= \frac{G\,(1 + 2z^{-1} + z^{-2})}{1 + a_1 z^{-1} + a_2 z^{-2}}\,, \quad \bar{n} = \frac{1}{\sqrt{2}\,\Omega_0}
\end{aligned}
\tag{6.20.10}
$$

and, for $M = 3$,

$$
\begin{aligned}
G &= \frac{\Omega_0^3}{(\Omega_0 + 1)\,(\Omega_0^2 + \Omega_0 + 1)}\,, \quad a_1 = \frac{(\Omega_0 - 1)\,(3\Omega_0^2 + 5\Omega_0 + 3)}{(\Omega_0 + 1)\,(\Omega_0^2 + \Omega_0 + 1)} \\
a_2 &= \frac{3\Omega_0^2 - 5\Omega_0 + 3}{\Omega_0^2 + \Omega_0 + 1}\,, \quad a_3 = \frac{(\Omega_0 - 1)\,(\Omega_0^2 - \Omega_0 + 1)}{(\Omega_0 + 1)\,(\Omega_0^2 + \Omega_0 + 1)} \\
H(z) &= \frac{G\,(1 + 3z^{-1} + 3z^{-2} + z^{-3})}{1 + a_1 z^{-1} + a_2 z^{-2} + a_3 z^{-3}}\,, \quad \bar{n} = \frac{1}{\Omega_0}
\end{aligned}
\tag{6.20.11}
$$

We note also that the $M = 1$ case has lag, $\bar{n} = 1/(2\Omega_0)$, and is equivalent to the modified EMA of Eq. (2.3.5). This can be seen by rewriting $H(z)$ in the form,

$$
H(z) = \frac{G_0\,(1 + z^{-1})}{1 + a_{01}z^{-1}} = \frac{\frac{1}{2}(1 - \lambda)\,(1 + z^{-1})}{1 - \lambda z^{-1}}\,, \quad \lambda = -a_{01} = \frac{1 - \Omega_0}{1 + \Omega_0}
$$

where $0 < \lambda < 1$ for $\Omega_0 < 1$, which requires $N > 4$.

The MATLAB function, **bma**, implements the design and filtering operations for any filter order M and any period $N > 2$,[†] with usage,

```
[y,nlag,b,a] = bma(x,N,M,yin);   % Butterworth moving average
[y,nlag,b,a] = bma(x,N,M);
```

where

```
x = input signal
N = 3-dB period, need not be integer, but N>2
M = filter order
yin = any Mx1 vector of initial values of the output y
      default, yin = repmat(x(1),M,1)
      yin = 'c' for standard convolutional output

y = output signal
nlag = filter lag
b = [b0, b1, b2, ..., bM], numerator filter coefficients
a = [ 1, a1, a2, ..., aM], denominator filter coefficients
```

[†] the sampling theorem requires, $f_0 < f_s/2$, or, $N = f_s/f_0 > 2$

Fig. 6.20.1 shows the BMA output for Butterworth orders $M = 2, 3$ applied to the same Nicor-Gas data of Fig. 6.18.2. It has noticeably shorter lag than SMA. The graphs were produced by the MATLAB code,

```
Y = xlsread('nicor.xls');       % load Nicor-Gas data
Y = Y(1:130,1:4);               % 130 trading days, and [O,H,L,C] prices
y = Y(:,4);                     % closing prices
t = 0:length(y)-1;              % trading days

N = 20;                         % period,    SMA lag = (N-1)/2 = 9.50
[y2,n2] = bma(y,N,2);           % order-2 Butterworth,  lag n2 = 4.46
[y3,n3] = bma(y,N,3);           % order-3 Butterworth,  lag n3 = 6.31

figure; plot(t,sma(y,N), t,y2, t,y3);     % plot SMA, y2, y3
        hold on; ohlc(t,Y,'color','b');   % add OHLC bar chart
```

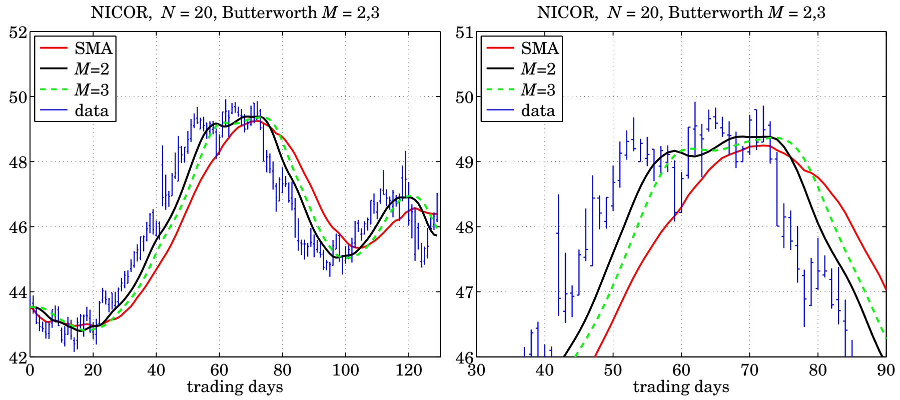


Fig. 6.20.1 Comparison of SMA and Butterworth filters of orders $M = 2, 3$.

6.21 Moving Average Filters with Reduced Lag

The PMA/linear regression and the DEMA/TEMA indicators have zero lag by design. There are other useful indicators that are easily implemented and have zero or very much reduced lag. Examples are twicing and Kaiser-Hamming (KH) filter sharpening [60], the Hull moving average (HMA) [309], the zero-lag EMA indicator (ZEMA) [284], the generalized DEMA (GDEMA) [307], and their variants. Here, we discuss a general procedure for constructing such reduced-lag filters, including the corresponding local-slope filters.

Consider three lowpass filters $H_1(z), H_2(z), H_3(z)$ with unity DC gains and lags, $\bar{n}_1, \bar{n}_2, \bar{n}_2$, respectively, and define the following filters for estimating the local level and local slope of the data, generalizing the twicing operations of Eq. (6.10.9),

$$\boxed{\begin{aligned} H_a(z) &= H_1(z)\big[(1+\nu)H_2(z)-\nu H_3(z)\big] = \text{local level} \\ H_b(z) &= \frac{1}{\bar{n}_3-\bar{n}_2}H_1(z)\big[H_2(z)-H_3(z)\big] = \text{local slope} \end{aligned}} \tag{6.21.1}$$

where ν is a positive constant. One may view $H_a(z)$ as the smoothed, by $H_1(z)$, version of $(1+\nu)H_2(z)-\nu H_3(z)$. The filter $H_a(z)$ will still have unity DC gain as follows by evaluating Eq. (6.21.1) at $z = 1$, that is,

$$H_a(1) = (1+\nu)H_1(1)H_2(1)-\nu H_1(1)H_3(1) = (1+\nu)-\nu = 1$$

Using the fact that the lag of a product of filters is the sum of the corresponding lags (cf. Problem 6.2), we find that the lag of $H_a(z)$ is,

$$\bar{n}_a = (1+\nu)(\bar{n}_1+\bar{n}_2)-\nu(\bar{n}_1+\bar{n}_3)\,, \quad \text{or,}$$

$$\boxed{\bar{n}_a = \bar{n}_1+(1+\nu)\bar{n}_2-\nu\bar{n}_3} \tag{6.21.2}$$

By appropriately choosing the parameters $\nu, \bar{n}_1, \bar{n}_2, \bar{n}_3$, the lag $\bar{n}_a$ can be made very small, even zero. Indeed, the following choice for ν will generate any particular $\bar{n}_a$,

$$\boxed{\nu = \frac{\bar{n}_1+\bar{n}_2-\bar{n}_a}{\bar{n}_3-\bar{n}_2}} \tag{6.21.3}$$

Below we list a number of examples that are special cases of the above constructions. In this list, the filter $H(z)$, whose lag is denoted by $\bar{n}$, represents any unity-gain lowpass filter, such as WMA, EMA, or SMA and similarly, $H_N(z)$ represents either a length-N FIR filter such as WMA or SMA, or an EMA filter with SMA-equivalent length N. Such filters have a lag related to N via a relationship of the form, $\bar{n} = r\cdot(N-1)$, for example, $r = 1/3$, for WMA, and, $r = 1/2$, for EMA and SMA.

	reduced-lag filters	lag
(twicing)	$H_a(z) = 2H(z)-H^2(z)\,,$	$\bar{n}_a = 0$
(GDEMA)	$H_a(z) = (1+\nu)H(z)-\nu H^2(z)\,,$	$\bar{n}_a = (1-\nu)\bar{n}$
(KH)	$H_a(z) = (1+\nu)H^2(z)-\nu H^3(z)\,,$	$\bar{n}_a = (2-\nu)\bar{n}$
(HMA)	$H_a(z) = H_{\sqrt{N}}(z)\big[2H_{N/2}(z)-H_N(z)\big]\,,$	$\bar{n}_a = r\big[\sqrt{N}-2\big]$
(ZEMA)	$H_a(z) = 2H(z)-z^{-d}H(z)\,,$	$\bar{n}_a = \bar{n}-d$
(ZEMA)	$H_a(z) = (1+\nu)H(z)-\nu z^{-d}H(z)\,,$	$\bar{n}_a = \bar{n}-\nu d$

(6.21.4)

The corresponding local-slope filters are as follows (they do not depend on ν),

$$\begin{array}{|ll|}\hline \multicolumn{2}{|c|}{\text{local-slope filters}} \\ \hline \text{(DEMA/GDEMA)} & H_b(z)=\dfrac{1}{\bar{n}}\big[H(z)-H^2(z)\big] \\ \text{(KH)} & H_b(z)=\dfrac{1}{\bar{n}}\big[H^2(z)-H^3(z)\big] \\ \text{(HMA)} & H_b(z)=\dfrac{2}{rN}H_{\sqrt{N}}(z)\big[H_{N/2}(z)-H_N(z)\big] \\ \text{(ZEMA)} & H_b(z)=\dfrac{1}{d}\big[H(z)-z^{-d}H(z)\big] \\ \hline \end{array} \qquad (6.21.5)$$

The standard twicing method, $H_a(z)=2H(z)-H^2(z)$, coincides with DEMA if we choose $H(z)$ to be a single EMA filter,

$$H_{\text{EMA}}(z)=\frac{\alpha}{1-\lambda z^{-1}}\,,\quad \alpha=1-\lambda\,,\quad \lambda=\frac{N-1}{N+1}\,,\quad \bar{n}=\frac{N-1}{2} \qquad (6.21.6)$$

but the filter $H(z)$ can also be chosen to be an SMA, WMA, or BMA filter, leading to what may be called, "double SMA," or, "double WMA,", or, 'double BMA."

The generalized DEMA, $H_{\text{GDEMA}}(z)=(1+v)H(z)-vH^2(z)$, also has, $H=H_{\text{EMA}}$, and is usually operated in practice with $v=0.7$. It reduces to standard DEMA for $v=1$. The so-called Tillson's *T3 indicator* [307] is obtained by cascading GDEMA three times,

$$H_{\text{T3}}(z)=\big[H_{\text{GDEMA}}(z)\big]^3 \qquad \text{(T3 indicator)} \qquad (6.21.7)$$

The Kaiser-Hamming (KH) filter sharpening case is not currently used as an indicator, but it has equally smooth output as GDEMA and T3. It reduces to the standard filter sharpening case with zero lag for $v=2$.

In the original Hull moving average [309], $H_a(z)=H_{\sqrt{N}}(z)\big[2H_{N/2}(z)-H_N(z)\big]$, the filter H_N is chosen to be a length-N weighted moving average (WMA) filter, as defined in Eqs. (6.15.1) and (6.15.2), and similarly, $H_{N/2}$ and $H_{\sqrt{N}}$ are WMA filters of lengths $N/2$ and $\sqrt{N}$ respectively. Assuming for the moment that these filter lengths are integers, then the corresponding lags of the three WMA filters, $H_{\sqrt{N}}, H_{N/2}, H_N$, will be,

$$\bar{n}_1=\frac{\sqrt{N}-1}{3}\,,\quad \bar{n}_2=\frac{N/2-1}{3}\,,\quad \bar{n}_3=\frac{N-1}{3}\,,$$

and setting $v=1$ in Eq. (6.21.2), we find,

$$\bar{n}_a=\bar{n}_1+2\bar{n}_2-\bar{n}_3=\frac{\sqrt{N}-1}{3}+\frac{N-2}{3}-\frac{N-1}{3}=\frac{\sqrt{N}-2}{3} \qquad (6.21.8)$$

Thus, for larger Ns, the lag is effectively reduced by a factor of $\sqrt{N}$. The extra filter factor $H_{\sqrt{N}}(z)$ provides some additional smoothing. In practice, the filter lengths $N_1=\sqrt{N}$ and $N_2=N/2$ are replaced by their rounded values. This changes the lag $\bar{n}_a$ somewhat. If one wishes to maintain the same lag as that given by Eq. (6.21.8), then one can compensate for the replacement of N_1, N_2 by their rounded values by using a slightly different value for v. It is straightforward to show that the following procedure will generate the desired lag value, where the required v is evaluated from Eq. (6.21.3),

$$\begin{aligned} N_1 &= \text{round}\big(\sqrt{N}\big)\,,\quad \epsilon_1=N_1-\sqrt{N}=\text{rounding error} \\ N_2 &= \text{round}\left(\frac{N}{2}\right)\,,\quad \epsilon_2=N_2-\frac{N}{2}=\text{rounding error} \\ v &= \frac{\bar{n}_1+\bar{n}_2-\bar{n}_a}{\bar{n}_3-\bar{n}_2}=\frac{N/2+\epsilon_1+\epsilon_2}{N/2-\epsilon_2} \\ \bar{n}_a &= \frac{N_1-1}{3}+(1+v)\frac{N_2-1}{3}-v\frac{N-1}{3}=\frac{\sqrt{N}-2}{3} \\ \bar{n}_3-\bar{n}_2 &= \frac{N-N_2}{3} \end{aligned} \qquad (6.21.9)$$

with transfer functions,

$$\begin{aligned} H_a(z) &= H_{N_1}(z)\big[(1+v)H_{N_2}(z)-vH_N(z)\big]=\text{local level} \\ H_b(z) &= \frac{1}{\bar{n}_3-\bar{n}_2}H_{N_1}(z)\big[H_{N_2}(z)-H_N(z)\big]=\text{local slope} \end{aligned} \qquad (6.21.10)$$

The WMA filters in the HMA indicator can be replaced with EMA filters resulting in the so-called "exponential Hull moving average" (EHMA), which has been found to be very competitive with other indicators [347]. Because N does not have to be an integer in EMA, it is not necessary to round the lengths $N_1=\sqrt{N}$ and $N_2=N/2$, and one can implement the indicator as follows, where H_N denotes the single EMA of Eq. (6.21.6),

$$\begin{aligned} &\bar{n}_a=\frac{\sqrt{N}-2}{2}\,,\quad \bar{n}_3-\bar{n}_2=\frac{N}{4} \\ &H_a(z)=H_{\sqrt{N}}(z)\big[2H_{N/2}(z)-H_N(z)\big] \\ &H_b(z)=\frac{4}{N}H_{\sqrt{N}}(z)\big[H_{N/2}(z)-H_N(z)\big] \end{aligned}$$

One can also replace the WMA filters by SMAs leading to the "simple Hull moving average" (SHMA). The filter H_N now stands for a length-N SMA filter, resulting in $\bar{n}_a=(\sqrt{N}-1)/2$, and $\bar{n}_3-\bar{n}_2=(N-N_2)/2$. Except for these changes, the computational procedure outlined in Eq. (6.21.9) remains the same.

The following MATLAB code illustrates the computation of the local-level output signal a_n from the data y_n, for the three versions of HMA and a given value of $N>1$,

```
N1 = round(sqrt(N));  e1 = N1 - sqrt(N);
N2 = round(N/2);      e2 = N2 - N/2;

v = (N/2 + e1 + e2) / (N/2 - e2);

a = wma((1+v)*wma(y,N2) - v*wma(y,N), N1);     % HMA
a = sma((1+v)*sma(y,N2) - v*sma(y,N), N1);     % SHMA
a = sema(2*sema(y,N/2) - sema(y,N), sqrt(N));  % EHMA
```

The functions, **hma**, **shma**, **ehma**, which are discussed below, implement these operations but offer more options, including the computation of the slope signals.

In the zero-lag EMA (ZEMA or ZLEMA) indicator [284], $H_a(z) = 2H(z) - z^{-d}H(z)$, the filter $H(z)$ is chosen to be a single EMA filter of the form of Eq. (6.21.6), and the delay d is chosen to coincide with the filter lag, that is, $d = \bar{n} = (N-1)/2$. It follows from Eq. (6.21.4) that the lag will be exactly zero, $\bar{n}_a = \bar{n} - d = 0$. This assumes that $\bar{n}$ is an integer, which happens only for odd N. For even N, the delay d may be chosen as the rounded-up version of $\bar{n}$, that is,

$$d = \text{round}(\bar{n}) = \text{round}\left(\frac{N-1}{2}\right) = \frac{N}{2}\,, \quad N = \text{even}$$

Then, the lag $\bar{n}_a$ can still be made to be zero by choosing the parameter v such that $\bar{n}_a = \bar{n} - vd = 0$, or, $v = \bar{n}/d = \bar{n}/\text{round}(\bar{n})$. Thus, the generalized form of the ZEMA indicator is constructed by,

$$\begin{aligned}
&\bar{n} = \frac{N-1}{2}\,, \quad d = \text{round}(\bar{n})\,, \quad v = \frac{\bar{n}}{d} \\
&H_a(z) = (1+v)H(z) - vz^{-d}H(z) \qquad\qquad (6.21.11) \\
&H_b(z) = \frac{1}{d}\big[H(z) - z^{-d}H(z)\big]
\end{aligned}$$

The code segment below illustrates the computation of the local-level ZEMA signal. It uses the function, **delay**, which implements the required delay.

```
nbar = (N-1)/2;
d = round(nbar);
v = nbar/d;
a = (1+v)*sema(y,N) - v*delay(sema(y,N), d);   % ZEMA
```

The following MATLAB functions implement the reduced-lag filters discussed above, where the input array **y** represents the financial data to be filtered, and the outputs **a**, **b** represent the local-level and local-slope signals.

```
[a,b] = hma(y,N,yin);       % Hull moving average
[a,b] = ehma(y,N,yin);      % exponential Hull moving average
[a,b] = shma(y,N,yin);      % simple Hull moving average
[a,b] = zema(y,N,yin);      % zero-lag EMA

y = delay(x,d);             % d-fold delay, y(n) = x(n-d)
a = gdema(y,N,v,yin);       % generalized DEMA
a = t3(y,N,v,yin);          % Tillson's T3
```

The input variable `yin` defines the initialization and defaults to progressive filtering for **hma**, **shma**, and **zema**, `yin='f'`, and to, `yin` $= y_0$, for **ehma**.

Fig. 6.21.1 compares the PMA/linear regression indicator with HMA, EHMA, and ZEMA on the same Nicor-Gas data, with filter length $N = 20$. Fig. 6.21.2 compares the corresponding slope indicators. The MATLAB code below illustrates the computation.

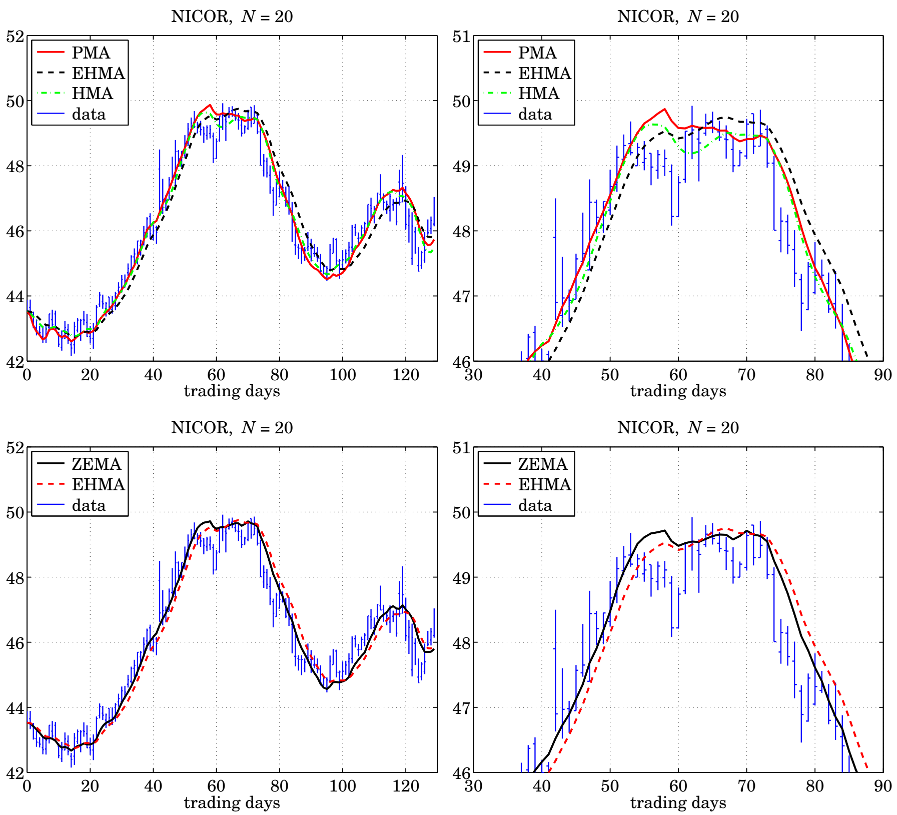


Fig. 6.21.1 Comparison of PMA/LREG, HMA, EHMA, and ZEMA indicators.

```
Y = xlsread('nicor.xls');     % load Nicor-Gas data
Y = Y(1:130,1:4);             % keep 130 trading days, and [O,H,L,C] prices
y = Y(:,4);                   % closing prices
t = 0:length(y)-1;            % trading days

N = 20;                                   % filter length

[al,bl] = lreg(y,N);                      % PMA/LREG
[ah,bh] = hma(y,N);                       % HMA
[ae,be] = ehma(y,N);                      % EHMA
[az,bz] = zema(y,N);                      % ZEMA

figure; plot(t,al, t,ae, t,ah);           % PMA/LREG, EHMA, HMA
hold on; ohlc(t,Y);                       % add OHLC chart

figure; plot(t,az, t,ae);                 % ZEMA, EHMA
hold on; ohlc(t,Y);                       % add OHLC chart
```

```
figure; plot(t,bh, t,be); hold on;     % HMA, EHMA slopes
stem(t,bl,'marker','none');            % plot LREG slope as stem

figure; plot(t,bh, t,bz); hold on;     % HMA, ZEMA slopes
stem(t,bl,'marker','none');
```

We note that the reduced-lag HMA, EHMA, and ZEMA local-level and local-slope filters have comparable performance as the PMA/linear regression and DEMA/TEMA filters, with a comparable degree of smoothness.

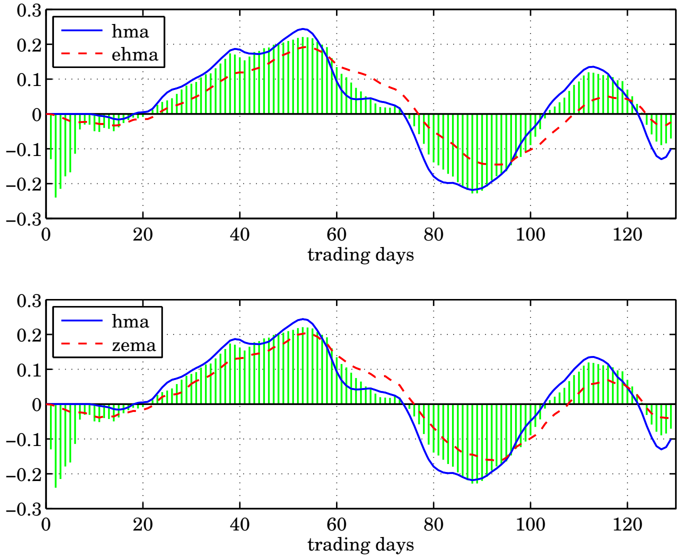


Fig. 6.21.2 Slope indicators, linear regression (stem) vs. HMA, EHMA, ZEMA.

6.22 Envelopes, Bands, and Channels

Moving averages help market traders discern trends by smoothing out variations in the data. However, such variations can provide additional useful information, such as gauging volatility, or, identifying extremes in the data that provide trading opportunities, or observing how prices settle into their trends.

Trading envelopes or bands or channels consist of two curves drawn above and below a moving average trendline. The two bounds define a zone of variation, or volatility, about the average, within which most of the price fluctuations are expected to lie.

The typical trading rule is that when a price closes near or above the upper bound, it signals that the stock is overbought and suggests trading in the opposite direction. Similarly, if a price moves below the lower bound it signals that the stock is oversold and suggests an opposite reaction.

In this section we discuss the following types of bands and their computation,

- Bollinger bands
- Projection bands
- Fixed-width bands
- Starc bands
- Standard-error bands
- Donchian channels
- Keltner bands
- Parabolic SAR

Examples of these are shown in Figs. 6.22.1 and 6.22.2 applied to the same Nicor data that we used previously. Below we give brief descriptions of how such bands are computed. We use our previously discussed MATLAB functions, such as SMA, LREG, etc., to summarize the computations. Further references are given in [323–334].

Bollinger Bands

Bollinger bands [323–327] are defined relative to an N-day SMA of the closing prices, where typically, $N = 14$. The two bands are taken to be two standard deviations above and below the SMA. The following MATLAB code clarifies the computation,

```
M = sma(y,N);                     % N-point SMA of closing prices y
S = stdev(y,N);                   % std-dev relative to M
L = M - 2*S;                      % lower bound
U = M + 2*S;                      % upper bound
```

where the function, **stdev**, uses the built-in function **std** to calculate the standard deviation over each length-N data window, and its essential code is,

```
for n=1:length(y),
   S(n) = std(y(max(1,n-N+1):n));
end
```

where the data window length is N for $n \ge N$, and n during the initial transients $n < N$.

Standard-Error Bands

Standard-error bands [329] use the PMA/linear-regression moving average as the middle trendline and shift it by two standard errors up and down. The calculation is summarized below with the help of the function **lreg**, in which the quantities, **y, a, se**, represent the closing prices, the local level, and the standard error,

```
[a,~,~,se] = lreg(y,N);    % N-point linear regression
L = a - 2*se;              % lower bound
U = a + 2*se;              % upper bound
```

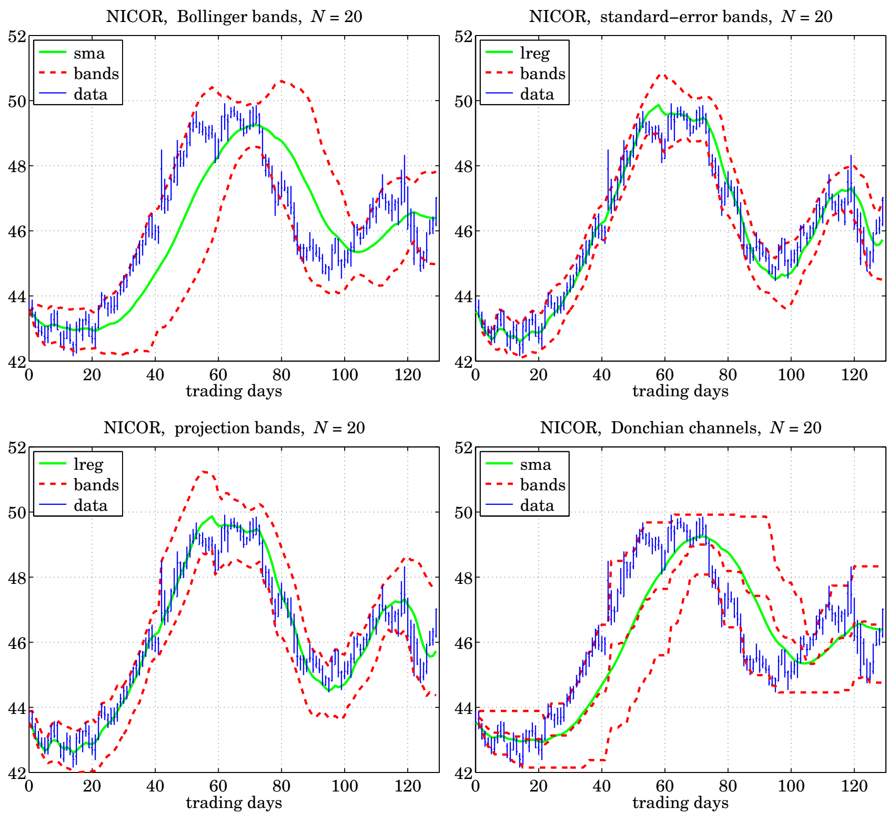


Fig. 6.22.1 Bollinger bands, standard-error bands, projection bands, and Donchian channels.

Projection Bands

Projection bands [328] also use the linear regression function **lreg** to calculate the local slopes for the high and low prices, H, L. The value of the upper (lower) band at the n-th time instant is determined by considering the values of the highs H (lows L) over the look-back period, $n - N + 1 \le t \le n$, extrapolating each of them linearly according to their slope to the current time instant n, and taking the maximum (minimum) among them. The following MATLAB code implements the procedure,

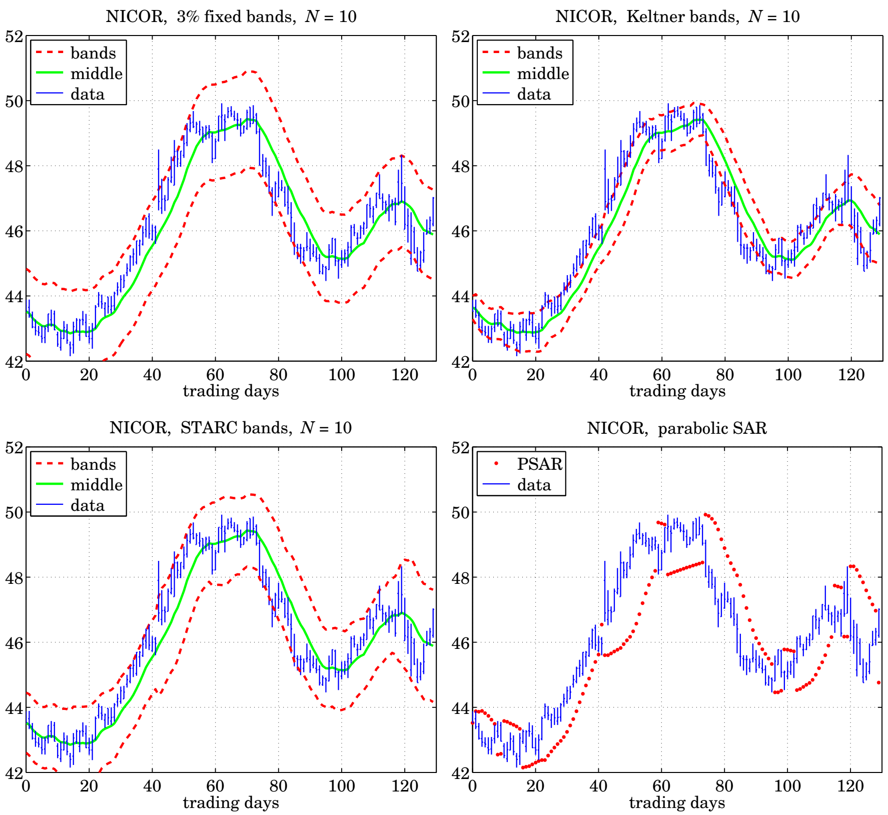


Fig. 6.22.2 Fixed-width bands, Keltner bands, STARC bands, and parabolic SAR.

```
[~,bL] = lreg(L,N);     % linear regression slope for Low
[~,bH] = lreg(H,N);     % linear regression slope for High
for n=0:length(H)-1,
   t = (max(0,n-N+1) : n)';            % look-back interval
   Lo(n+1) = min(L(t+1) + bL(n+1)*(n-t));    % lower band
   Up(n+1) = max(H(t+1) + bH(n+1)*(n-t));    % upper band
end
```

Donchian Channels

Donchian channels [331] are constructed by finding, at each time instant n, the highest high (resp. lowest low) over the past time interval, $n - N \le t \le n - 1$, that is, the value of the upper bound at the n-th day, is the maximum of the highs over the previous N

days, not including the current day, i.e., $\max[H_{n-1}, H_{n-2}, \dots, H_{n-N}]$. The code below describes the procedure,

```
for n = 2:length(H)         % n is MATLAB index
   t = max(1,n-N) : n-1;    % past N days
   Lo(n) = min(L(t));       % lower band
   Up(n) = max(H(t));       % upper band
end
Mid = (Up + Lo)/2;          % middle band
```

Fixed-Width Bands

Fixed-width bands or envelopes [330] shift an N-point SMA of the closing prices by a certain percentage, such as, typically, 3 percent,

```
M = sma(C,N);       % N-point SMA of closing prices C
L = M - p*M;        % lower band, e.g., p = 0.03
U = M + p*M;        % upper band
```

Keltner Bands

Keltner bands or channels [330], use as the middle trendline an N-point SMA of the average of the high, low, and closing prices, $(H+L+C)/3$, and use an N-point SMA of the difference $(H-L)$ as the bandwidth, representing a measure of volatility. The code below implements the operations,

```
M = sma((H+L+C)/3,N);    % SMA of (H+L+C)/3
D = sma(H-L,N);          % SMA of (H-L)
L = M - D;               % lower band
U = M + D;               % upper band
```

The typical value of N is 10, and the trading rule is that a “buy” signal is generated when the closing price C lies above the upper band, and a “sell” signal when C lies below the lower band.

Starc Bands

In Starc† bands [330] all three prices, high, low, and closing, H, L, C, are used. The middle band is an N-point SMA of the closing prices C, but the bandwidth is defined in terms of an N_a-point of the so-called “average true range” (ATR), which represents another measure of volatility. The code below describes the computation,

```
M = sma(C,N);          % SMA of closing prices
R = atr([H,L,C],Na);   % ATR = average true range
L = M - 2*R;           % lower and
U = M + 2*R;           % upper band
```

† Stoller Average Range Channels

The ATR [281] is an N_a-point WEMA of the “true range”, defined as follows, at each time n,

$$\begin{aligned} T_n &= \max\big[H_n - L_n,\, H_n - C_{n-1},\, C_{n-1} - L_n\big] = \text{true range in } n\text{-th day} \\ R_n &= \text{wema}(T_n, N_a) = \text{ATR} \end{aligned} \tag{6.22.1}$$

and is implemented by the function **atr**, with the help of the **delay** function. Its essential MATLAB code is as follows, where `H,L,C` are column vectors,

```
T = max([H-L, H-delay(C,1), delay(C,1)-L], [], 2);      % row-wise max
R = wema(T,N);
```

MATLAB Functions

The following MATLAB functions implement the band indicators discussed above, where the various parameters are fully explained in the help files for these functions,

```
[L,U,M] = bbands(y,N,d);            % Bollinger bands
[L,U,a] = sebands(y,N,d,init);      % standard-error bands
[L,U,R,Rs] = pbands(Y,N,Ns);        % projection bands & oscillator
[L,U,M] = donch(Y,N);               % Donchian channels
[L,U,M] = fbands(Y,N,p);            % fixed-width bands
[L,U,M] = kbands(Y,N);              % Keltner bands
[L,U,M] = stbands(Y,N,Na);          % Starc bands

S = stdev(y,N,flag);                % standard deviation
[R,TR] = atr(Y,N);                  % average true range
```

The essential MATLAB code for generating Figs. 6.22.1 and 6.22.2 is as follows,

```
Y = xlsread(’nicor.xls’);   % load Nicor-Gas data
Y = Y(1:130,1:4);           % 130 trading days, and [O,H,L,C] prices
y = Y(:,4);                 % closing prices
t = 0:length(y)-1;          % trading days

N = 20;                                    % used in Fig.6.22.1

[L,U,M] = bbands(y,N);                     % Bollinger
figure; ohlc(t,Y); hold on;                % make OHLC bar chart
plot(t,M,’g-’, t,U,’r--’, t,L,’r--’);

[L,U,a] = sebands(y,N);                    % standard-error
figure; ohlc(t,Y); hold on;
plot(t,a,’g-’, t,U,’r--’, t,L,’r--’);

a = lreg(y,N);
[L,U] = pbands(Y,N);                       % projection
figure; ohlc(t,Y); hold on;
plot(t,a,’g-’, t,U,’r--’, t,L,’r--’);

[L,U,M] = donch(Y,N);                      % Donchian
figure; ohlc(t,Y); hold on;
plot(t,M,’r--’, t,L,’r--’, t,U,’r--’);
plot(t,sma(y,N),’g-’);
```

```
N=10;                                      % used in Fig.6.22.2

p=0.03;
[L,U,M] = fbands(Y,N,p);                   % fixed-width
figure; ohlc(t,Y); hold on;
plot(t,M,'g-', t,U,'r--', t,L,'r--');

[L,U,M] = kbands(Y,N);                     % Keltner
figure; ohlc(t,Y); hold on;
plot(t,M,'g-', t,U,'r--', t,L,'r--');

[L,U,M] = stbands(Y,N);                    % Starc
figure; ohlc(t,Y); hold on;
plot(t,M,'g-', t,U,'r--', t,L,'r--');

H = Y(:,2); L = Y(:,3); ni=1; Ri=1;        % parabolic SAR
S = psar(H,L,Ri,ni);
figure; ohlc(t,Y); hold on;
plot(t,S,'r.');
```

Parabolic SAR

Wilder's parabolic stop & reverse (SAR) [281] is a trend-following indicator that helps a trader to switch positions from long to short or vice versa.

While holding a long position during a period of increasing prices, the SAR indicator lies *below* the prices and is also increasing. When the prices begin to fall and touch the SAR from above, then a "sell" signal is triggered with a reversal of position from long to short, and the SAR switches sides and begins to fall, lying *above* the falling prices. If subsequently, the prices begin to rise again and touch the SAR from below, then a "buy" signal is triggered and the position is reversed from short to long again, and so on.

The indicator is very useful as it keeps a trader constantly in the market and works well during trending markets with steady periods of increasing or decreasing prices, even though it tends to recommend buying relatively high and selling relatively low—the opposite of what is the ideal. It does not work as well during "sideways" or trading markets causing so-called "whipsaws." It is usually used in conjunction with other indicators that confirm trend, such as the RSI or DMI. Some further references on the SAR are [335–340].

The SAR is computed in terms of the high and low price signals H_n, L_n and is defined as the exponential moving average of the *extreme price* reached within each trending period, but it uses a time-varying EMA parameter, $\lambda_n = 1 - \alpha_n$, as well as additional conditions that enable the reversals. Its basic EMA recursion from day n to day $n+1$ is,

$$S_{n+1} = \lambda_n S_n + \alpha_n E_n = (1 - \alpha_n) S_n + \alpha_n E_n\,, \quad \text{or,}$$

$$\boxed{S_{n+1} = S_n + \alpha_n (E_n - S_n)} \quad \text{(SAR)} \tag{6.22.2}$$

where E_n is the extreme price reached during the current trending position, that is, the highest high reached up to day n during an up-trending period, or the lowest low up to day n during a down-trending period. At the beginning of each trending period, S_n is initialized to be the extreme price of the previous trending period.

The EMA factor α_n increases linearly with time, starting with an initial value, α_i, at the beginning of each trending period, and then increasing by a fixed increment $\Delta\alpha$, but only every time a *new* extreme value is reached, that is,

$$\alpha_{n+1} = \begin{cases} \alpha_n + \Delta\alpha\,, & \text{if } \ E_{n+1} \neq E_n \\ \alpha_n\,, & \text{if } \ E_{n+1} = E_n \end{cases} \tag{6.22.3}$$

where we note that $E_{n+1} \neq E_n$ happens when E_{n+1} is strictly greater than E_n during an up-trend, or, E_{n+1} is strictly less than E_n during a down-trend. Moreover, an additional constraint is that α_n is not allowed to exceed a certain maximum value, α_m. The values recommended by Wilder [281] are,

$$\alpha_i = 0.02\,, \quad \Delta\alpha = 0.02\,, \quad \alpha_m = 0.2$$

Because of the increasing α_n parameter, the EMA has a time-varying decreasing lag,[†] thus, tracking more quickly the extreme prices as time goes by. As a result, S_n has a particular curved shape that resembles a parabola, hence the name "parabolic" SAR.

The essential steps in the calculation of the SAR are summarized in the following MATLAB code, in which the inputs are the quantities, H,L, representing the high and low price signals, H_n, L_n, while the output quantities, S,E,a,R, represent, S_n, E_n, α_n, R_n, where R_n holds the current position and is equal to ±1 for long/short.

```
Hi = max(H(1:ni));          % initial highest high, default
Li = min(L(1:ni));          % initial lowest low

R(ni) = Ri;                 % initialize outputs at starting time n=ni
a(ni) = ai;
S(ni) = Li*(Ri==1) + Hi*(Ri==-1);
E(ni) = Hi*(Ri==1) + Li*(Ri==-1);

for n = ni : length(H)-1

   S(n+1) = S(n) + a(n) * (E(n) - S(n));                    % SAR update

   r = R(n);                                          % current position

   if (r==1 & L(n+1)<=S(n+1)) | (r==-1 & H(n+1)>=S(n+1))    % reversal
      r = -r;                                               % reverse r
      S(n+1) = E(n);                                        % reset new S
      E(n+1) = H(n+1)*(r==1) + L(n+1)*(r==-1);              % reset new E
      a(n+1) = ai;                                          % reset new a
   else                                                     % no reversal
      if n>2                                                % new S
        S(n+1) = min([S(n+1), L(n-1), L(n)])*(r==1) ...     % additional
               + max([S(n+1), H(n-1), H(n)])*(r==-1);       % conditions
      end

      E(n+1) = max(E(n),H(n+1))*(r==1) ...                  % new E
             + min(E(n),L(n+1))*(r==-1);

      a(n+1) = min(a(n) + (E(n+1)~=E(n)) * Da, am);         % new a
   end

   R(n+1) = r;                                              % new R

end  % for-loop
```

[†] The EMA equivalent length decreases from, $N_i = 2/\alpha_i - 1 = 99$, down to, $N_m = 2/\alpha_m - 1 = 9$.

If the current trading position is long ($r = 1$), corresponding to an up-trending market, then, a reversal of position to short ($r = -1$) will take place at time $n+1$ if the low price L_{n+1} touches or becomes less than the SAR, that is, if, $L_{n+1} \le S_{n+1}$. Similarly, if the current position is short, corresponding to a down-trending market, then, a reversal of position to long will take place at time $n+1$ if the high price H_{n+1} touches or becomes greater than the SAR, that is, if, $H_{n+1} \ge S_{n+1}$. At such reversal time points, the SAR is reset to be equal to the extreme price of the previous trend, that is, $S_{n+1} = E_n$, and the E_{n+1} is reset to be either L_{n+1} if reversing to short, or H_{n+1} if reversing to long, and the EMA parameter is reset to, $\alpha_{n+1} = \alpha_i$.

An additional condition is that during an up-trend, the SAR for tomorrow, S_{n+1}, is not allowed to become greater that either today's or yesterday's lows, L_n, L_{n-1}, and in such case it is reset to the minimum of the two lows. Similarly, during a down-trend, the S_{n+1} is not allowed to become less that either today's or yesterday's highs, H_n, H_{n-1}, and is reset to the maximum of the two highs. This is enforced by the code line,

$$S_{n+1} = \min\big([S_{n+1}, L_{n-1}, L_n]\big)\cdot(r{=}{=}1) + \max\big([S_{n+1}, H_{n-1}, H_n]\big)\cdot(r{=}{=}{-1})$$

The parabolic SAR is implemented with the MATLAB function **psar**, with usage,

```
[S,E,a,R] = psar(H,L,Ri,ni,af,Hi,Li);     % parabolic SAR
```

```
H  = vector of High prices, column
L  = vector of Low prices, column, same length as H
Ri = starting position, long Ri = 1, short Ri = -1
ni = starting time index, default ni = 1,  all outputs are NaNs for n<ni
af = [ai,da,am] = [initial EMA factor, increment, maximum factor]
     default, af = [0.02, 0.02, 0.2]
Hi,Li = initial high and low used to initialize S(n),E(n) at n=ni,
        default, Hi = max(H(1:ni)), Li = min(L(1:ni))

S = parabolic SAR, same size as H
E = extremal price, same size as H
a = vector of EMA factors, same size as H
R = vector of positions, R = +1/-1 for long/short, same size as H
```

The SAR signal S_n is usually plotted with dots, as shown for example, in the bottom right graph of Fig. 6.22.2. Fig. 6.22.3 shows two more examples.

The left graph reproduces Wilder's original example [281] and was generated by the following MATLAB code,

```
Y = xlsread('psarexa.xls');               % data from Wilder [281]
t = Y(:,1); H = Y(:,2); L = Y(:,3);       % extract H,L signals

Ri = 1; ni = 4;                           % initialize SAR
[S,E,a,R] = psar(H,L,Ri,ni);

num2str([t, H, L, a, E, S, R], '%8.2f');  % reproduces table on p.13 of [281]

figure; ohlc(t,[H,L], t,S,'r.');          % make OHLC plot, including SAR
```

The right graph is from Achelis [280]. The SAR is plotted with filled dots, but at the end of each trending period and shown with open circles are the points that triggered the reversals. The MATLAB code for this example is similar to the above,

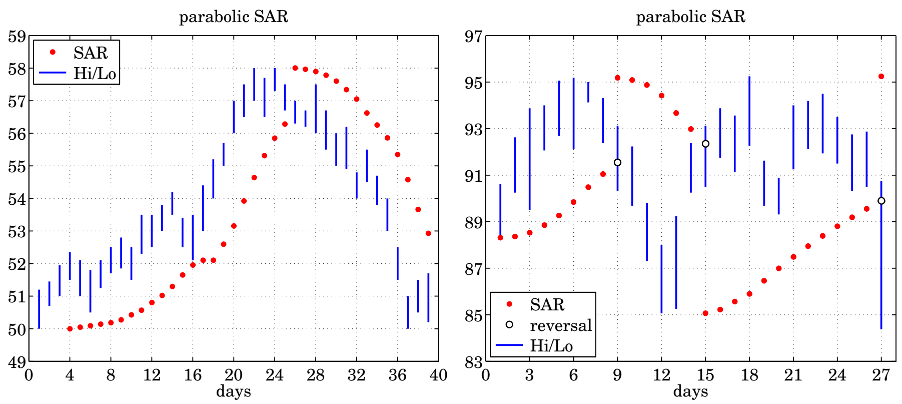


Fig. 6.22.3 Parabolic SAR examples from Wilder [281] and Achelis [280].

```
Y = xlsread('psarexb.xls');                 % data from Ref.[208]
t = Y(:,1); H = Y(:,2); L = Y(:,3);

Ri = 1; ni = 1;                             % initialize
[S,E,a,R] = psar(H,L,Ri,ni);                % compute SAR

num2str([t ,H, L, a, E, S, R], '%9.4f');    % reproduces table from Ref.[280]
figure; ohlc(t,[H,L], t,S,'r.');            % make OHLC plot, including SAR
```

The first up-trending period ends at day $n = 9$ at which the would be SAR, shown as an opened-circle, lies above the low of that day, thus, causing a reversal to short and that SAR is then replaced with the filled-circle value that lies above the highs, and corresponds to the highest high of the previous period that had occurred on day $n = 6$.

The second down-trending period ends at $n = 15$ at which point the SAR, shown as an opened-circle, is breached by the high on that day, thus causing a reversal to long, and the SAR is reset to the filled-circle value on that day lying below the data, and corresponds to the lowest low during the previous period that had been reached on day $n = 12$. Finally, another such reversal takes place on day $n = 27$ and the SAR is reset to the highest high that had occurred on day $n = 18$. To clarify, we list below the values of the SAR at the reversal points before and after the reversal takes place,

n	$S_{\text{before}}(n)$	$S_{\text{after}}(n)$
9	91.5448	$95.1875 = H_6$
15	92.3492	$85.0625 = L_{12}$
27	89.8936	$95.2500 = H_{18}$

6.23 Momentum, Oscillators, and Other Indicators

There exist several other indicators that are used in technical analysis, many of them built on those we already discussed. The following MATLAB functions implement some of the more popular ones, several are also included in MATLAB's financial toolbox. Additional references can be found in the Achelis book [280] and in [281–347].

```
R = rsi(y,N,type);                  % relative strength index, RSI
R = cmo(y,N);                       % Chande momentum oscillator, CMO
R = vhfilt(y,N);                    % Vertical Horizontal Filter, VHF
[Dp,Dm,DX,ADX] = dirmov(Y,N);       % directional movement system, +-DI,DX,ADX
------------------------------------------------------------------------------
[y,yr,ypr] = mom(x,d,xin);          % momentum and price rate of change
[y,ys,ypr] = prosc(x,N1,N2,N3);     % price oscillator & MACD
[pK,pD] = stoch(Y,K,Ks,D,M);        % stochastic, %K, %D oscillators
------------------------------------------------------------------------------
R = accdist(Y);                     % accumulation/distribution line
R = chosc(Y,N1,N2);                 % Chaikin oscillator
R = cmflow(Y,N);                    % Chaikin money flow
R = chvol(Y,N);                     % Chaikin volatility
------------------------------------------------------------------------------
[P,N] = pnvi(Y,P0);                 % positive/negative volume indices, PVI/NVI
R = cci(Y,N);                       % commodity channel index, CCI
R = dpo(Y,N);                       % detrended price oscillator, DPO
[R,N] = dmi(y,Nr,Ns,Nm);            % dynamic momentum index, DMI
[R,Rs] = forosc(y,N,Ns);            % forecast oscillator
------------------------------------------------------------------------------
[R,Rs] = trix(y,N,Ns,yin);          % TRIX oscillator
a = vema(y,N,Nv);                   % variable-length EMA
```

Below we discuss briefly their construction. Examples of their use are included in their respective help files. Several online examples can be found in the Fidelity Guide [345] and in the TradingView Wiki [346].

Relative Strength Index, RSI

The relative strength index (RSI) was introduced by Wilder [281] to be used in conjunction with the parabolic SAR to confirm price trends. It is computed as follows, where $\mathbf{y}$ is the column vector of daily closing prices,

```
x = diff(y);                % price differences
xu = +x.*(x>0);             % upward differences
xd = -x.*(x<=0);            % downward differences
su = wema(xu,N);            % smoothed differences
sd = wema(xd,N);

RSI = 100*su/(su+sd);       % RSI
```

Chande Momentum Oscillator, CMO

If the **wema** function is replaced by **sma**, one obtains the Chande momentum oscillator,

```
x = diff(y);                % price differences
xu = +x.*(x>0);             % upward differences
xd = -x.*(x<=0);            % downward differences
su = sma(xu,N);             % smoothed differences
sd = sma(xd,N);

CMO = 100*(su-sd)/(su+sd);  % CMO
```

Thus, the SMA-based RSI is related to CMO via,

$$\text{CMO} = 2\,\text{RSI} - 100 \quad\Leftrightarrow\quad \text{RSI} = \frac{\text{CMO} + 100}{2}$$

Vertical Horizontal Filter, VHF

The vertical horizontal filter (VHF) is similar to the RSI or CMO and helps to confirm a trend in trending markets. It is computed as follows, where the first N outputs are NaNs,

```
x = [NaN; diff(y)];            % y = column of closing prices
                               % x = price differences

for n=N+1:length(y),
   yn = y(n-N+1:n);            % length-N look-back window
   xn = x(n-N+1:n);
   R(n) = abs(max(yn)-min(yn)) / sum(abs(xn));     % VHF
end
```

Directional Movement System

The directional movement system was also proposed by Wilder [281] and consists of several indicators, the plus/minus directional indicators, (±DI), the directional index (DX), and the average directional index (ADX). These are computed as follows,

```
R = atr(Y,N);                        % average true range

DH = [0;  diff(H)];                  % high price differences
DL = [0; -diff(L)];                  % low price differences

Dp = DH .* (DH>DL) .* (DH>0);        % daily directional movements
Dm = DL .* (DL>DH) .* (DL>0);        %

Dp = wema(Dp,N);                     % averaged directional movements
Dm = wema(Dm,N);                     %

Dp = 100 * Dp ./ R;                  % +DI,-DI directional indicators
Dm = 100 * Dm ./ R;
DX = 100*abs(Dp - Dm)./(Dp + Dm);    % directional index, DI
ADX = wema(DX,N);                    % average directional index, ADX
```

Momentum and Price Rate of Change

In its simplest form a momentum indicator is the difference between a price today, $x(n)$, and the price d days ago, $x(n-d)$, but it can also be expressed as a ratio, or as a percentage, referred to as *price rate of change*,

$$
\begin{aligned}
y(n) &= x(n) - x(n-d) = \text{momentum} \\
y_r(n) &= 100 \cdot \frac{x(n)}{x(n-d)} = \text{momentum as ratio} \\
y_p(n) &= 100 \cdot \frac{x(n) - x(n-d)}{x(n-d)} = \text{price rate of change}
\end{aligned}
$$

It can be implemented simply with the help of the function, **delay**,

```
y   = x - delay(x,d);
yr  = x/delay(x,d) * 100;
yp = (x-delay(x,d))/delay(x,d) * 100;
```

Price Oscillator and MACD

The standard moving average convergence/divergence (MACD) indicator is defined as the difference between two EMAs of the daily closing prices: a length-12 shorter/faster EMA and a length-26 longer/slower EMA. A length-9 EMA of the MACD difference is also computed as a trigger signal.

Typically, a buy (sell) signal is indicated when the MACD rises above (falls below) zero, or when it rises above (falls below) its smoothed signal line.

The MACD can also be represented as a percentage resulting into the price oscillator, and also, different EMA lengths can be used. The following code segment illustrates the computation, where **x** are the closing prices,

```
y1 = sema(x,N1);      % fast EMA, default N1=12
y2 = sema(x,N2);      % slow EMA, default N2=26
y = y1 - y2;          % MACD
ys = sema(y,N3);      % smoothed MACD signal, default N3=9
ypr = 100 * y./y2;    % price oscillator
```

Stochastic Oscillator

```
H = Y(:,1); L = Y(:,2); C = Y(:,3);        % extract H,L,C inputs
Lmin = NaN(size(C)); Hmax = NaN(size(C));  % NaNs for n<K
for n = K:length(C),                       % look-back period K
   Lmin(n) = min(L(n-K+1:n));              % begins at n=K
   Hmax(n) = max(H(n-K+1:n));
end
pK = 100 * sma(C-Lmin, Ks) ./ sma(Hmax-Lmin, Ks);   % percent-K
pD = sma(Pk, D);                                     % percent-D
```

Fast Stochastic has $K_s = 1$, i.e., no smoothing, and *Slow Stochastic* has, typically, $K_s = 3$.

Accumulation/Distribution

```
H=Y(:,1); L=Y(:,2); C=Y(:,3); V=Y(:,4);    % extract H,L,C,V
R = cumsum((2*C-H-L)./(H-L).*V);           % ACCDIST
```

Chaikin Oscillator

```
y = accdist(Y);                  % Y = [H,L,C,V] data matrix
R = sema(y,N1) - sema(y,N2);     % CHOSC, default, N1=3, N2=10
```

Chaikin Money Flow

```
H=Y(:,1); L=Y(:,2); C=Y(:,3); V=Y(:,4);        % extract H,L,C,V
R = sma((2*C-H-L)./(H-L).*V, N) ./ sma(V,N);   % CMFLOW
```

Chaikin Volatility

```
S = sema(H-L,N);                            % H,L given
R = (S - delay(S,N)) ./ delay(S,N) * 100;   % volatility
```

Positive/Negative Volume Indices, PNVI

These are defined recursively as follows, in MATLAB-like notation, where C_n, V_n are the closing prices and the volume,

$$
\begin{aligned}
P_n &= P_{n-1} + (V_n > V_{n-1}) \cdot \frac{C_n - C_{n-1}}{C_{n-1}} \cdot P_{n-1} = P_{n-1} \left(\frac{C_n}{C_{n-1}}\right)^{(V_n > V_{n-1})} = \text{PVI} \\
N_n &= N_{n-1} + (V_n < V_{n-1}) \cdot \frac{C_n - C_{n-1}}{C_{n-1}} \cdot N_{n-1} = N_{n-1} \left(\frac{C_n}{C_{n-1}}\right)^{(V_n < V_{n-1})} = \text{NVI}
\end{aligned}
$$

and initialized to some arbitrary initial value, such as, $P_0 = N_0 = 1000$. The MATLAB implementation uses the function, **delay**,

```
P = P0 * cumprod( (C./delay(C,1)) .^ (V>delay(V,1)) );   % PNVI
N = P0 * cumprod( (C./delay(C,1)) .^ (V<delay(V,1)) );
```

Commodity Channel Index, CCI

```
T = (H+L+C)/3;                               % H,L,C given
M = sma(T,N);
for n=N+1:length(C),
   D(n) = mean(abs(T(n-N+1:n) - M(n)));     % mean deviation
end
R = (T-M)./D/0.015;                          % CCI
```

Detrended Price Oscillator, DPO

```
S = sma(y,N,'n');           % y = column of closing prices
M = floor(N/2) + 1;         % advancing time
R = y - delay(S,-M,'n');    % DPO, i.e., R(n) = y(n) - S(n+M)
```

Dynamic Momentum Index, DMI

```
x = [NaN; diff(y)];         % y = column of closing prices
xu =  x .* (x>0);           % updward differences
xd = -x .* (x<=0);          % downward differences
S = stdev(y,Ns);            % Ns-period stdev
V = S ./ sma(S,Nm);         % volatility measure
N = floor(Nr ./ V);         % effective length
N(N>Nmax) = Nmax;           % restrict max and min N
N(N<Nmin) = Nmin;
Nstart = Nr + Ns + Nm;
su1 = mean(xu(2:Nstart));       % initialize at start time
sd1 = mean(xd(2:Nstart));
switch lower(type)
  case 'wema'                   % WEMA type
    for n = Nstart+1:length(y),
       su(n) = su1 + (xu(n) - su1) / N(n); su1 = su(n);
       sd(n) = sd1 + (xd(n) - sd1) / N(n); sd1 = sd(n);
    end
  case 'sma'                    % SMA type
    for n = Nstart+1:length(y),
       su(n) = mean(xu(n-N(n)+1:n));
       sd(n) = mean(xd(n-N(n)+1:n));
    end
end
R = 100 * su./(su+sd);          % DMI
```

Forecast Oscillator

```
yp = pma(y,N,1);            % time series forecast
x = y - delay(yp,1);
R = 100 * x./y;             % forecast oscillator
Rs = sma(R,Ns);             % trigger signal
```

TRIX Oscillator

```
[~,~,~,~,~,a3] = tema(y,N,cin);     % triple EMA
R = 100*(a3 - delay(a3,1))./a3;     % TRIX
Rs = sma(R,Ns);                     % smoothed TRIX
```

Variable-Length EMA

```
la = (N-1)/(N+1); al = 1-la;          % EMA parameter
switch lower(type)
  case 'cmo'                          % CMO volatility
    V = abs(cmo(y,Nv))/100;
  case 'r2'                           % R^2 volatility
    [~,~,V] = lreg(y,Nv);
end
for n=Nv+1:length(y),                 % default si=y(Nv)
   s(n) = si + al*V(n)*(y(n)-si);     % EMA recursion
   si = s(n);
end
```

6.24 MATLAB Functions

We summarize the MATLAB functions discussed in this chapter:

```
% ------------------------------------
% Exponential Moving Average Functions
% ------------------------------------
% ema       - exponential moving average - exact version
% stema     - steady-state exponential moving average
% lpbasis   - fit order-d polynomial to first L inputs
% emap      - map equivalent lambdas's between d=0 and d=1 EMAs
% emaerr    - MSE, MAE, MAPE error criteria
% emat      - transformation matrix from polynomial to cascaded basis
% mema      - multiple exponential moving average
% holt      - Holt's exponential smoothing
% holterr   - MSE, MAE, MAPE error criteria for Holt
```

The technical analysis functions are:

```
% ----------------------------
% Technical Analysis Functions
% ----------------------------
% accdist   - accumulation/distribution line
% atr       - true range & average true range
% cci       - commodity channel index
% chosc     - Chaikin oscillator
% cmflow    - Chaikin money flow
% chvol     - Chaikin volatility
% cmo       - Chande momentum oscillator
% dirmov    - directional movement system, +-DI, DX, ADX
% dmi       - dynamic momentum index (DMI)
% dpo       - detrended price oscillator
% forosc    - forecast oscillator
% pnvi      - positive and negative volume indices, PVI, NVI
% prosc     - price oscillator & MACD
% psar      - Wilder's parabolic SAR
% rsi       - relative strength index, RSI
% stdev     - standard deviation index
% stoch     - stochastic oscillator, %K, %D oscillators
% trix      - TRIX oscillator
% vhfilt    - Vertical Horizontal Filter
%
% ------------ moving averages ---------------------------
%
% bma       - Butterworth moving average
% dema      - steady-state double exponential moving average
% ehma      - exponential Hull moving average
% gdema     - generalized dema
% hma       - Hull moving average
% ilrs      - integrated linear regression slope indicator
% delay     - delay or advance by d samples
% mom       - momentum and price rate of change
% lreg      - linear regression, slope, and R-squared indicators
% pma       - predictive moving average, linear fit
% pmaimp    - predictive moving average impulse response
% pma2      - predictive moving average, polynomial order d=1,2
% pmaimp2   - predictive moving average impulse response, d=1,2
% sema      - single exponential moving average
% shma      - SMA-based Hull moving average
% sma       - simple moving average
% t3        - Tillson's T3 indicator, triple gdema
% tema      - triple exponential moving average
% tma       - triangular moving average
% vema      - variable-length exponential moving average
% wema      - Wilder's exponential moving average
% wma       - weighted or linear moving average
% zema      - zero-lag EMA
%
% --------------- bands ----------------------------------
%
% bbands    - Bollinger bands
% donch     - Donchian channels
% fbands    - fixed-envelope bands
% kbands    - Keltner bands or channels
% pbands    - Projection bands and projection oscillator
% sebands   - standard-error bands
% stbands   - STARC bands
%
% --------------- misc ----------------------------------
%
% ohlc      - make Open-High-Low-Close bar chart
% ohlcyy    - OHLC with other indicators on the same graph
% yylim     - adjust left/right ylim
%
% r2crit    - R-squared critical values
% tcrit     - critical values of Student's t-distribution
% tdistr    - cumulative t-distribution
```

6.25 Problems

6.1 Consider a filter with a real-valued impulse response h_n. Let $H(\omega)= M(\omega)e^{-j\theta(\omega)}$ be its frequency response, where $M(\omega)=|H(\omega)|$ and $\theta(\omega)=-\arg H(\omega)$. First, argue that $\theta(0)=0$ and $M'(0)=0$, where $M'(\omega)=dM(\omega)/d\omega$. Then, show that the filter delay $\bar{n}$ of Eq. (6.1.18) is the group delay at DC, that is, show Eq. (6.1.19),

$$\bar{n} = \frac{d\theta(\omega)}{d\omega}\bigg|_{\omega=0}$$

6.2 The lag of a filter was defined by Eqs. (6.1.17) and (6.1.18) to be,

$$\bar{n} = \frac{\sum_n n h_n}{\sum_n h_n} = -\frac{H'(z)}{H(z)}\bigg|_{z=1}$$

If the filter $H(z)$ is the cascade of two filters, $H(z)= H_1(z)H_2(z)$, with individual lags, $\bar{n}_1, \bar{n}_2$, then show that, regardless of whether $H_1(z), H_2(z)$ are normalized to unity gain at DC, the lag of $H(z)$ will be the sum of the lags,

$$\bar{n} = \bar{n}_1 + \bar{n}_2$$

6.3 Consider a low-frequency signal $s(n)$ whose spectrum $S(\omega)$ is limited within a narrow band around DC, $|\omega| \le \Delta\omega$, and therefore, its inverse DTFT representation is:

$$s(n)= \frac{1}{2\pi}\int_{-\Delta\omega}^{\Delta\omega} S(\omega)e^{j\omega n}\,d\omega$$

For the purposes of this problem, we may think of the above relationship as defining $s(n)$ also for non-integer values of n. Suppose that the signal $s(n)$ is filtered through a filter $H(\omega)$ with real-valued impulse response whose magnitude response $|H(\omega)|$ is approximately equal to unity over the $\pm\Delta\omega$ signal bandwidth. Show that the filtered output can be written approximately as the delayed version of the input by an amount equal to the group delay at DC, that is,

$$y(n)= \frac{1}{2\pi}\int_{-\Delta\omega}^{\Delta\omega} H(\omega)S(\omega)e^{j\omega n}\,d\omega \approx s(n-\bar{n})$$

6.4 Show that the general filter-sharpening formula (6.9.5) results in the following special cases:

$$p = 0,\ q = d \quad \Rightarrow \quad H_{\text{impr}} = 1-(1-H)^{d+1}$$

$$p = 1,\ q = d \quad \Rightarrow \quad H_{\text{impr}} = 1-(1-H)^{d+1}\big[1+(d+1)H\big]$$

6.5 Prove the formulas in Eqs. (6.10.5) and (6.10.7).

6.6 Prove Eq. (6.4.10).

6.7 Consider the single and double EMA filters:

$$H(z)= \frac{1-\lambda}{1-\lambda z^{-1}}\,, \quad H_a(z)= 2H(z)-H^2(z)= \frac{(1-\lambda)(1+\lambda-2\lambda z^{-1})}{(1-\lambda z^{-1})^2}$$

a. Show that the impulse response of $H_a(z)$ is:

$$h_a(n)= (1-\lambda)\big[1+\lambda-(1-\lambda)n\big]\lambda^n u(n)$$

b. Show the relationships:

$$\sum_{n=0}^{\infty} n h_a(n) = 0, \quad \sum_{n=0}^{\infty} n^2 h_a(n) = -\frac{2\lambda^2}{(1-\lambda)^2} \tag{6.25.1}$$

c. Show that the NRR of the filter $H_a(z)$ is:

$$\mathcal{R} = \sum_{n=0}^{\infty} h_a^2(n) = \frac{(1-\lambda)(1+4\lambda+5\lambda^2)}{(1+\lambda)^3}$$

d. Show that the magnitude response squared of $H_a(z)$ is:

$$|H_a(\omega)|^2 = \frac{(1-\lambda)^2\big[1+2\lambda+5\lambda^2-4\lambda(1+\lambda)\cos\omega\big]}{\big[1-2\lambda\cos\omega+\lambda^2\big]^2} \tag{6.25.2}$$

e. Show that Eq. (6.25.2) has local minima at $\omega = 0$ and $\omega = \pi$, and a local maximum at $\omega = \omega_{\max}$:

$$\cos\omega_{\max} = \frac{1+4\lambda-\lambda^2}{2(1+\lambda)} \tag{6.25.3}$$

and that the corresponding extremal values are:

$$\begin{aligned} &|H_a(0)|^2 = 1\,, \quad |H_a(\pi)|^2 = \frac{(1-\lambda)^2(1+3\lambda)^2}{(1+\lambda)^4} \\ &|H_a(\omega_{\max})|^2 = \frac{(1+\lambda)^2}{1+2\lambda} \end{aligned} \tag{6.25.4}$$

6.8 Consider the modified EMA of Eq. (2.3.5) and its twicing,

$$H(z) = \frac{(1-\lambda)(1+z^{-1})}{2(1-\lambda z^{-1})}\,, \quad H_a(z) = 2H(z) - H^2(z) = \frac{(1-\lambda)\big(1+z^{-1}\big)\big(3+\lambda-z^{-1}(1+3\lambda)\big)}{4(1-\lambda z^{-1})^2}$$

a. Show the relationships:

$$\sum_{n=0}^{\infty} n h_a(n) = 0, \quad \sum_{n=0}^{\infty} n^2 h_a(n) = -\frac{(1+\lambda)^2}{2(1-\lambda)^2}$$

b. Show that the NRR of the filter $H_a(z)$ is:

$$\mathcal{R} = \sum_{n=0}^{\infty} h_a^2(n) = \frac{1}{8}(1-\lambda)(3\lambda+7)$$

6.9 Consider the optimum length-N predictive FIR filter $h_\tau(k)$ of polynomial order $d = 1$ given by Eq. (6.4.1).

a. Show that its effective lag is related to the prediction distance τ by $\bar{n} = -\tau$.

b. Show that its NRR is given by

$$\mathcal{R} = \sum_{k=0}^{N-1} h_\tau^2(k) = \frac{1}{N} + \frac{3(N-1+2\tau)^2}{N(N^2-1)}$$

Thus, it is minimized when $\tau = -(N-1)/2$. What is the filter $h_\tau(k)$ in this case?

c. Show that the second-derivative of its frequency response at DC is given by:

$$\frac{d^2}{d\omega^2}H(\omega)_{\omega=0} = -\sum_{n=0}^{\infty} k^2 h_\tau(k) = \frac{1}{6}(N-1)(N-2+6\tau)$$

Determine the range of τs for which $|H(\omega)|^2$ is sloping upwards or downwards in the immediate vicinity of $\omega = 0$.

d. It is evident from the previous question that the value $\tau = -(N-2)/6$ corresponds to the vanishing of the second-derivative of the magnitude response. Show that in this case the filter is simply,

$$h(k) = \frac{3N(N-1) - 2k(2N-1)}{N(N^2-1)}\,, \quad k = 0, 1, \dots, N-1$$

and verify explicitly the results:

$$\sum_{k=0}^{N-1} h(k) = 1\,, \quad \sum_{k=0}^{N-1} k h(k) = \frac{N-2}{6}\,, \quad \sum_{k=0}^{N-1} k^2 h(k) = 0\,, \quad \sum_{k=0}^{N-1} h^2(k) = \frac{7N^2-4N-2}{3N(N^2-1)}$$

e. Show that $h_\tau(k)$ interpolates linearly between the $\tau = 0$ and $\tau = 1$ filters, that is, show that for $k = 0, 1, \dots, N-1$,

$$h_\tau(k) = (1-\tau)h_a(k) + \tau h_1(k) = h_a(k) + \big[h_1(k) - h_a(k)\big]\tau$$

f. Another popular choice for the delay parameter is $\tau = -(N-1)/3$. Show that,

$$h(k) = \frac{2(N-k)}{N(N+1)}\,, \quad k = 0, 1, \dots, N-1$$

and that,

$$\sum_{k=0}^{N-1} h(k) = 1\,, \quad \sum_{k=0}^{N-1} k h(k) = \frac{N-1}{3}\,, \quad \sum_{k=0}^{N-1} k^2 h(k) = \frac{N(N-1)}{6}\,, \quad \sum_{k=0}^{N-1} h^2(k) = \frac{2(2N+1)}{3N(N+1)}$$

In financial market trading, the cases $\tau = -(N-1)/2$ and $\tau = -(N-1)/3$ correspond, respectively, to the so-called “simple” and “weighted” moving average indicators. The case $\tau = -(N-2)/6$ is not currently used, but it provides a useful compromise between reducing the lag while preserving the flatness of the passband. By comparison, the relative lags of three cases are:

$$\frac{1}{6}(N-2) < \frac{1}{3}(N-1) < \frac{1}{2}(N-1)$$

6.10 *Computer Experiment: Response of predictive FIR filters.* Consider the predictive FIR filter $h_\tau(k)$ of the previous problem. For $N = 9$, compute and on the same graph plot the magnitude responses $|H(\omega)|^2$ for the following values of the prediction distance:

$$\tau = -\frac{N-1}{2}\,, \quad \tau = -\frac{N-2}{6}\,, \quad \tau = 0\,, \quad \tau = 1$$

Using the calculated impulse response values $h_\tau(k)$, $0 \le k \le N-1$, and for each value of τ, calculate the filter lag, $\bar{n}$, the NRR, $\mathcal{R}$, and the “curvature” parameter of Eq. (6.10.5). Recall from part (d) of the Problem 6.9 that the second τ should result in zero curvature.

Repeat all the questions for $N = 18$.

6.11 *Moving-Average Filters with Prescribed Moments.* The predictive FIR filter of Eq. (6.16.3) has lag equal to $\bar{n} = -\tau$ by design. Show that its second moment is not independently specified but is given by,

$$\overline{n^2} = \sum_{n=0}^{N-1} n^2 h(n) = -\frac{1}{6}(N-1)(N-2+6\tau) \tag{6.25.5}$$

The construction of the predictive filters (6.16.3) can be generalized to allow arbitrary specification of the first and second moments, that is, the problem is to design a length-N FIR filter with the prescribed moments,

$$\overline{n^0} = \sum_{n=0}^{N-1} h(n) = 1\,, \quad \overline{n^1} = \sum_{n=0}^{N-1} n\,h(n) = -\tau_1\,, \quad \overline{n^2} = \sum_{n=0}^{N-1} n^2 h(n) = \tau_2 \tag{6.25.6}$$

Show that such filter is given by an expression of the form,

$$h(n) = c_0 + c_1 n + c_2 n^2\,, \quad n = 0, 1, \dots, N-1$$

where the coefficients c_0, c_1, c_2 are the solutions of the linear system,

$$\begin{bmatrix} S_0 & S_1 & S_2 \\ S_1 & S_2 & S_3 \\ S_2 & S_3 & S_4 \end{bmatrix} \begin{bmatrix} \lambda_0 \\ \lambda_1 \\ \lambda_2 \end{bmatrix} = \begin{bmatrix} 1 \\ -\tau_1 \\ \tau_2 \end{bmatrix}$$

where

$$S_p = \sum_{n=0}^{N-1} n^p\,, \quad p = 0, 1, 2, 3, 4$$

Then, show that the S_p are given explicitly by,

$$S_0 = N\,, \quad S_1 = \frac{1}{2}N(N-1)\,, \quad S_2 = \frac{1}{6}N(N-1)(2N-1)$$

$$S_3 = \frac{1}{4}N^2(N-1)^2\,, \quad S_4 = \frac{1}{30}N(N-1)(2N-1)(3N^2-3N-1)$$

and that the coefficients are given by,

$$c_0 = \frac{3(3N^2-3N+2) + 18(2N-1)\tau_1 + 30\tau_2}{N(N+1)(N+2)}$$

$$c_1 = -\frac{18(N-1)(N-2)(2N-1) + 12(2N-1)(8N-11)\tau_1 + 180(N-1)\tau_2}{N(N^2-1)(N^2-4)}$$

$$c_2 = \frac{30(N-1)(N-2) + 180(N-1)\tau_1 + 180\tau_2}{N(N^2-1)(N^2-4)}$$

Finally, show that the condition $c_2 = 0$ recovers the predictive FIR case of Eq. (6.16.3) with second moment given by Eq. (6.25.5).

6.12 Consider the Butterworth filter of Eq. (6.20.2). Show that the lag of the first-order section and the lag of the ith second-order section are given by,

$$\bar{n}_0 = \frac{1}{2\Omega_0}\,, \quad \bar{n}_i = -\frac{\cos\theta_i}{\Omega_0}\,, \quad i = 1, 2, \dots, K$$

Using these results, prove Eq. (6.20.8) for the full lag $\bar{n}$, and show that it is valid for both even and odd filter orders M.

7

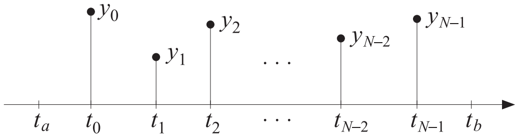
Smoothing Splines

7.1 Interpolation versus Smoothing

Besides their extensive use in drafting and computer graphics, splines have many other applications. A large online bibliography can be found in [350]. A small subset of references on interpolating and smoothing splines and their applications is [351–404].

We recall from Sec. 4.2 that the minimum-R_s filters had the property of maximizing the smoothness of the filtered output signal by minimizing the mean-square value of the s-differenced output, that is, the quantity $E\big[(\nabla^s \hat{x}_n)^2\big]$ in the notation of Eq. (4.2.11). Because of their finite span, minimum-R_s filters belong to the class of local smoothing methods. Smoothing splines are global methods in the sense that their design criterion involves the entire data signal to be smoothed, but their objective is similar, that is, to maximize smoothness.

We assume an observation model of the form $y(t) = x(t) + v(t)$, where $x(t)$ is a smooth trend to be estimated on the basis of N noisy observations $y_n = y(t_n)$ measured at N time instants t_n, for $n = 0, 1, \dots, N-1$, as shown below.



The times t_n, called the *knots*, are not necessarily equally-spaced, but are in increasing order and are assumed to lie within a slightly larger interval $[t_a, t_b]$, that is,

$$t_a < t_0 < t_1 < t_2 < \cdots < t_{N-1} < t_b$$

A smoothing spline fits a continuous function $x(t)$, taken to be the estimate of the underlying smooth trend, by solving the optimization problem:

$$\mathcal{J} = \sum_{n=0}^{N-1} w_n\big(y_n - x(t_n)\big)^2 + \lambda \int_{t_a}^{t_b} \big[x^{(s)}(t)\big]^2 dt = \min \tag{7.1.1}$$

where $x^{(s)}(t)$ denotes the s-th derivative of $x(t)$, λ is a positive "smoothing parameter," and w_n are given non-negative weights.

The performance index strikes a balance between interpolation and smoothing. The first term attempts to interpolate the data by $x(t)$, while the second attempts to minimize the roughness or maximize the smoothness of $x(t)$. The balance between the two terms is controlled by the parameter λ; larger λ increases smoothing, smaller λ interpolates the data more closely.

Schoenberg [357] has shown that the solution to the problem (7.1.1) is a so-called *natural smoothing spline* of polynomial order $2s-1$, that is, $x(t)$ has $2s-2$ continuous derivatives, it is a polynomial of degree $2s-1$ within each subinterval (t_n, t_{n+1}), for $n = 0, 1, \dots, N-2$, and it is a polynomial of order $s-1$ within the end subintervals $[t_a, t_0)$ and $(t_{N-1}, t_b]$.

For discrete-time sampled data, the problem was originally posed and solved for special cases of s by Thiele, Bohlmann, Whittaker, and Henderson [405–412], and is referred to as *Whittaker-Henderson* smoothing. We will consider it in Sec. 8.1. In this case, the performance index becomes:

$$\mathcal{J} = \sum_{n=0}^{N-1} w_n \big(y_n - x_n\big)^2 + \lambda \sum_{n=s}^{N-1} \big[\nabla^s x_n\big]^2 = \min \tag{7.1.2}$$

In this chapter, we concentrate on the case $s = 2$ for the problem (7.1.1), but allow an arbitrary s for problem (7.1.2). For $s = 2$, the performance index (7.1.1) reads:

$$\mathcal{J} = \sum_{n=0}^{N-1} w_n \big(y_n - x(t_n)\big)^2 + \lambda \int_{t_a}^{t_b} \big[\ddot{x}(t)\big]^2 dt = \min \tag{7.1.3}$$

Eq. (7.1.3) will be minimized under the assumption that the desired $x(t)$ and its first and second derivatives $\dot{x}(t), \ddot{x}(t)$ are continuous over $[t_a, t_b]$.

In the next section we solve the problem from a variational point of view and derive the solution as a natural cubic spline.

7.2 Variational Approach

We begin with a short review of variational calculus [354]. Consider first a Lagrangian $\mathcal{L}(x, \dot{x})$ that depends on a function $x(t)$ and its first derivative $\dot{x}(t)$.[†]

A prototypical variational problem is to find the function $x(t)$ that maximizes or minimizes the "action" functional:

$$\mathcal{J}(x) = \int_{t_a}^{t_b} \mathcal{L}(x, \dot{x})\, dt = \text{extremum} \tag{7.2.1}$$

The optimum function $x(t)$ is found by solving the Euler-Lagrange equation for (7.2.1):

$$\frac{\partial \mathcal{L}}{\partial x} - \frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{x}} = 0 \tag{7.2.2}$$

This can be derived as follows. Consider a small deviation from the optimum solution, $x(t) \to x(t) + \delta x(t)$. Then, the corresponding first-order variation of the functional (7.2.1) will be:

$$\begin{aligned}\delta\mathcal{J} = \mathcal{J}(x+\delta x) - \mathcal{J}(x) &= \int_{t_a}^{t_b} \big[\mathcal{L}(x+\delta x, \dot{x}+\delta\dot{x}) - \mathcal{L}(x, \dot{x})\big]\, dt \\ &= \int_{t_a}^{t_b} \Big[\frac{\partial \mathcal{L}}{\partial x}\delta x + \frac{\partial \mathcal{L}}{\partial \dot{x}}\delta\dot{x}\Big]\, dt = \int_{t_a}^{t_b} \Big[\frac{\partial \mathcal{L}}{\partial x}\delta x - \Big(\frac{\partial \mathcal{L}}{\partial \dot{x}}\Big)'\delta x + \Big(\frac{\partial \mathcal{L}}{\partial \dot{x}}\delta x\Big)'\Big]\, dt\end{aligned}$$

where we used the differential identity[‡]

$$\Big(\frac{\partial \mathcal{L}}{\partial \dot{x}}\delta x\Big)' = \frac{\partial \mathcal{L}}{\partial \dot{x}}\delta\dot{x} + \Big(\frac{\partial \mathcal{L}}{\partial \dot{x}}\Big)'\delta x \tag{7.2.3}$$

Integrating the last term in $\delta\mathcal{J}$, we obtain:

$$\delta\mathcal{J} = \int_{t_a}^{t_b} \Big[\frac{\partial \mathcal{L}}{\partial x} - \frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{x}}\Big]\delta x\, dt + \frac{\partial \mathcal{L}}{\partial \dot{x}}\delta x\Big|_{t_b} - \frac{\partial \mathcal{L}}{\partial \dot{x}}\delta x\Big|_{t_a} \tag{7.2.4}$$

The boundary terms can be removed by assuming the condition:

$$\frac{\partial \mathcal{L}}{\partial \dot{x}}\delta x\Big|_{t_b} - \frac{\partial \mathcal{L}}{\partial \dot{x}}\delta x\Big|_{t_a} = 0 \tag{7.2.5}$$

It follows that

$$\delta\mathcal{J} = \int_{t_a}^{t_b} \Big[\frac{\partial \mathcal{L}}{\partial x} - \frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{x}}\Big]\delta x\, dt \tag{7.2.6}$$

which defines the *functional derivative* of $\mathcal{J}(x)$:

$$\frac{\delta\mathcal{J}}{\delta x} = \frac{\partial \mathcal{L}}{\partial x} - \frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{x}} \tag{7.2.7}$$

The Euler-Lagrange equation (7.2.2) is obtained by requiring the vanishing of the functional derivative, or the vanishing of the first-order variation $\delta\mathcal{J}$ around the optimum solution for any choice of δx subject to (7.2.5).

The boundary condition (7.2.5) can be achieved in a number of ways. The typical one is to assume that the variation $\delta x(t)$ vanish at the endpoints, $\delta x(t_a) = \delta x(t_b) = 0$. Alternatively, if no restrictions are to be made on $\delta x(t)$, then one must assume the so-called *natural* boundary conditions [354]:

$$\frac{\partial \mathcal{L}}{\partial \dot{x}}\Big|_{t_a} = \frac{\partial \mathcal{L}}{\partial \dot{x}}\Big|_{t_b} = 0 \tag{7.2.8}$$

A mixed case is also possible in which at one end one assumes the vanishing of δx and at the other end, the vanishing of $\partial\mathcal{L}/\partial\dot{x}$.

The above results can be extended to the case when the Lagrangian is also a function of the second derivative $\ddot{x}$, that is, $\mathcal{L}(x, \dot{x}, \ddot{x})$. Using Eq. (7.2.3) and the identity,

$$\frac{\partial \mathcal{L}}{\partial \ddot{x}}\delta\ddot{x} = \Big(\frac{\partial \mathcal{L}}{\partial \ddot{x}}\delta\dot{x} - \Big(\frac{\partial \mathcal{L}}{\partial \ddot{x}}\Big)'\delta x\Big)' + \Big(\frac{\partial \mathcal{L}}{\partial \ddot{x}}\Big)''\delta x$$

[†] $\mathcal{L}$ can also have an explicit dependence on t, but we suppress it in the notation.

[‡] primes and dots denote differentiation with respect to t.

the first-order variation of $\mathcal{J}$ becomes

$$\delta\mathcal{J} = \mathcal{J}(x+\delta x) - \mathcal{J}(x) = \int_{t_a}^{t_b} \big[\mathcal{L}(x+\delta x,\, \dot{x}+\delta\dot{x},\, \ddot{x}+\delta\ddot{x}) - \mathcal{L}(x,\dot{x},\ddot{x})\big]\,dt$$

$$= \int_{t_a}^{t_b}\left[\frac{\partial\mathcal{L}}{\partial x}\delta x + \frac{\partial\mathcal{L}}{\partial\dot{x}}\delta\dot{x} + \frac{\partial\mathcal{L}}{\partial\ddot{x}}\delta\ddot{x}\right]dt = \int_{t_a}^{t_b}\left[\frac{\partial\mathcal{L}}{\partial x} - \frac{d}{dt}\frac{\partial\mathcal{L}}{\partial\dot{x}} + \frac{d^2}{dt^2}\frac{\partial\mathcal{L}}{\partial\ddot{x}}\right]\delta x\,dt$$

$$+ \left(\frac{\partial\mathcal{L}}{\partial\dot{x}} - \frac{d}{dt}\frac{\partial\mathcal{L}}{\partial\ddot{x}}\right)\delta x\,\Big|_{t_a}^{t_b} + \frac{\partial\mathcal{L}}{\partial\ddot{x}}\delta\dot{x}\,\Big|_{t_a}^{t_b}$$

To eliminate the boundary terms, we must assume that

$$\left(\frac{\partial\mathcal{L}}{\partial\dot{x}} - \frac{d}{dt}\frac{\partial\mathcal{L}}{\partial\ddot{x}}\right)\delta x\,\Big|_{t_a}^{t_b} + \frac{\partial\mathcal{L}}{\partial\ddot{x}}\delta\dot{x}\,\Big|_{t_a}^{t_b} = 0 \tag{7.2.9}$$

Then, the first-order variation and functional derivative of $\mathcal{J}$ become:

$$\delta\mathcal{J} = \int_{t_a}^{t_b}\left[\frac{\partial\mathcal{L}}{\partial x} - \frac{d}{dt}\frac{\partial\mathcal{L}}{\partial\dot{x}} + \frac{d^2}{dt^2}\frac{\partial\mathcal{L}}{\partial\ddot{x}}\right]\delta x\,dt\,, \quad \frac{\delta\mathcal{J}}{\delta x} = \frac{\partial\mathcal{L}}{\partial x} - \frac{d}{dt}\frac{\partial\mathcal{L}}{\partial\dot{x}} + \frac{d^2}{dt^2}\frac{\partial\mathcal{L}}{\partial\ddot{x}} \tag{7.2.10}$$

Their vanishing leads to the Euler-Lagrange equation for this case:

$$\frac{\partial\mathcal{L}}{\partial x} - \frac{d}{dt}\frac{\partial\mathcal{L}}{\partial\dot{x}} + \frac{d^2}{dt^2}\frac{\partial\mathcal{L}}{\partial\ddot{x}} = 0 \tag{7.2.11}$$

subject to the condition (7.2.9). In the spline problem, because the endpoints t_a, t_b lie slightly outside the knot range, we do not want to impose any restrictions on the values of δx and $\delta\dot{x}$ there. Therefore, to satisfy (7.2.9), we will assume the four *natural* boundary conditions:

$$\frac{\partial\mathcal{L}}{\partial\dot{x}} - \frac{d}{dt}\frac{\partial\mathcal{L}}{\partial\ddot{x}}\,\Big|_{t_a} = 0\,, \quad \frac{\partial\mathcal{L}}{\partial\ddot{x}}\,\Big|_{t_a} = 0\,, \quad \frac{\partial\mathcal{L}}{\partial\dot{x}} - \frac{d}{dt}\frac{\partial\mathcal{L}}{\partial\ddot{x}}\,\Big|_{t_b} = 0\,, \quad \frac{\partial\mathcal{L}}{\partial\ddot{x}}\,\Big|_{t_b} = 0 \tag{7.2.12}$$

The spline problem (7.1.3) can be put in a variational form as follows,

$$\mathcal{J} = \sum_{n=0}^{N-1} w_n\big(y_n - x(t_n)\big)^2 + \lambda\int_{t_a}^{t_b}\big[\ddot{x}(t)\big]^2 dt = \int_{t_a}^{t_b}\mathcal{L}\,dt = \min \tag{7.2.13}$$

where the Lagrangian depends only on x and $\ddot{x}$,

$$\mathcal{L} = \sum_{n=0}^{N-1} w_n\big(y_n - x(t)\big)^2\delta(t-t_n) + \lambda\big[\ddot{x}(t)\big]^2 \tag{7.2.14}$$

The Euler-Lagrange equation (7.2.11) then reads:

$$\frac{\partial\mathcal{L}}{\partial x} - \frac{d}{dt}\frac{\partial\mathcal{L}}{\partial\dot{x}} + \frac{d^2}{dt^2}\frac{\partial\mathcal{L}}{\partial\ddot{x}} = -2\sum_{n=0}^{N-1} w_n\big(y_n - x(t)\big)\delta(t-t_n) + 2\lambda\ddddot{x}(t) = 0\,, \quad \text{or,}$$

$$\boxed{\ddddot{x}(t) = \lambda^{-1}\sum_{n=0}^{N-1} w_n\big(y_n - x(t_n)\big)\delta(t-t_n)} \tag{7.2.15}$$

where we replaced $\big(y_n - x(t)\big)\delta(t-t_n)$ by $\big(y_n - x(t_n)\big)\delta(t-t_n)$ in the right-hand side. The natural boundary conditions (7.2.12) become:

$$\dddot{x}(t_a) = 0\,, \quad \ddot{x}(t_a) = 0\,, \quad \dddot{x}(t_b) = 0\,, \quad \ddot{x}(t_b) = 0 \tag{7.2.16}$$

7.3 Natural Cubic Smoothing Splines

Eq. (7.2.15) implies that $\ddddot{x}(t) = 0$ for all t except at the knot times t_n. This means that $x(t)$ must be a cubic polynomial in t. Within each knot interval $[t_n, t_{n+1}]$, for $n = 0, 1, \dots, N-2$, and within the end-point intervals $[t_a, t_0]$ and $[t_{N-1}, t_b]$, the function $x(t)$ must be a cubic polynomial, albeit with different coefficients in each interval.

Specifically, the boundary conditions (7.2.16) imply that within $[t_a, t_0]$ and $[t_{N-1}, t_b]$, the third-degree polynomials must actually be polynomials of first-degree. Thus, $x(t)$ will have the form:

$$x(t) = \begin{cases} p_{-1}(t) = a_{-1} + b_{-1}(t - t_a)\,, & t_a \le t \le t_0 \\ p_n(t) = a_n + b_n(t-t_n) + \dfrac{1}{2}c_n(t-t_n)^2 + \dfrac{1}{6}d_n(t-t_n)^3\,, & t_n \le t \le t_{n+1} \\ p_{N-1}(t) = a_{N-1} + b_{N-1}(t - t_{N-1})\,, & t_{N-1} \le t \le t_b \end{cases} \tag{7.3.1}$$

where $n = 0, 1, \dots, N-2$ for the interval $[t_n, t_{n+1}]$, and we have referred the time origin to the left end of each subinterval. We note that $a_n = x(t_n) = p_n(t_n)$, $b_n = \dot{p}_n(t_n)$, $c_n = \ddot{p}_n(t_n)$, and $d_n = \dddot{p}_n(t_n)$, for $n = 0, 1, \dots, N-1$. The a_n are the smoothed values.

The polynomial pieces join continuously at the knots. The term "natural" cubic spline refers to the property that $x(t)$ is a linear function of t outside the knot range, and consists of cubic polynomial pieces that are continuous and have continuous first and second derivatives at the knot times. Fig. 7.3.1 illustrates the case of $N = 5$ and the numbering convention that we follow.

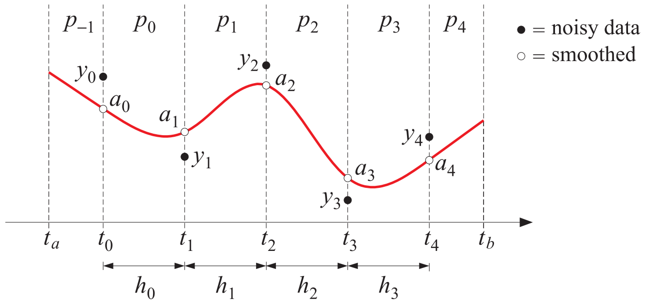


Fig. 7.3.1 Smoothing with natural cubic splines.

Although $x(t)$, $\dot{x}(t)$, $\ddot{x}(t)$ are continuous at the knots, Eq. (7.2.15) implies that the third derivatives $\dddot{x}(t)$ must be discontinuous. Indeed, integrating (7.2.15) around the interval $[t_n - \epsilon, t_n + \epsilon]$ and taking the limit $\epsilon \to 0$, we obtain the N discontinuity conditions:

$$\dddot{x}(t_n)_+ - \dddot{x}(t_n)_- = \lambda^{-1}w_n(y_n - a_n)\,, \quad n = 0, 1, \dots, N-1 \tag{7.3.2}$$

where $\dddot{x}(t_n)_\pm = \lim_{\epsilon\to 0}\dddot{x}(t_n \pm \epsilon)$, and $a_n = x(t_n)$. Expressed in terms of the polynomial pieces, the continuity and discontinuity conditions can be stated as follows:

$$\begin{aligned}
p_n(t_n) &= p_{n-1}(t_n)\,, \quad n = 0,1,\dots,N-1\\
\dot{p}_n(t_n) &= \dot{p}_{n-1}(t_n)\\
\ddot{p}_n(t_n) &= \ddot{p}_{n-1}(t_n)\\
\dddot{p}_n(t_n) - \dddot{p}_{n-1}(t_n) &= \lambda^{-1}w_n(y_n - a_n)
\end{aligned} \tag{7.3.3}$$

These provide $4N$ equations. The number of unknown coefficients is also $4N$. Indeed, there are $N-1$ strictly cubic polynomials plus the two linear polynomials at the ends, thus, the total number of coefficients is $4(N-1)+2\cdot 2 = 4N$.

In solving these equations, we follow Reinsch's procedure [358] that eliminates b_n, d_n in favor of a_n, c_n. We begin by applying the continuity conditions (7.3.3) at $t = t_0$,

$$\begin{aligned}
a_0 &= a_{-1} + b_{-1}(t_0 - t_a)\\
b_0 &= b_{-1}\\
c_0 &= 0\\
d_0 &= \lambda^{-1}w_0(y_0 - a_0)
\end{aligned} \tag{7.3.4}$$

where in the last two we used $c_{-1} = d_{-1} = 0$. From the first two, it follows that the left-most polynomial can be referred to time origin t_0 and written alternatively as,

$$p_{-1}(t) = a_{-1} + b_{-1}(t - t_a) = a_0 + b_0(t - t_0) \tag{7.3.5}$$

For $n = 1,2,\dots,N-1$, defining $h_{n-1} = t_n - t_{n-1}$, conditions (7.3.3) read:

$$\begin{aligned}
a_n &= a_{n-1} + b_{n-1}h_{n-1} + \frac{1}{2}c_{n-1}h_{n-1}^2 + \frac{1}{6}d_{n-1}h_{n-1}^3\\
b_n &= b_{n-1} + c_{n-1}h_{n-1} + \frac{1}{2}d_{n-1}h_{n-1}^2\\
c_n &= c_{n-1} + d_{n-1}h_{n-1}\\
d_n - d_{n-1} &= \lambda^{-1}w_n(y_n - a_n)
\end{aligned} \tag{7.3.6}$$

Since $c_{N-1} = d_{N-1} = 0$, we have at $n = N-1$:

$$\begin{aligned}
a_{N-1} &= a_{N-2} + b_{N-2}h_{N-2} + \frac{1}{2}c_{N-2}h_{N-2}^2 + \frac{1}{6}d_{N-2}h_{N-2}^3\\
b_{N-1} &= b_{N-2} + c_{N-2}h_{N-2} + \frac{1}{2}d_{N-2}h_{N-2}^2\\
0 &= c_{N-2} + d_{N-2}h_{N-2}\\
0 - d_{N-2} &= \lambda^{-1}w_{N-1}(y_{N-1} - a_{N-1})
\end{aligned}$$

Using the third into the first two equations, we may rewrite them as,

$$\begin{aligned}
a_{N-1} &= a_{N-2} + b_{N-2}h_{N-2} + \frac{1}{3}c_{N-2}h_{N-2}^2\\
b_{N-1} &= b_{N-2} + \frac{1}{2}c_{N-2}h_{N-2}\\
c_{N-2} &= -d_{N-2}h_{N-2}\\
d_{N-2} &= -\lambda^{-1}w_{N-1}(y_{N-1} - a_{N-1})
\end{aligned} \tag{7.3.7}$$

From the third of Eq. (7.3.6), we have

$$d_{n-1} = \frac{c_n - c_{n-1}}{h_{n-1}}\,, \quad n = 1,2,\dots,N-1 \tag{7.3.8}$$

In particular, we obtain at $n = 1$ and $n = N-1$,

$$\begin{aligned}
d_0 &= \frac{c_1 - c_0}{h_0} = \frac{c_1}{h_0} = \lambda^{-1}w_0(y_0 - a_0)\\
-d_{N-2} &= -\frac{c_{N-1} - c_{N-2}}{h_{N-2}} = \frac{c_{N-2}}{h_{N-2}} = \lambda^{-1}w_{N-1}(y_{N-1} - a_{N-1})
\end{aligned} \tag{7.3.9}$$

where we used Eqs. (7.3.4) and (7.3.7). Inserting Eq. (7.3.8) into the last of (7.3.6), we obtain for $n = 1,2,\dots,N-2$:

$$d_n - d_{n-1} = \frac{c_{n+1} - c_n}{h_n} - \frac{c_n - c_{n-1}}{h_{n-1}} = \lambda^{-1}w_n(y_n - a_n) \tag{7.3.10}$$

Thus, combining these with (7.3.9), we obtain an $N\times(N-2)$ tridiagonal system of equations that relates the $(N-2)$-dimensional vector $\mathbf{c} = [c_1, c_2, \dots, c_{N-2}]^T$ to the N-dimensional vector $\mathbf{a} = [a_0, a_1, \dots, a_{N-1}]^T$:

$$\begin{aligned}
&\frac{c_1}{h_0} = \lambda^{-1}w_0(y_0 - a_0)\\
&\frac{1}{h_{n-1}}c_{n-1} - \left(\frac{1}{h_{n-1}} + \frac{1}{h_n}\right)c_n + \frac{1}{h_n}c_{n+1} = \lambda^{-1}w_n(y_n - a_n)\,, \quad n = 1,2,\dots,N-2\\
&\frac{c_{N-2}}{h_{N-2}} = \lambda^{-1}w_{N-1}(y_{N-1} - a_{N-1})
\end{aligned} \tag{7.3.11}$$

where we must use $c_0 = c_{N-1} = 0$. These may be written in a matrix form by defining the vector $\mathbf{y} = [y_0, y_1, \dots, y_{N-1}]^T$ and weight matrix $W = \text{diag}\big([w_0, w_1, \dots, w_{N-1}]\big)$,

$$Q\mathbf{c} = \lambda^{-1}W(\mathbf{y} - \mathbf{a}) \tag{7.3.12}$$

The $N\times(N-2)$ tridiagonal matrix Q has non-zero matrix elements:

$$Q_{n-1,n} = \frac{1}{h_{n-1}}\,, \quad Q_{n,n} = -\left(\frac{1}{h_{n-1}} + \frac{1}{h_n}\right)\,, \quad Q_{n+1,n} = \frac{1}{h_n} \tag{7.3.13}$$

for $n = 1, 2, \dots, N-2$. We note that the matrix elements Q_{ni} were assumed to be indexed such that $0 \le n \le N-1$ and $1 \le i \le N-2$. Next, we determine another relationship between a_n and c_n. Substituting Eq. (7.3.8) into the first and second of (7.3.6), we obtain:

$$\begin{aligned} a_n - a_{n-1} &= b_{n-1}h_{n-1} + \frac{1}{6}(c_n + 2c_{n-1})h_{n-1}^2\,, \quad n = 1, 2, \dots, N-1 \\ b_n - b_{n-1} &= \frac{1}{2}(c_n + c_{n-1})h_{n-1} \end{aligned} \tag{7.3.14}$$

The first of these can be solved for b_{n-1} in terms of a_n:

$$\begin{aligned} b_{n-1} &= \frac{a_n - a_{n-1}}{h_{n-1}} - \frac{1}{6}(c_n + 2c_{n-1})h_{n-1}\,, \qquad n = 1, 2, \dots, N-1 \\ b_n &= \frac{a_{n+1} - a_n}{h_n} - \frac{1}{6}(c_{n+1} + 2c_n)h_n\,, \quad n = 0, 1, \dots, N-2 \end{aligned} \tag{7.3.15}$$

Substituting these into the second of (7.3.14), we obtain for $n = 1, 2, \dots, N-2$:

$$\frac{1}{h_{n-1}}a_{n-1} - \left(\frac{1}{h_{n-1}} + \frac{1}{h_n}\right)a_n + \frac{1}{h_n}a_{n+1} = \frac{1}{6}h_{n-1}c_{n-1} + \frac{1}{3}(h_{n-1} + h_n)c_n + \frac{1}{6}h_n c_{n+1} \tag{7.3.16}$$

This an $(N-2)\times N$ tridiagonal system with the transposed of Q appearing on the left, and the following $(N-2)\times(N-2)$ symmetric tridiagonal matrix on the right,

$$\begin{aligned} T_{n,n} &= \frac{1}{3}(h_{n-1} + h_n)\,, \qquad 1 \le n \le N-2 \\ T_{n+1,n} = T_{n,n+1} &= \frac{1}{6}h_n\,, \qquad 1 \le n \le N-3 \end{aligned} \tag{7.3.17}$$

Thus, the system (7.3.16) can be written compactly as,

$$Q^T\mathbf{a} = T\mathbf{c} \tag{7.3.18}$$

To summarize, the optimal coefficients $\mathbf{a}, \mathbf{c}$ are coupled by

$$\boxed{\begin{aligned} Q^T\mathbf{a} &= T\mathbf{c} \\ Q\mathbf{c} &= \lambda^{-1}W(\mathbf{y} - \mathbf{a}) \end{aligned}} \tag{7.3.19}$$

To clarify the nature of the matrices Q, T, consider the case $N = 6$ with data vector $\mathbf{y} = [y_0, y_1, y_2, y_3, y_4, y_5]^T$. The matrix equations (7.3.19) read explicitly,

$$\begin{bmatrix} h_0^{-1} & -(h_0^{-1} + h_1^{-1}) & h_1^{-1} & 0 & 0 & 0 \\ 0 & h_1^{-1} & -(h_1^{-1} + h_2^{-1}) & h_2^{-1} & 0 & 0 \\ 0 & 0 & h_2^{-1} & -(h_2^{-1} + h_3^{-1}) & h_3^{-1} & 0 \\ 0 & 0 & 0 & h_3^{-1} & -(h_3^{-1} + h_4^{-1}) & h_4^{-1} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix}$$

$$= \frac{1}{6}\begin{bmatrix} 2(h_0 + h_1) & h_1 & 0 & 0 \\ h_1 & 2(h_1 + h_2) & h_2 & 0 \\ 0 & h_2 & 2(h_2 + h_3) & h_3 \\ 0 & 0 & h_3 & 2(h_3 + h_4) \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix}$$

$$\begin{bmatrix} h_0^{-1} & 0 & 0 & 0 \\ -(h_0^{-1} + h_1^{-1}) & h_1^{-1} & 0 & 0 \\ h_1^{-1} & -(h_1^{-1} + h_2^{-1}) & h_2^{-1} & 0 \\ 0 & h_2^{-1} & -(h_2^{-1} + h_3^{-1}) & h_3^{-1} \\ 0 & 0 & h_3^{-1} & -(h_3^{-1} + h_4^{-1}) \\ 0 & 0 & 0 & h_4^{-1} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix} = \lambda^{-1}\begin{bmatrix} w_0(y_0 - a_0) \\ w_1(y_1 - a_1) \\ w_2(y_2 - a_2) \\ w_3(y_3 - a_3) \\ w_4(y_4 - a_4) \\ w_5(y_5 - a_5) \end{bmatrix}$$

In order to have a non-trivial vector $\mathbf{c}$, we will assume that $N \ge 3$. Eqs. (7.3.19) can be solved in a straightforward way. Since T is square and invertible, we may solve the first for $\mathbf{c} = T^{-1}Q^T\mathbf{a}$ and substitute into the second,

$$QT^{-1}Q^T\mathbf{a} = \lambda^{-1}W(\mathbf{y} - \mathbf{a}) \quad \Rightarrow \quad (W + \lambda QT^{-1}Q^T)\mathbf{a} = W\mathbf{y} \quad \text{or,}$$

$$\boxed{\mathbf{a} = (W + \lambda QT^{-1}Q^T)^{-1}W\mathbf{y}} \tag{7.3.20}$$

so that the filtering (the so-called "hat") matrix for the smoothing operation $\mathbf{a} = H\mathbf{y}$ is

$$H = (W + \lambda QT^{-1}Q^T)^{-1}W \tag{7.3.21}$$

Although both Q and T are banded matrices with bandwidth three, the inverse T^{-1} is not banded and neither is $(W + \lambda QT^{-1}Q^T)$. Therefore, the indicated matrix inverse is computationally expensive, requiring $O(N^3)$ operations. However, there is an alternative algorithm due to Reinsch [358] that reduces the computational cost to $O(N)$ operations. From the second of (7.3.19), we have after multiplying it by Q^TW^{-1},

$$\lambda W^{-1}Q\mathbf{c} = \mathbf{y} - \mathbf{a} \quad \Rightarrow \quad \lambda Q^TW^{-1}Q\mathbf{c} = Q^T\mathbf{y} - Q^T\mathbf{a} = Q^T\mathbf{y} - T\mathbf{c}$$

which may be solved for $\mathbf{c}$

$$(T + \lambda Q^TW^{-1}Q)\mathbf{c} = Q^T\mathbf{y} \quad \Rightarrow \quad \mathbf{c} = (T + \lambda Q^TW^{-1}Q)^{-1}Q^T\mathbf{y}$$

where now because W is diagonal, the matrix $R = T + \lambda Q^TW^{-1}Q$ is banded with bandwidth five, and therefore it can be inverted in $O(N)$ operations. This leads to Reinsch's efficient computational algorithm:

$$\boxed{\begin{aligned} R &= T + \lambda Q^TW^{-1}Q \\ \mathbf{c} &= R^{-1}Q^T\mathbf{y} \\ \mathbf{a} &= \mathbf{y} - \lambda W^{-1}Q\mathbf{c} \end{aligned}} \tag{7.3.22}$$

This implies an alternative expression for the matrix H. Eliminating $\mathbf{c}$, we have,

$$\mathbf{a} = \mathbf{y} - \lambda W^{-1}QR^{-1}Q^T\mathbf{y} \quad \Rightarrow \quad \mathbf{a} = (I - \lambda W^{-1}QR^{-1}Q^T)\mathbf{y}\,, \quad \text{or,}$$

$$H = I - \lambda W^{-1}QR^{-1}Q^T = I - \lambda W^{-1}Q(T + \lambda Q^TW^{-1}Q)^{-1}Q^T \tag{7.3.23}$$

The equivalence of Eqs. (7.3.21) and (7.3.23) follows from the matrix inversion lemma. Once the polynomial coefficients $\mathbf{c} = [c_1, c_2, \dots, c_{N-2}]^T$ and $\mathbf{a} = [a_0, a_1, \dots, a_{N-1}]^T$

have been computed, the b_n and d_n coefficients can be obtained from Eqs. (7.3.8) and (7.3.14), and (7.3.7), with $c_0 = c_{N-1} = 0$,

$$\begin{aligned} d_n &= \frac{c_{n+1} - c_n}{h_n}\,, \quad n = 0, 1, \dots, N-2\,, \quad \text{and} \quad d_{N-1} = 0 \\ b_n &= \frac{a_{n+1} - a_n}{h_n} - \frac{1}{6}(c_{n+1} + 2c_n)h_n\,, \quad n = 0, 1, \dots, N-2 \\ b_{N-1} &= b_{N-2} + \frac{1}{2}c_{N-2}h_{N-2} \end{aligned} \tag{7.3.24}$$

Eqs. (7.3.22) and (7.3.24) provide the complete solution for the coefficients for all the polynomial pieces. We note two particular limits of the solution. For $\lambda = 0$, Eq. (7.3.22) gives $R = T$ and

$$\mathbf{c} = T^{-1}Q^T\mathbf{y}\,, \quad \mathbf{a} = \mathbf{y} \tag{7.3.25}$$

Thus, the smoothing spline interpolates the data, that is, $x(t_n) = a_n = y_n$. Interpolating splines are widely used in image processing and graphics applications.

For $\lambda \to \infty$, the solution corresponds to fitting a straight line to the entire data set. In this case, Eq. (7.3.23) has a well-defined limit,

$$H = I - \lambda W^{-1}Q(T + \lambda Q^T W^{-1} Q)^{-1}Q^T \to I - W^{-1}Q(Q^T W^{-1}Q)^{-1}Q^T \tag{7.3.26}$$

and Eqs. (7.3.22) become:

$$\mathbf{c} = 0\,, \quad \mathbf{a} = \mathbf{y} - W^{-1}Q(Q^T W^{-1} Q)^{-1}Q^T\mathbf{y} \tag{7.3.27}$$

Since $\mathbf{c} = 0$, Eqs. (7.3.24) imply that $d_n = 0$, therefore, the polynomial pieces $p_n(t)$ are first-order polynomials, and we also have $b_n = (a_{n+1} - a_n)/h_n$. The vector $\mathbf{a}$ lies in the null space of Q^T. Indeed, multiplying by Q^T, we have from (7.3.27),

$$Q^T\mathbf{a} = Q^T\mathbf{y} - (Q^T W^{-1}Q)(Q^T W^{-1}Q)^{-1}Q^T\mathbf{y} = Q^T\mathbf{y} - Q^T\mathbf{y} = 0$$

Component-wise this means that the slopes b_n of the $p_n(t)$ polynomials are the same,

$$(Q^T\mathbf{a})_n = \frac{a_{n+1} - a_n}{h_n} - \frac{a_n - a_{n-1}}{h_{n-1}} = b_n - b_{n-1} = 0 \tag{7.3.28}$$

Thus, the polynomials $p_n(t)$ represent pieces of the same straight line. Indeed, setting $b_n \equiv \beta$, and using $a_n = a_{n-1} + \beta h_{n-1}$, we obtain,

$$p_n(t) = a_n + \beta(t - t_n) = a_{n-1} + \beta h_{n-1} + \beta(t - t_n) = a_{n-1} + \beta(t - t_{n-1}) = p_{n-1}(t)$$

This line corresponds to a weighted least-squares straight-line fit through the data y_n, that is, fitting a polynomial $p(t) = \alpha + \beta t$ to

$$\mathcal{J} = \sum_{n=0}^{N-1} w_n\big(y_n - p(t_n)\big)^2 = (\mathbf{y} - \hat{\mathbf{y}})^T W(\mathbf{y} - \hat{\mathbf{y}}) = \min$$

It is easily verified that the coefficients and fitted values $\hat{\mathbf{y}} = [p(t_0), p(t_1), \dots, p(t_n)]^T$ are given by

$$\begin{bmatrix} \alpha \\ \beta \end{bmatrix} = (S^T WS)^{-1}S^T W\mathbf{y}\,, \quad \hat{\mathbf{y}} = S(S^T WS)^{-1}S^T W\mathbf{y} \tag{7.3.29}$$

where S is the $N\times 2$ polynomial basis matrix defined by

$$S = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ t_0 & t_1 & \cdots & t_{N-1} \end{bmatrix}^T \tag{7.3.30}$$

The fitted values $\hat{\mathbf{y}}$ are exactly equal to those of (7.3.27), as can be verified using the following projection matrix identity, which can be proved using the property $Q^T S = 0$,

$$W^{-1}Q(Q^T W^{-1}Q)^{-1}Q^T + S(S^T WS)^{-1}S^T W = I \tag{7.3.31}$$

7.4 *Optimality of Natural Splines*

The smoothing spline solution just derived is not only an extremum of the performance index (7.1.3), but also a minimum. To show this, consider a deviation from the optimum solution, $x(t) + f(t)$, where $x(t)$ is the solution (7.3.1) and $f(t)$ an arbitrary twice differentiable function. Then, we must show that $\mathcal{J}(x + f) \geq \mathcal{J}(x)$. Noting that $a_n = x(t_n)$ and denoting $f_n = f(t_n)$, we have,

$$\begin{aligned} \mathcal{J}(x + f) &= \sum_{n=0}^{N-1} w_n(y_n - a_n - f_n)^2 + \lambda\int_{t_a}^{t_b} [\ddot{x}(t) + \ddot{f}(t)]^2 dt \\ \mathcal{J}(x) &= \sum_{n=0}^{N-1} w_n(y_n - a_n)^2 + \lambda\int_{t_a}^{t_b} [\ddot{x}(t)]^2 dt \end{aligned}$$

and by subtracting,

$$\begin{aligned} \mathcal{J}(x + f) - \mathcal{J}(x) &= \sum_{n=0}^{N-1} w_n[f_n^2 - 2(y_n - a_n)f_n] + \lambda\int_{t_a}^{t_b} [\ddot{f}(t)^2 + 2\ddot{x}(t)\ddot{f}(t)]dt \\ &= \sum_{n=0}^{N-1} w_n f_n^2 + \lambda\int_{t_a}^{t_b} \ddot{f}(t)^2\,dt - 2\sum_{n=0}^{N-1} w_n(y_n - a_n)f_n + 2\lambda\int_{t_a}^{t_b} \ddot{x}(t)\ddot{f}(t)\,dt \end{aligned}$$

The first two terms are non-negative. Therefore, the desired result $\mathcal{J}(x + f) \geq \mathcal{J}(x)$ would follow if we can show that the last two terms that are linear in f cancel each other. Indeed, this follows from the optimality conditions (7.3.3). Splitting the integration range as a sum over the subintervals, and replacing $\ddot{x}(t)$ by $\ddot{p}_n(t)$ over the nth subinterval, we have,

$$\begin{aligned} \int_{t_a}^{t_b} \ddot{x}(t)\ddot{f}(t)dt &= \int_{t_a}^{t_0} \ddot{p}_{-1}(t)\ddot{f}(t)dt + \sum_{n=0}^{N-2}\int_{t_n}^{t_{n+1}} \ddot{p}_n(t)\ddot{f}(t)dt + \int_{t_{N-1}}^{t_b} \ddot{p}_{N-1}(t)\ddot{f}(t)dt \\ &= \sum_{n=0}^{N-2}\int_{t_n}^{t_{n+1}} \Big[\big(\ddot{p}_n(t)\dot{f}(t)\big)' - \dddot{p}_n(t)\dot{f}(t)\Big]\,dt \end{aligned}$$

where we dropped the first an last integrals because $p_{-1}(t)$ and $p_{N-1}(t)$ are linear and have vanishing second derivatives, and used the identity $\ddot{p}_n\ddot{f} = (\ddot{p}_n\dot{f})' - \dddot{p}_n\dot{f}$. The first term is a complete derivative and can be integrated simply. In the second term, we may use $\dddot{p}_n(t)=d_n$ over the nth subinterval to obtain,

$$\int_{t_a}^{t_b}\ddot{x}(t)\ddot{f}(t)\,dt=\ddot{p}_{N-2}(t_{N-1})\dot{f}(t_{N-1})-\ddot{p}_0(t_0)\dot{f}(t_0)-\sum_{n=0}^{N-2}\int_{t_n}^{t_{n+1}}d_n\dot{f}(t)\,dt$$

$$=\ddot{p}_{N-2}(t_{N-1})\ddot{f}(t_{N-1})-\ddot{p}_0(t_0)\ddot{f}(t_0)-\sum_{n=0}^{N-2}d_n(f_{n+1}-f_n)$$

From the continuity at $t=t_0$ and $t=t_{N-1}$, we have $\ddot{p}_{N-2}(t_{N-1})=\ddot{p}_{N-1}(t_{N-1})=0$ and $\ddot{p}_0(t_0)=\ddot{p}_{-1}(t_0)=0$. Thus, we find,

$$\int_{t_a}^{t_b}\ddot{x}(t)\ddot{f}(t)\,dt=-\sum_{n=0}^{N-2}d_n(f_{n+1}-f_n)=d_0f_0+\sum_{n=1}^{N-2}(d_n-d_{n-1})f_n-d_{N-2}f_{N-1}$$

Using Eqs. (7.3.4), (7.3.6), and (7.3.7), we obtain

$$\int_{t_a}^{t_b}\ddot{x}(t)\ddot{f}(t)\,dt=\lambda^{-1}\sum_{n=0}^{N-1}w_n(y_n-a_n)f_n$$

Thus, these two terms cancel in the difference of the performance indices,

$$\mathcal{J}(x+f)-\mathcal{J}(x)=\sum_{n=0}^{N-1}w_nf_n^2+\lambda\int_{t_a}^{t_b}\ddot{f}(t)^2\,dt \tag{7.4.1}$$

Hence, $\mathcal{J}(x+f)\geq\mathcal{J}(x)$, with equality achieved when $\ddot{f}(t)=0$ and $f_n=f(t_n)=0$, which imply that $f(t)=0$.

Although we showed that the *interpolating* spline case corresponds to the special case $\lambda=0$, it is worth looking at its optimality properties from a variational point of view. Simply setting $\lambda=0$ into the performance index (7.1.3) is not useful because it only implies the interpolation property $x(t_n)=y_n$. An alternative point of view is to consider the following constrained variational problem:

$$\begin{aligned}&\mathcal{J}=\int_{t_a}^{t_b}\big[\ddot{x}(t)\big]^2dt=\min\\&\text{subject to }\ x(t_n)=y_n\,,\quad n=0,1,\dots,N-1\end{aligned} \tag{7.4.2}$$

The constraints can be incorporated using a set of Lagrange multipliers μ_n,

$$\mathcal{J}=\sum_{n=0}^{N-1}2\mu_n\big(y_n-x(t_n)\big)+\int_{t_a}^{t_b}\big[\ddot{x}(t)\big]^2dt=\min \tag{7.4.3}$$

The corresponding effective Lagrangian is,

$$\mathcal{L}=\sum_{n=0}^{N-1}2\mu_n\big(y_n-x(t)\big)\delta(t-t_n)+\big[\ddot{x}(t)\big]^2 \tag{7.4.4}$$

The Euler-Lagrange equation (7.2.11) then gives,

$$\ddddot{x}(t)=\sum_{n=0}^{N-1}\mu_n\delta(t-t_n) \tag{7.4.5}$$

which is to be solved subject to the same natural boundary conditions as (7.2.16),

$$\dddot{x}(t_a)=0\,,\quad\ddot{x}(t_a)=0\,,\quad\dddot{x}(t_b)=0\,,\quad\ddot{x}(t_b)=0 \tag{7.4.6}$$

This is identical to the smoothing spline case with the replacement of $\lambda^{-1}(y_n-a_n)$ by μ_n, or, vectorially $\lambda^{-1}W(\mathbf{y}-\mathbf{a})\rightarrow\boldsymbol{\mu}$. Therefore, the solution will be a natural spline with Eq. (7.3.19) replaced by

$$Q^T\mathbf{y}=T\mathbf{c}\,,\quad Q\mathbf{c}=\boldsymbol{\mu}$$

which is the same as the $\lambda=0$ smoothing spline case. Thus, the interpolating spline solution is defined by $\mathbf{a}=\mathbf{y}$ and $\mathbf{c}=T^{-1}Q^T\mathbf{y}$, with the equation $Q\mathbf{c}=\boldsymbol{\mu}$ fixing the Lagrange multiplier vector.

7.5 Generalized Cross Validation

The cross-validation and generalized cross-validation criteria are popular ways of choosing the smoothing parameter λ. We encountered these criteria in sections 4.5 and 5.2.

The cross-validation criterion selects the λ that minimizes the weighted sum of squared errors [352]:

$$\text{CV}(\lambda)=\frac{1}{N}\sum_{i=0}^{N-1}w_i(y_i-a_i^-)^2=\min \tag{7.5.1}$$

where a_i^- is the estimate of the sample y_i obtained by *deleting* the ith observation y_i and basing the spline smoothing on the remaining observations. As was the case in Sec. 5.2, we may show that

$$y_i-a_i^-=\frac{y_i-a_i}{1-H_{ii}} \tag{7.5.2}$$

where H_{ii} is the ith diagonal element of the filtering matrix H of the smoothing problem with the observation y_i included, and a_i, the corresponding estimate of y_i. Thus, the CV index can be expressed as:

$$\text{CV}(\lambda)=\frac{1}{N}\sum_{i=0}^{N-1}w_i(y_i-a_i^-)^2=\frac{1}{N}\sum_{i=0}^{N-1}w_i\left(\frac{y_i-a_i}{1-H_{ii}}\right)^2=\min \tag{7.5.3}$$

The generalized cross-validation criterion replaces H_{ii} by its average over i, that is,

$$\text{GCV}(\lambda)=\frac{1}{N}\sum_{i=0}^{N-1}w_i\left(\frac{y_i-a_i}{1-\bar{H}}\right)^2=\min\,,\quad\bar{H}=\frac{1}{N}\sum_{i=0}^{N-1}H_{ii}=\frac{1}{N}\,\text{tr}(H) \tag{7.5.4}$$

The GCV can be evaluated efficiently with $O(N)$ operations for each value of λ using the algorithm of [377]. Noting that $1-\bar{H}=\big(N-\text{tr}(H)\big)/N=\text{tr}(I-H)/N$, and defining $\mathbf{e}=\mathbf{y}-\mathbf{a}=(I-H)\mathbf{y}$, the GCV can be written in a slightly different form,

$$\frac{1}{N}\,\mathrm{GCV}(\lambda) = \frac{\sum_{i=0}^{N-1} w_i(y_i - a_i)^2}{\big[\mathrm{tr}(I-H)\big]^2} = \frac{\mathbf{e}^T W\mathbf{e}}{\big[\mathrm{tr}(I-H)\big]^2} = \min \tag{7.5.5}$$

To show Eq. (7.5.2), consider the index (7.1.3) with the i-th observation y_i deleted:

$$\mathcal{J}_- = \sum_{\substack{n=0\\ n\neq i}}^{N-1} w_n\big(y_n - x(t_n)\big)^2 + \lambda\int_{t_a}^{t_b}\big[\ddot{x}(t)\big]^2 dt = \min \tag{7.5.6}$$

The i-th term can be included provided we attach zero weight to it, that is, we may define $w_n^- = w_n$, if $n \neq i$, and $w_i^- = 0$:

$$\mathcal{J}_- = \sum_{n=0}^{N-1} w_n^-\big(y_n - x(t_n)\big)^2 + \lambda\int_{t_a}^{t_b}\big[\ddot{x}(t)\big]^2 dt = \min \tag{7.5.7}$$

It follows from Eq. (7.3.20) that the optimum solutions with and without the i observation are given by

$$\begin{aligned}
\mathbf{a} &= H\mathbf{y} = F^{-1}W\mathbf{y}\,, & F &= W + \lambda QT^{-1}Q^T\\
\mathbf{a}^- &= H_-\mathbf{y} = F_-^{-1}W_-\mathbf{y}\,, & F_- &= W_- + \lambda QT^{-1}Q^T
\end{aligned} \tag{7.5.8}$$

where W_- is the diagonal matrix of the w_n^-. Defining the i-th unit vector that has one in its i-th slot, $\mathbf{u}_i = [0,\dots,0,1,0,\dots,0]^T$, then W_- is related to the original W by

$$W_- = W - w_i\,\mathbf{u}_i\mathbf{u}_i^T \quad\Rightarrow\quad F_- = F - w_i\,\mathbf{u}_i\mathbf{u}_i^T$$

It follows from Eq. (7.5.8) that,

$$F_-\mathbf{a}^- = W_-\mathbf{y} \quad\Rightarrow\quad (F - w_i\,\mathbf{u}_i\mathbf{u}_i^T)\mathbf{a}^- = (W - w_i\,\mathbf{u}_i\mathbf{u}_i^T)\mathbf{y}$$

Noting that $y_i = \mathbf{u}_i^T\mathbf{y}$ and $a_i^- = \mathbf{u}_i^T\mathbf{a}^-$, we have after multiplying by F^{-1},

$$\mathbf{a}^- - w_iF^{-1}\mathbf{u}_i\,a_i^- = \mathbf{a} - w_iF^{-1}\mathbf{u}_i y_i \quad\Rightarrow\quad \mathbf{a} - \mathbf{a}^- = w_iF^{-1}\mathbf{u}_i\,(y_i - a_i^-)$$

Multiplying by $\mathbf{u}_i^T$ and noting that $H_{ii} = \mathbf{u}_i^T H\mathbf{u}_i = \mathbf{u}_i^T F^{-1}W\mathbf{u}_i = (\mathbf{u}_i^T F^{-1}\mathbf{u}_i)\,w_i$, we find,

$$a_i - a_i^- = H_{ii}(y_i - a_i^-) \tag{7.5.9}$$

which is equivalent to (7.5.2). An intuitive interpretation [352] of $\mathbf{a}^-$ is that it is obtainable by the original filtering matrix H acting on a modified observation vector $\mathbf{y}^*$ whose i-th entry has been replaced by the estimated value $y_i^* = a_i^-$, and whose other entries agree with those of $\mathbf{y}$. To show it, we note that $W\mathbf{y}^* = W_-\mathbf{y} + w_i\mathbf{u}_i y_i^*$. Then, we have

$$F_-\mathbf{a}^- = W_-\mathbf{y} \quad\Rightarrow\quad (F - w_i\,\mathbf{u}_i\mathbf{u}_i^T)\mathbf{a}^- = W\mathbf{y}^* - w_i\mathbf{u}_i y_i^* \quad\Rightarrow\quad F\mathbf{a}^- = W\mathbf{y}^* - w_i\mathbf{u}_i\,(y_i^* - a_i^-)$$

Thus, if we choose $y_i^* = a_i^-$, we have $F\mathbf{a}^- = W\mathbf{y}^*$, which gives $\mathbf{a}^- = F^{-1}W\mathbf{y}^* = H\mathbf{y}^*$. A similar result was obtained in Sec. 4.5.

7.6 Repeated Observations

We discussed how to handle repeated observations in local polynomial modeling in Sec. 5.5, replacing the repeated observations by their averages and using their multiplicities to modify the weighting function. A similar procedure can be derived for the spline smoothing case.

Assuming that at each knot time t_n there are m_n observations, y_{ni} with weights w_{ni}, $i = 1, 2, \dots, m_n$, the performance index (7.1.3) may be modified as follows:

$$\mathcal{J} = \sum_{n=0}^{N-1}\sum_{i=1}^{m_n} w_{ni}\big(y_{ni} - x(t_n)\big)^2 + \lambda\int_{t_a}^{t_b}\big[\ddot{x}(t)\big]^2 dt = \min \tag{7.6.1}$$

Let us define the weighted-averaged observations and corresponding weights by:

$$\boxed{\bar{y}_n = \frac{1}{\bar{w}_n}\sum_{i=1}^{m_n} w_{ni}y_{ni}\,, \quad \bar{w}_n = \sum_{i=1}^{m_n} w_{ni}} \tag{7.6.2}$$

If the weights w_{ni} are unity, $\bar{y}_n$ and $\bar{w}_n$ reduce to ordinary averages and multiplicities. It is easily verified that $\mathcal{J}$ can be written in the alternative form:

$$\mathcal{J} = \sum_{n=0}^{N-1} \bar{w}_n\big(\bar{y}_n - x(t_n)\big)^2 + \lambda\int_{t_a}^{t_b}\big[\ddot{x}(t)\big]^2 dt + \text{const.} = \min \tag{7.6.3}$$

up to a constant that does not depend on the unknown function $x(t)$ to be determined. Thus, the case of multiple observations may be reduced to an ordinary spline smoothing problem.

7.7 Equivalent Filter

The filtering equation of a smoothing spline, $\mathbf{a} = H\mathbf{y}$, raises the question of whether it is possible to view it as an ordinary convolutional filtering operation. Such a viewpoint indeed arises if we replace the performance index (7.1.3) with the following one, which assumes the availability of continuous-time observations $y(t)$ for $-\infty < t < \infty$,

$$\mathcal{J} = \int_{-\infty}^{\infty}\big|y(t) - x(t)\big|^2 dt + \lambda\int_{-\infty}^{\infty}\big|\ddot{x}(t)\big|^2 dt = \min \tag{7.7.1}$$

The solution can be carried out easily in the frequency domain. Using Parseval's identity and denoting the Fourier transforms of $y(t), x(t)$ by $Y(\omega), X(\omega)$, and noting that the transform of $\ddot{x}(t)$ is $-\omega^2 X(\omega)$, we obtain the equivalent criterion,

$$\mathcal{J} = \int_{-\infty}^{\infty}\big|Y(\omega) - X(\omega)\big|^2\frac{d\omega}{2\pi} + \lambda\int_{-\infty}^{\infty}\omega^4\big|X(\omega)\big|^2\frac{d\omega}{2\pi} = \min \tag{7.7.2}$$

Setting the functional derivative of $\mathcal{J}$ with respect to $X^*(\omega)$ to zero,[†] we obtain the Euler-Lagrange equation in this case:[‡]

$$\frac{\delta\mathcal{J}}{\delta X^*(\omega)} = -\big[Y(\omega) - X(\omega)\big] + \lambda\omega^4 X(\omega) = 0 \tag{7.7.3}$$

[†] $X(\omega)$ and its complex conjugate $X^*(\omega)$ are treated as independent variables in Eqs. (7.7.2) and (7.7.3).

[‡] The boundary conditions for this variational problem are that $X(\omega)\to 0$ for $\omega \to \pm\infty$.

which leads to the transfer function $H(\omega)=X(\omega)/Y(\omega)$ between the input $Y(\omega)$ and the output $X(\omega)$:

$$H(\omega)=\frac{1}{1+\lambda\omega^4} \qquad \text{(equivalent smoothing filter)} \tag{7.7.4}$$

Its impulse response (i.e., the inverse Fourier transform) is

$$h(t)=\frac{a}{2}\big(\sin a|t|+\cos at\big)e^{-a|t|}\,, \qquad -\infty<t<\infty \tag{7.7.5}$$

where $a=(4\lambda)^{-1/4}$. The impulse response $h(t)$ is double-sided, and therefore, it cannot be used in real-time applications. However, it is evident that the filter is a lowpass filter with a (6-dB) cutoff frequency of $\omega_0=\lambda^{-1/4}$. Fig. 7.7.1 depicts $h(t)$ and $H(\omega)$ for three values of the smoothing parameter, $\lambda=1$, $\lambda=1/5$, and $\lambda=5$.

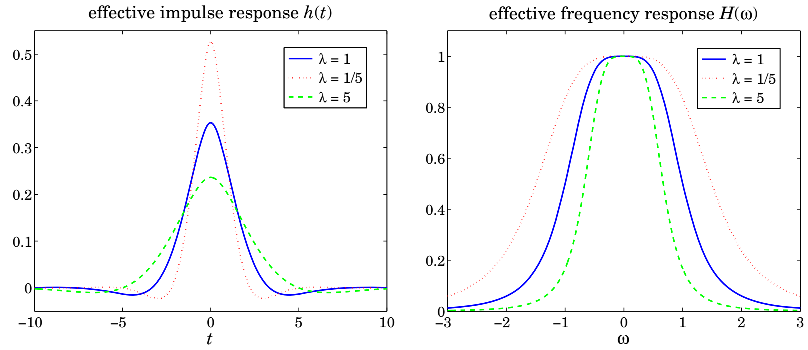


Fig. 7.7.1 Effective impulse and frequency responses in spline smoothing.

One can also work in the time-domain with similar results. The Euler-Lagrange equation (7.2.11) leads to,

$$x(t)-y(t)+\lambda\ddddot{x}(t)=0 \quad\Rightarrow\quad x(t)+\lambda\ddddot{x}(t)=y(t) \tag{7.7.6}$$

Fourier transforming both sides we obtain $(1+\lambda\omega^4)X(\omega)=Y(\omega)$, which leads to Eq. (7.7.4) by solving for $H(\omega)=X(\omega)/Y(\omega)$.

A similar approach will be used in the Whittaker-Henderson discrete-time case discussed in Sec. 8.1 The resulting filter is often referred to in the business and finance literature as the Hodrick-Prescott filter.

Variants of the Whittaker-Henderson approach were first introduced in 1880 by Thiele [405,406] and in 1899 by Bohlmann [407]. Bohlmann considered and solved both the discrete- and continuous-time versions of the performance index,

$$\begin{aligned}\mathcal{J}&=\sum_{n=0}^{N-1}(y_n-x_n)^2+\lambda\sum_{n=1}^{N-1}(x_n-x_{n-1})^2=\min\\ \mathcal{J}&=\int_{t_a}^{t_b}\big|y(t)-x(t)\big|^2dt+\lambda\int_{t_a}^{t_b}\big|\dot{x}(t)\big|^2dt=\min\end{aligned} \tag{7.7.7}$$

In the continuous-time case, the Euler-Lagrange equation, transfer function, and impulse response of the resulting smoothing filter are:

$$x(t)-\lambda\ddot{x}(t)=y(t) \quad\Rightarrow\quad H(\omega)=\frac{1}{1+\lambda\omega^2}\,,\quad h(t)=\frac{1}{2\sqrt{\lambda}}e^{-|t|/\sqrt{\lambda}} \tag{7.7.8}$$

Thiele considered the unequally-spaced knot case and the weighted performance index:

$$\mathcal{J}=\sum_{n=0}^{N-1}\frac{1}{\sigma_n^2}\big[y_n-x(t_n)\big]^2+\sum_{n=1}^{N-1}\frac{1}{w_n^2}\big[x(t_n)-x(t_{n-1})\big]^2=\min \tag{7.7.9}$$

It is remarkable that Thiele formulated this problem as a state-space model—to use modern parlance—and solved it recursively using essentially the Kalman filter and associated smoother. Moreover, he showed how to estimate the unknown model parameters using the EM algorithm. We will be discussing these ideas later on.

7.8 Stochastic Model

Like the exponential smoothing case, spline smoothing can be given a stochastic state-space model interpretation [397–404]. The spline function solution $x(t)$ of Eq. (7.3.1) can be regarded as an optimum linear estimate of the underlying stochastic process based on the N observations $\{y_0,y_1,\dots,y_{N-1}\}$ subject to some additional assumptions on the initial conditions [399].

The state-space model allows the use of Kalman filtering techniques resulting in efficient computational algorithms, which like the Reinsch algorithm are also $O(N)$. But in addition, the state-space model allows the estimation of the smoothing parameter.

The basis of such a stochastic model (for the cubic spline case) is the stochastic differential equation:

$$\ddot{x}(t)=w(t) \tag{7.8.1}$$

where $w(t)$ is a zero-mean white-noise process of variance σ_w^2, that is, its autocorrelation function is $E\big[w(t)w(\tau)\big]=\sigma_w^2\delta(t-\tau)$.

In the observation model $y(t)=x(t)+v(t)$, we may assume that $v(t)$ is uncorrelated with $w(t)$ and is white noise with variance σ_v^2. It turns out that the smoothing parameter can be identified as the ratio $\lambda=\sigma_v^2/\sigma_w^2$. The N actual observed values are $y_n=x(t_n)+v(t_n)$. Integrating Eq. (7.8.1) over the interval $[t_n,t]$, we obtain,

$$\begin{aligned}\dot{x}(t)&=\dot{x}(t_n)+\int_{t_n}^{t}w(\tau)d\tau\\ x(t)&=x(t_n)+(t-t_n)\dot{x}(t_n)+\int_{t_n}^{t}(t-\tau)w(\tau)d\tau\end{aligned} \tag{7.8.2}$$

The process $\dot{x}(t)$ is integrated white noise, or a Wiener or Brownian process. The process $x(t)$ is an integrated Wiener process. We may write these in vector form by defining the state and noise vectors,

$$\mathbf{x}_t=\begin{bmatrix}x(t)\\ \dot{x}(t)\end{bmatrix},\quad \mathbf{x}_n=\begin{bmatrix}x(t_n)\\ \dot{x}(t_n)\end{bmatrix},\quad \mathbf{w}_t=\int_{t_n}^{t}\begin{bmatrix}t-\tau\\ 1\end{bmatrix}w(\tau)d\tau \tag{7.8.3}$$

and the state transition matrix,

$$A(t,t_n)=\begin{bmatrix} 1 & t-t_n \\ 0 & 1 \end{bmatrix} \tag{7.8.4}$$

Then, Eq. (7.8.2) can be written compactly as

$$\mathbf{x}_t = A(t,t_n)\mathbf{x}_n + \mathbf{w}_t \tag{7.8.5}$$

The covariance matrix of the noise component $\mathbf{w}_t$ is:

$$\begin{aligned} E[\mathbf{w}_t\mathbf{w}_t^T] &= \int_{t_n}^{t}\int_{t_n}^{t}\begin{bmatrix} t-\tau \\ 1 \end{bmatrix}[t-\tau',\, 1]\, E[w(\tau)w(\tau')]\,d\tau d\tau' \\ &= \int_{t_n}^{t}\int_{t_n}^{t}\begin{bmatrix} t-\tau \\ 1 \end{bmatrix}[t-\tau',\, 1]\,\sigma_w^2\,\delta(\tau-\tau')\,d\tau d\tau' \\ &= \sigma_w^2\begin{bmatrix} \frac{1}{3}(t-t_n)^3 & \frac{1}{2}(t-t_n)^2 \\ \frac{1}{2}(t-t_n)^2 & (t-t_n) \end{bmatrix} \end{aligned} \tag{7.8.6}$$

At $t = t_{n+1}$, we obtain the state equation,

$$\mathbf{x}_{n+1} = A(t_{n+1},t_n)\mathbf{x}_n + \mathbf{w}_{n+1}\,, \quad \mathbf{w}_{n+1} = \int_{t_n}^{t_{n+1}}\begin{bmatrix} t_{n+1}-\tau \\ 1 \end{bmatrix} w(\tau)\,d\tau \tag{7.8.7}$$

where, using $h_n = t_{n+1} - t_n$,

$$A(t_{n+1},t_n)=\begin{bmatrix} 1 & h_n \\ 0 & 1 \end{bmatrix}, \quad E[\mathbf{w}_{n+1}\mathbf{w}_{n+1}^T]=\sigma_w^2\begin{bmatrix} \frac{1}{3}h_n^3 & \frac{1}{2}h_n^2 \\ \frac{1}{2}h_n^2 & h_n \end{bmatrix} \tag{7.8.8}$$

In terms of the spline coefficients, we have $a_n = x(t_n)$ and $b_n = \dot{x}(t_n)$ at $t = t_n$, and similarly at $t = t_{n+1}$. Following [28], we would like to show the following estimation result. Given the state-vectors $\mathbf{x}_n, \mathbf{x}_{n+1}$ at the two end points of the interval $[t_n, t_{n+1}]$, the spline function $x(t)$ of (7.3.1), and its derivative $\dot{x}(t)$, can be regarded as the mean-square estimates of the state-vector $\mathbf{x}_t$ based on $\mathbf{x}_n, \mathbf{x}_{n+1}$, that is, assuming gaussian noises, given by the conditional mean,

$$\hat{\mathbf{x}}_t = E[\mathbf{x}_t|\mathbf{x}_n, \mathbf{x}_{n+1}] \tag{7.8.9}$$

If we orthogonalize $\mathbf{x}_{n+1}$ with respect to $\mathbf{x}_n$, that is, replacing it by the innovations vector $\boldsymbol{\varepsilon}_{n+1} = \mathbf{x}_{n+1} - E[\mathbf{x}_{n+1}|\mathbf{x}_n]$, then we may use the regression lemma from Chap. 1 to write (7.8.9) in the form:

$$\hat{\mathbf{x}}_t = E[\mathbf{x}_t|\mathbf{x}_n, \mathbf{x}_{n+1}] = E[\mathbf{x}_t|\mathbf{x}_n] + \Sigma_{x_t\varepsilon_{n+1}}\Sigma_{\varepsilon_{n+1}\varepsilon_{n+1}}^{-1}\boldsymbol{\varepsilon}_{n+1} \tag{7.8.10}$$

We have from Eq. (7.8.5) and (7.8.7),

$$E[\mathbf{x}_t|\mathbf{x}_n] = A(t,t_n)\mathbf{x}_n\,, \quad E[\mathbf{x}_{n+1}|\mathbf{x}_n] = A(t_{n+1},t_n)\mathbf{x}_n \tag{7.8.11}$$

the latter implying,

$$\boldsymbol{\varepsilon}_{n+1} = \mathbf{x}_{n+1} - E[\mathbf{x}_{n+1}|\mathbf{x}_n] = \mathbf{x}_{n+1} - A(t_{n+1},t_n)\mathbf{x}_n = \mathbf{w}_{n+1} \tag{7.8.12}$$

and therefore, we have for the covariance matrices:

$$\Sigma_{x_t\varepsilon_{n+1}} = E[\mathbf{x}_t\boldsymbol{\varepsilon}_{n+1}^T] = E[\mathbf{x}_t\mathbf{w}_{n+1}^T] = E[\mathbf{w}_t\mathbf{w}_{n+1}^T]\,, \quad \Sigma_{\varepsilon_{n+1}\varepsilon_{n+1}} = E[\mathbf{w}_{n+1}\mathbf{w}_{n+1}^T]$$

The latter has already been calculated in (7.8.7). For the former, we split the integration range of $\mathbf{w}_{n+1}$ as follows,

$$\mathbf{w}_{n+1} = \int_{t_n}^{t_{n+1}}\begin{bmatrix} t_{n+1}-\tau \\ 1 \end{bmatrix} w(\tau)\,d\tau = \left(\int_{t_n}^{t} + \int_{t}^{t_{n+1}}\right)\begin{bmatrix} t_{n+1}-\tau \\ 1 \end{bmatrix} w(\tau)\,d\tau$$

and note that only the first term is correlated with $\mathbf{w}_t$, thus, resulting in

$$\begin{aligned} \Sigma_{x_t\varepsilon_{n+1}} &= E[\mathbf{w}_t\mathbf{w}_{n+1}^T] = \int_{t_n}^{t}\int_{t_n}^{t}\begin{bmatrix} t-\tau \\ 1 \end{bmatrix}[t_{n+1}-\tau',\, 1]\, E[w(\tau)w(\tau')]\,d\tau d\tau' \\ &= \sigma_w^2\int_{t_n}^{t}\begin{bmatrix} t-\tau \\ 1 \end{bmatrix}[t_{n+1}-\tau, 1]\,d\tau \\ &= \begin{bmatrix} \frac{1}{6}(t-t_n)^2(t_n+2h_n-t) & \frac{1}{2}(t-t_n)^2 \\ \frac{1}{2}(t-t_n)(t_n+2h_n-t) & (t-t_n) \end{bmatrix} \end{aligned} \tag{7.8.13}$$

We may now calculate the estimation matrix $H_{n+1} = \Sigma_{x_t\varepsilon_{n+1}}\Sigma_{\varepsilon_{n+1}\varepsilon_{n+1}}^{-1}$,

$$H_{n+1} = \begin{bmatrix} \frac{1}{h_n^3}(t-t_n)^2(2t_n+3h_n-2t) & \frac{1}{h_n^2}(t-t_n)^2(t-t_n-h_n) \\ \frac{6}{h_n^3}(t-t_n)(t_n+h_n-t) & \frac{1}{h_n^2}(t-t_n)(3t-3t_n-2h_n) \end{bmatrix} \tag{7.8.14}$$

It follows that the estimate $\hat{\mathbf{x}}_t$ is

$$\hat{\mathbf{x}}_t = E[\mathbf{x}_t|\mathbf{x}_n] + H_{n+1}\boldsymbol{\varepsilon}_{n+1} = A(t,t_n)\mathbf{x}_n + H_{n+1}\big(\mathbf{x}_{n+1} - A(t_{n+1},t_n)\mathbf{x}_n\big) \tag{7.8.15}$$

Setting

$$\hat{\mathbf{x}}_t = \begin{bmatrix} \hat{x}(t) \\ \hat{\dot{x}}(t) \end{bmatrix}, \quad \mathbf{x}_n = \begin{bmatrix} a_n \\ b_n \end{bmatrix}, \quad \mathbf{x}_{n+1} = \begin{bmatrix} a_{n+1} \\ b_{n+1} \end{bmatrix}$$

we obtain

$$\begin{aligned} \hat{x}(t) &= a_n + b_n(t-t_n) + \frac{1}{h_n^3}(t-t_n)^2(2t_n+3h_n-2t)(a_{n+1}-a_n-b_nh_n) \\ &\quad + \frac{1}{h_n^2}(t-t_n)^2(t-t_n-h_n)(b_{n+1}-b_n) \\ \hat{\dot{x}}(t) &= b_n + \frac{6}{h_n^3}(t-t_n)(t_n+h_n-t)(a_{n+1}-a_n-b_nh_n) \\ &\quad + \frac{1}{h_n^2}(t-t_n)(3t-3t_n-2h_n)(b_{n+1}-b_n) \end{aligned}$$

Using the continuity relationships (7.3.6),

$$a_{n+1} = a_n + b_n h_n + \frac{1}{2}c_n h_n^2 + \frac{1}{6}d_n h_n^3$$

$$b_{n+1} = b_n + c_n h_n + \frac{1}{2}d_n h_n^2$$

it follows that the expressions for $\hat{x}(t)$ and $\hat{\dot{x}}(t)$ reduce to those of Eq. (7.3.1),

$$\begin{aligned}\hat{x}(t) &= a_n + b_n(t-t_n)+\frac{1}{2}c_n(t-t_n)^2+\frac{1}{6}d_n(t-t_n)^3\\ \hat{\dot{x}}(t) &= b_n + c_n(t-t_n)+\frac{1}{2}d_n(t-t_n)^2\end{aligned}$$

the second being of course the derivative of the first. The asymptotic filter (7.7.4) may also be given a stochastic interpretation in the sense that it can be regarded as the optimum double-sided (i.e., unrealizable) Wiener filter of estimating $x(t)$ from $y(t)$ of the signal model,

$$y(t)=x(t)+v(t)\,,\quad \ddot{x}(t)=w(t) \tag{7.8.16}$$

We will see in Chap. 11 that for stationary signals $x(t), y(t)$, with power spectral densities $S_{xy}(\omega)$ and $S_{yy}(\omega)$, the optimum double-sided Wiener filter has frequency response:

$$H(\omega)=\frac{S_{xy}(\omega)}{S_{yy}(\omega)} \tag{7.8.17}$$

Because $x(t)$ is an integrated Wiener process, it is not stationary, and therefore, $S_{xy}(\omega)$ and $S_{yy}(\omega)$ do not exist. However, it has been shown [643–649] that for certain types of nonstationary signals, which have the property that they become stationary under a suitable filtering transformation, Eq. (7.8.17) remains valid in the following modified form:

$$H(\omega)=\frac{S_{\bar{x}\bar{y}}(\omega)}{S_{\bar{y}\bar{y}}(\omega)} \tag{7.8.18}$$

where $\bar{x}(t), \bar{y}(t)$ are the stationary filtered versions of $x(t), y(t)$. For the model of Eq. (7.8.16), the necessary filtering operation is double differentiation, $\bar{x}(t)=\ddot{x}(t)=w(t)$, which can be expressed in the frequency domain as $\bar{X}(\omega)=D(\omega)X(\omega)$, with $D(\omega)=(j\omega)^2=-\omega^2$. For the observation signal, we have similarly $\bar{y}(t)=\ddot{y}=\ddot{x}+\ddot{v}$. Since $w(t), v(t)$ are uncorrelated, we find

$$\begin{aligned}S_{\bar{x}\bar{y}}(\omega) &= S_{ww}(\omega)=\sigma_w^2\\ S_{\bar{y}\bar{y}}(\omega) &= S_{ww}(\omega)+S_{vv}(\omega)|D(\omega)|^2=\sigma_w^2+\sigma_v^2\,\omega^4\end{aligned}$$

which leads to

$$H(\omega)=\frac{S_{\bar{x}\bar{y}}(\omega)}{S_{\bar{y}\bar{y}}(\omega)}=\frac{\sigma_w^2}{\sigma_w^2+\sigma_v^2\,\omega^4}=\frac{1}{1+\lambda\omega^4}\,,\quad \lambda=\frac{\sigma_v^2}{\sigma_w^2} \tag{7.8.19}$$

This can be written in the form,

$$H(\omega)=\frac{\sigma_w^2/\omega^4}{\sigma_w^2/\omega^4+\sigma_v^2} \tag{7.8.20}$$

which is what we would get from Eq. (7.8.17) had we pretended that the spectral densities did exist. Indeed it would follow in such a case from Eq. (7.8.16) that $S_{xy}(\omega)=\sigma_w^2/\omega^4$ and $S_{yy}(\omega)=\sigma_w^2/\omega^4+\sigma_v^2$.

7.9 Computational Aspects

Eqs. (7.3.22) and (7.3.24) describing the complete spline solution have been implemented by the MATLAB function `splsm`,

```
P = splsm(t,y,lambda,w);        % spline smoothing
```

where `t,y` are the knot times $[t_0, t_1, \dots, t_{N-1}]$ and data $[y_0, y_1, \dots, y_{N-1}]$ (entered as row or column vectors), `lambda` is the smoothing parameter λ, and `w` the vector of weights $[w_0, w_1, \dots, w_{N-1}]$, which default to unity values. The output `P` is an $N\times 4$ matrix whose n-th row are the polynomial coefficients $[a_n, b_n, c_n, d_n]$. Thus, the vector **a** is the first column of `P`. Internally, the matrices T, Q are computed as sparse banded matrices with the help of the function `splmat`,

```
[T,Q] = splmat(h);        % spline sparse matrices T,Q
```

where `h` is the vector of knot spacings $[h_0, h_1, \dots, h_{N-1}]$, which is simply computed by the `diff` operation on the knot times `t`, that is, `h=diff(t)`. The smoothing spline may be evaluated at any value of t in the range $t_a \le t \le t_b$ using Eq. (7.3.1). The function `splval` performs the evaluation of $x(t)$ at any vector of t's,

```
ys = splval(P,t,ts);        % spline evaluation at a vector of grid points ts
```

where `ys` is the vector of values $x(t_s)$, and `P,t` are the spline coefficients and knot times. The GCV criterion (7.5.5) (with the $1/N$ factor removed) may be calculated for any vector of λ values by the function `splgcv`:

```
gcv = splgcv(t,y,lambda,w);        % GCV evaluation at a vector of λ's
```

The optimum λ may be selected by finding the minimum of the GCV over the computed range. Alternatively, the optimum λ may be computed by the related function `splambda`, which performs a golden-mean search over a given interval of λ's,

```
[lopt,gcvopt] = splambda(t,y,la,lb,Nit,w);        % determine optimum λ
```

The starting interval is $[\lambda_a, \lambda_b]$ and `Nit` denotes the number of golden-mean iterations (typically, 10–20). The function `splsm2` is a "robustified" version of `splsm`,

```
[P,ta] = splsm2(t,y,la,w,Nit);        % robust spline smoothing
```

The function starts with the original triplet `[t,y,w]` and uses the LOESS method of repeatedly modifying the weights (with a total of `Nit` iterations), with the outliers being given smaller weights. Because of the modification and zeroing of some of the weights, the output matrix `P` will have dimension $N_a\times 4$ with $N_a \le N$. The function also outputs the corresponding knot times `ta` (also N_a-dimensional) that survive the down-weighting process.

All of the above functions assume that the observations y_n are unique at the knot times t_n. If there are repeated observations, then the weighted observations and their weights given by Eq. (7.6.2) must be the inputs to the above functions. They may be determined with the function `splav`, which is similar in spirit to the function `avobs`, except that it computes weighted averages instead of plain averages:

```
[ta,ya,wa] = splav(t,y,w);        % weighted averages of repeated observations
```

where the outputs `[ta,ya,wa]` are the resulting unique knot times, observations, and weights.

Example 7.9.1: *Motorcycle data.* The usage of these functions is illustrated by the motorcycle data that we considered earlier in local polynomial modeling. The upper-left graph of Fig. 7.9.1 shows a plot of the GCV calculated with the function `splgcv`. The optimum value was found to be $\lambda_{\text{opt}} = 15.25$ by the function `splambda` and placed on the graph.

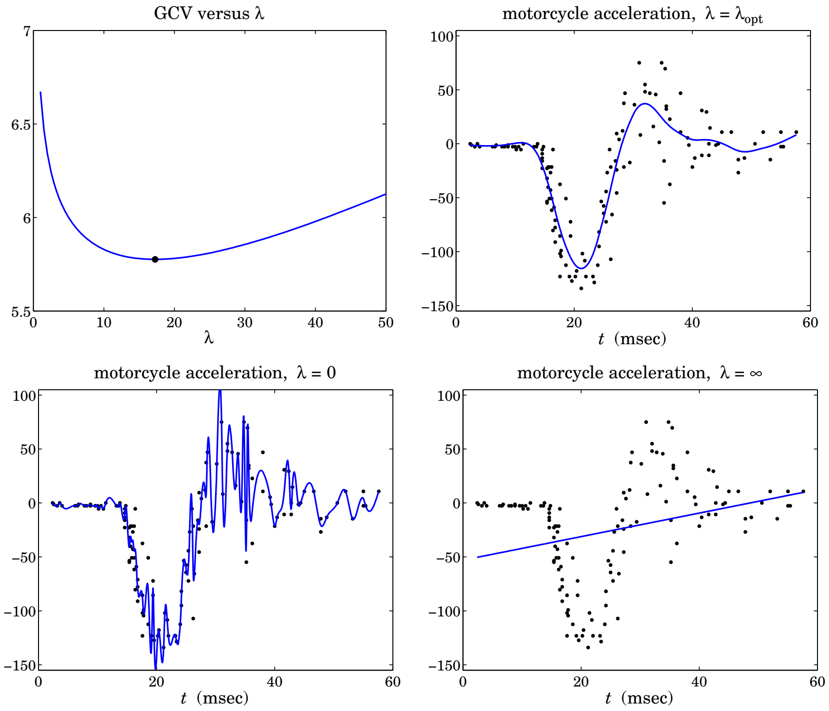


Fig. 7.9.1 Spline smoothing of motorcycle data.

The MATLAB code used for this graph was:

```
Y = loadfile('mcyc.dat');                         % load data

tobs = Y(:,1);  yobs = Y(:,2);                    % extract knot times and observations

[t,y,w] = splav(tobs,yobs);                       % average repeated observations

la1=1; la2=50; Nit=30;                            % search interval and no. of iterations
[lopt,gcvopt] = splambda(t,y,la1,la2,Nit,w);      % determine optimum λ

la = linspace(la1,la2,100);                       % evaluate GCV over λ1 ≤ λ ≤ λ2
gcv = splgcv(t,y,la,w);

figure; plot(la,gcv, lopt,gcvopt, '.');           % plot GCV versus λ
```

The upper-right graph shows the smoothing spline corresponding to $\lambda = \lambda_{\text{opt}}$, and evaluated at a uniform grid of time points. The lower two graphs depict the special cases of $\lambda = 0$ corresponding to spline interpolation, and $\lambda = \infty$ corresponding to a linear fit. The following MATLAB code generates these graphs:

```
P = splsm(t,y,lopt,w);                            % spline coefficients

ts = locgrid(t,1001);                             % grid time points
ys = splval(P,t,ts);                              % evaluate spline at grid

figure; plot(tobs,yobs,'.', ts,ys,'-');           % upper-right graph

la = 0; P = splsm(t,y,la,w);                      % spline coefficients for λ = 0
ys = splval(P,t,ts);                              % evaluate spline at grid

figure; plot(t,y,'.', ts,ys,'-');                 % lower-left graph

la = inf; P = splsm(t,y,la,w);                    % spline coefficients for λ = ∞
ys = splval(P,t,ts);

figure; plot(t,y,'.', ts,ys,'-');                % lower-right graph
```

Because the motorcycle data have repeated observations, the actual observations were replaced by their averaged values prior to evaluating the splines. The solid-line curves represent the evaluated splines, and the dots are the original data points, except for the lower-left graph in which the dots represent the averaged observations with the spline curve interpolating through them instead of the original points. □

.

Example 7.9.2: *Robust spline smoothing.* Fig. 7.9.2 shows an example of robust spline smoothing. It is the same example considered earlier in Figs. 4.5.3 and 5.6.1.

The optimum value of λ was determined by the function `splambda` to be $\lambda = 3.6562$, but neighboring value would be just as good. The left graph shows the case of ordinary spline smoothing with no robustness iterations, and the right graph, using $N_{\text{it}} = 10$ iterations. The MATLAB code generating the right graph was as follows,

```
t = (0:50)'; u = t/max(t);
x0 = (1-cos(2*pi*u))/2;                      % noise-free signal

seed=2005; randn('state',seed);
```

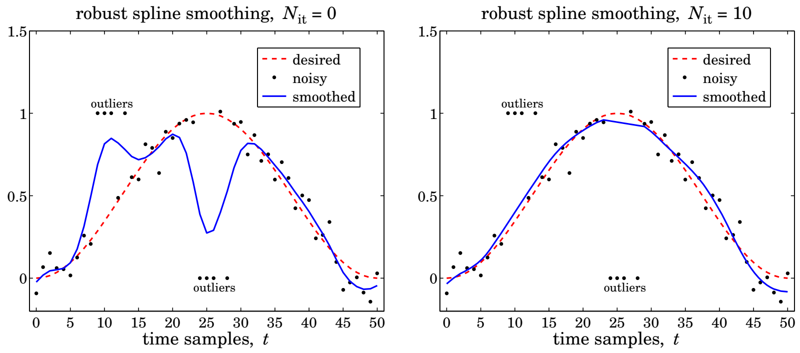


Fig. 7.9.2 Robust spline smoothing.

```
y = x0 + 0.1 * randn(size(x0));              % noisy observations

m = [-1 0 1 3];                              % outlier indices relative to n0, n1
n0=25; y(n0+m+1)=0.0;
n1=10; y(n1+m+1)=1.0;

la = splambda(t,y,1,10,30);                  % optimum λ = 3.6562

w = ones(size(t)); Nit = 10;                 % initial weights
[P,ta] = splsm2(t,y,la,w,Nit);               % robust spline smoothing
ya = P(:,1);                                 % smoothed values

plot(t,x0,'--', t,y,'.', ta,ya,'-');         % right graph
```

The optimum λ was searched for in the interval $1 \le \lambda \le 10$ calling `splambda` with 30 golden-mean iterations. The resulting knot vector t_a has length 42, while the original vector t had length 51. The missing knot times correspond to the positions of the outliers.

Example 7.9.3: *NIST ultrasonic data.* We apply spline smoothing to a nonlinear least-squares benchmark example from the NIST Statistical Reference Dataset Archives. The data file `Chwirut1.dat` is available online from the NIST web sites:

```
http://www.itl.nist.gov/div898/strd/nls/nls_main.shtml
http://www.itl.nist.gov/div898/strd/nls/data/chwirut1.shtml
```

The data are from a NIST study involving ultrasonic calibration and represent ultrasonic response versus metal distance. There are multiple observations for each distance. In fact, there are 214 observation pairs (x_n, y_n), but only 22 unique x_ns. The data have been fit by NIST using a nonlinear least squares method to a function of the form:

$$y = \frac{\exp(-b_1 x)}{b_2 + b_3 x} \qquad \text{(NIST fit)}$$

with the following fitted parameter values:

$$b_1 = 0.19027818370\,, \quad b_2 = 0.0061314004477\,, \quad b_3 = 0.0010530908399$$

The right graph in Fig. 7.9.3 compares the smoothing spline curve (solid line) with the above NIST fit (dotted line). Except for the rightmost end of the curves, the agreement is very close and the curves are almost indistinguishable.

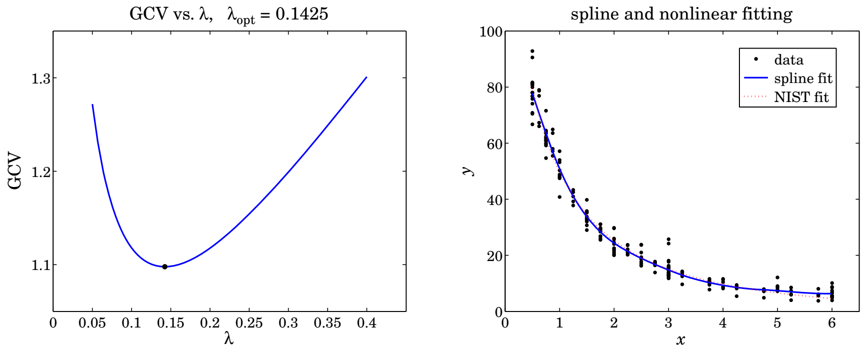


Fig. 7.9.3 Spline smoothing vs. Nonlinear fitting.

The function `splav` is called first to determine the unique x_ns and the corresponding averaged observations and their multiplicities. These are then used in `splambda` to determine the optimum GCV smoothing parameter, $\lambda_{\rm opt} = 0.1425$, which is used by the function `splsm` to perform the spline smoothing fit. The GCV is evaluated and plotted at a range of λs to illustrate its minimum. Finally, the spline is evaluated at dense grid of x-abscissas for plotting. The following code segment illustrates the computations:

```
Y = loadfile('Chwirut1.dat');                         % data file in OSP toolbox
x = Y(:,2); y = Y(:,1);                               % read (x,y) observation pairs

[x,i] = sort(x); y = y(i);                            % sort x's in increasing order

b1 = 1.9027818370E-01;   b2 = 6.1314004477E-03;   b3 = 1.0530908399E-02;

yf = exp(-b1*x)./(b2+b3*x);                           % NIST fit

[xa,ya,wa] = splav(x,y);              % unique x's, averaged observations, and multiplicities

la1=0.01; la2=1; Nit=20;                              % search interval
[lopt,gopt] = splambda(xa,ya,la1,la2,Nit,wa);         % determine optimum λ

la = linspace(0.05,0.4,51);                           % range of λ's
gcv = splgcv(xa,ya,la,wa);                            % evaluate GCV

figure; plot(la,gcv, lopt,gopt,'.');                  % left graph

P = splsm(xa,ya,lopt,wa);                             % spline smoothing coefficients
xs = locgrid(xa,200);                                 % evaluation grid of abscissas
ys = splval(P,xa,xs);                                 % evaluate spline at xs

figure; plot(x,y,'.', xs,ys,'-', x,yf,':');           % right graph
```

7.10 Problems

7.1 Show that the matrices Q, S defined in Eqs. (7.3.13) and (7.3.30) satisfy the orthogonality property $Q^T S = 0$. Assuming that the weighting diagonal matrix W has positive diagonal entries, argue that the $N\times N$ matrix $A = [W^{-1/2}Q, W^{1/2}S]$ is non-singular. Using this fact, prove the projection matrix property (7.3.31). [*Hint:* work with $A(A^T A)^{-1}A^T$.]

7.2 Show that the Euler-Lagrange equation for the variational problem (7.6.1) is:

$$\ddddot{x}(t) = \lambda^{-1}\sum_{n=0}^{N-1}\sum_{i=1}^{m_n} w_{ni}\big(y_{ni} - x(t_n)\big)\delta(t - t_n)$$

and show that it is equivalent to

$$\ddddot{x}(t) = \lambda^{-1}\sum_{n=0}^{N-1}\bar{w}_n\big(\bar{y}_n - x(t_n)\big)\delta(t - t_n)$$

where $\bar{w}_n, \bar{y}_n$ were defined in (7.6.2). This is an alternative way to establish the equivalence of the variational problems (7.6.1) and (7.6.3).

7.3 First prove the following Fourier transform pair:

$$\exp(-b|t|) \longleftrightarrow \frac{2b}{b^2 + \omega^2}$$

where b is any complex number with $\mathrm{Re}(b) > 0$. Then, use it to prove that Eqs. (7.7.4) and (7.7.5) are a Fourier transform pair. Show the same for the pair in Eq. (7.7.8).

8

Whittaker-Henderson Smoothing

8.1 Whittaker-Henderson Smoothing Methods

Whittaker-Henderson smoothing is a discrete-time version of spline smoothing for equally spaced data. Some of the original papers by Bohlmann, Whittaker, Henderson and others,[†] and their applications to trend extraction in the actuarial sciences, physical sciences, and business and finance, are given in [405–438], including Hodrick-Prescott filters in finance [439–467], and recent realizations in terms of the ℓ_1 norm [468–478], as well as extensions to seasonal data [622–625,636,638]. The performance index was defined in Eq. (7.1.2),

$$\boxed{\mathcal{J} = \sum_{n=0}^{N-1} w_n\big|y_n - x_n\big|^2 + \lambda\sum_{n=s}^{N-1}\big|\nabla^s x_n\big|^2 = \min} \tag{8.1.1}$$

where $\nabla^s x_n$ represents the backward-difference operator $\nabla x_n = x_n - x_{n-1}$ applied s times. We encountered this operation in Sec. 4.2 on minimum-R_s Henderson filters. The corresponding difference filter and its impulse response are

$$\begin{aligned} D_s(z) &= (1 - z^{-1})^s \\ d_s(k) &= (-1)^k\binom{s}{k}, \quad 0 \le k \le s \end{aligned} \tag{8.1.2}$$

For example, we have for $s = 1, 2, 3$,

$$\mathbf{d}_1 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \quad \mathbf{d}_2 = \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix}, \quad \mathbf{d}_3 = \begin{bmatrix} 1 \\ -3 \\ 3 \\ -1 \end{bmatrix}$$

Because $D_s(z)$ is a highpass filter, the performance index attempts, in its second term, to minimize the spectral energy of x_n at the high frequency end, while attempting to interpolate the noisy observations with the first term. The result is a lowpass,

[†] Bohlmann considered the case $s = 1$, Whittaker and Henderson, $s = 3$, and Hodrick-Prescott, $s = 2$.

smoothing, operation. In fact, the filter $D_s(z)$ may be replaced by any other (causal) FIR highpass filter $D(z)$, or d_n in the time domain, with a similar result. Thus, a more general version of (8.1.1) would be:

$$\mathcal{J}=\sum_{n=0}^{N-1} w_n\big|y_n-x_n\big|^2+\lambda\sum_{n=s}^{N-1}\big|d_n*x_n\big|^2=\min \tag{8.1.3}$$

where $d_n * x_n$ denotes convolution and s is the filter order, that is, we assume that the impulse response is $d_n=[d_0,d_1,\dots,d_s]$. The criteria (8.1.1) and (8.1.3) are examples of the method of *regularization*, which we discuss in more general terms in Sec. 8.6 and for ill-conditioned linear systems in particular in Sec. 15.10.

The summation limits of the second terms in Eqs. (8.1.1) and (8.1.3) restrict the convolutional operations to their steady-state range. For example, for a length-N causal input $\{x_0,x_1,\dots,x_{N-1}\}$, the s-difference filter has the full convolutional output:

$$g_n=\nabla^s x_n=\sum_{k=\max(0,n-N+1)}^{\min(s,n)} d_s(k)x_{n-k}\,,\quad 0\le n\le N-1+s \tag{8.1.4}$$

and the steady-state output (assuming $N>s$):

$$g_n=\nabla^s x_n=\sum_{k=0}^{s} d_s(k)x_{n-k}\,,\quad s\le n\le N-1 \tag{8.1.5}$$

Similarly, we have in the more general case,

$$\begin{aligned} g_n&=d_n*x_n=\sum_{k=\max(0,n-N+1)}^{\min(s,n)} d_k x_{n-k}\,,\quad 0\le n\le N-1+s\\ g_n&=d_n*x_n=\sum_{k=0}^{s} d_k x_{n-k}\,,\quad s\le n\le N-1\end{aligned} \tag{8.1.6}$$

In Sec. 4.2 we worked with the full convolutional form (8.1.4) and implemented it in a matrix form using the convolution matrix. We recall that the MATLAB functions `binom` and `diffmat` can be used to compute the impulse response $d_s(k)$ and the corresponding $(N+s)\times N$ full convolutional matrix D_s.

The filtering operation $g_n=\nabla^s x_n$, $0\le n\le N-1+s$, can be expressed vectorially as $\mathbf{g}=D_s\mathbf{x}$, where $\mathbf{x}$ is the N-dimensional input vector $\mathbf{x}=[x_0,x_1,\dots,x_{N-1}]^T$, and $\mathbf{g}=[g_0,g_1,\dots,g_{N-1+s}]^T$, the $(N+s)$-dimensional output vector. Similarly, the operation $g_n=d_n*x_n$ can be expressed as $\mathbf{g}=D_{\text{full}}\mathbf{x}$, where the $(N+s)\times N$ full convolution matrix can be constructed using `convmat`—the sparse version of `convmtx`,

```
Dfull = convmat(d,N);      % sparse full convolution matrix
```

where $\mathbf{d}=[d_0,d_1,\dots,d_s]^T$. The steady-state versions of the full convolution matrices are obtained by extracting their middle $N-s$ rows, and therefore, they have dimension $(N-s)\times N$. For example, we have for $N=5$ and $s=2$, with $\mathbf{d}=[d_0,d_1,d_2]^T$,

$$D_{\text{full}}=\left[\begin{array}{ccccc} d_0&0&0&0&0\\ d_1&d_0&0&0&0\\ \hline d_2&d_1&d_0&0&0\\ 0&d_2&d_1&d_0&0\\ 0&0&d_2&d_1&d_0\\ \hline 0&0&0&d_2&d_1\\ 0&0&0&0&d_2 \end{array}\right]\quad\Rightarrow\quad D=\begin{bmatrix} d_2&d_1&d_0&0&0\\ 0&d_2&d_1&d_0&0\\ 0&0&d_2&d_1&d_0\end{bmatrix}=\begin{bmatrix} d_2&0&0\\ d_1&d_2&0\\ d_0&d_1&d_2\\ 0&d_0&d_1\\ 0&0&d_0\end{bmatrix}^T$$

The last expression shows that the steady matrix can also be viewed as the transposed of the convolution matrix of the reversed filter with $N-s$ columns. Thus, in MATLAB two possible ways of constructing D are:

$$\boxed{\begin{array}{ll} 1) & D_{\text{full}}=\texttt{convmat}(d,N);\quad D=D_{\text{full}}(s+1:N,:)\\ 2) & D=\texttt{convmat}(\texttt{flip}(d),N-s)'; \end{array}} \tag{8.1.7}$$

For the s-difference filter, we can use the equivalent (sparse) constructions:

$$\boxed{\begin{array}{ll} 1) & D_{\text{full}}=\texttt{diffmat}(s,N);\quad D=D_{\text{full}}(s+1:N,:)\\ 2) & D=(-1)^s*\texttt{diffmat}(s,N-s)';\\ 3) & D=\texttt{diff}(\texttt{speye}(N),s); \end{array}} \tag{8.1.8}$$

where the second method is valid because the reversed binomial filter is $(-1)^s$ times the unreversed one. The third method is the fastest [428], but does not generalize to an arbitrary filter $\mathbf{d}$. As an example, we have for $N=7$, $s=2$, and $\mathbf{d}=[1,-2,1]^T$:

$$D=\begin{bmatrix} 1&-2&1&0&0&0&0\\ 0&1&-2&1&0&0&0\\ 0&0&1&-2&1&0&0\\ 0&0&0&1&-2&1&0\\ 0&0&0&0&1&-2&1\end{bmatrix}$$

The corresponding steady-state output vector $\mathbf{g}=[g_s,g_{s+1},\dots,g_{N-1}]^T$ is given by $\mathbf{g}=D\mathbf{x}$, with squared norm,

$$\sum_{n=s}^{N-1}[d_n*x_n]^2=\sum_{n=s}^{N-1}g_n^2=\mathbf{g}^T\mathbf{g}=\mathbf{x}^T(D^TD)\mathbf{x}$$

Therefore, the performance index (8.1.1) or (8.1.3) can be written compactly as:

$$\mathcal{J}=(\mathbf{y}-\mathbf{x})^TW(\mathbf{y}-\mathbf{x})+\lambda\mathbf{x}^T(D^TD)\mathbf{x}=\min \tag{8.1.9}$$

where W is the diagonal matrix of the weights, $W=\text{diag}([w_0,w_1,\dots,w_{N-1}])$. The optimum solution is obtained by setting the gradient with respect to $\mathbf{x}$ to zero,

$$\frac{\partial\mathcal{J}}{\partial\mathbf{x}}=-2W(\mathbf{y}-\mathbf{x})+2\lambda(D^TD)\mathbf{x}=0\quad\Rightarrow\quad(W+\lambda D^TD)\mathbf{x}=W\mathbf{y}$$

with solution, which may be regarded as the estimate of $\mathbf{x}$ in the signal model $\mathbf{y} = \mathbf{x} + \mathbf{v}$,

$$\boxed{\hat{\mathbf{x}} = (W + \lambda D^T D)^{-1} W \mathbf{y}} \tag{8.1.10}$$

The matrix D plays the same role as the matrix Q^T in the spline smoothing case, but for equally-spaced data. As was the case in Sec. 4.2, the matrix $D^T D$ is essentially equivalent to the $(2s)$-differencing operator ∇^{2s}, after ignoring the first s and last s rows. For example, we have for $N = 7$ and $s = 2$,

$$D^T D = \left[\begin{array}{rrrrrrr} 1 & -2 & 1 & 0 & 0 & 0 & 0 \\ -2 & 5 & -4 & 1 & 0 & 0 & 0 \\ \hline 1 & -4 & 6 & -4 & 1 & 0 & 0 \\ 0 & 1 & -4 & 6 & -4 & 1 & 0 \\ 0 & 0 & 1 & -4 & 6 & -4 & 1 \\ \hline 0 & 0 & 0 & 1 & -4 & 5 & -2 \\ 0 & 0 & 0 & 0 & 1 & -2 & 1 \end{array}\right]$$

where we recognize the expansion coefficients $(1 - z^{-1})^4 = 1 - 4z^{-1} + 6z^{-2} - 4z^{-3} + z^{-4}$.

The $N \times N$ matrix $(W + \lambda D^T D)$ is sparse and banded with bandwidth $2s + 1$, and therefore, MATLAB solves Eq. (8.1.10) very efficiently by default (as long as it is implemented by the backslash operator). The function `whsm` implements Eq. (8.1.10):

```
x = whsm(y,lambda,s,w);          % Whittaker-Henderson smoothing
```

where method (2) is used internally to compute D, and `w` is the vector of weights, which defaults to unity. The function `whgen` is the generalized version that uses an arbitrary highpass filter $\mathbf{d}$, whose steady convolution matrix D is also computed by method (2):

```
x = whgen(y,lambda,d,w);          % generalized Whittaker-Henderson smoothing
```

Denoting the "hat" filtering matrix $H = (W + \lambda D^T D)^{-1} W$, and defining the error $\mathbf{e} = \mathbf{y} - \hat{\mathbf{x}} = (I - H)\mathbf{y}$, we may define a generalized cross-validation criterion for determining the smoothing parameter λ, which is analogous to that of Eq. (7.5.5):

$$\mathrm{GCV}(\lambda) = \frac{\mathbf{e}^T W \mathbf{e}}{\left[\mathrm{tr}(I - H)\right]^2} = \min \tag{8.1.11}$$

The function `whgcv` calculates it at any vector of λ's and finds the corresponding optimum:

```
[gcv,lopt] = whgcv(y,la,s,w);          % Whittaker-Henderson GCV evaluation
```

The GCV criterion should be used with some caution because it suffers from the same problem, as in the spline case, of typically underestimating the proper value of λ.

The Whittaker-Henderson method was compared to the local polynomial and minimum roughness filters in Examples 3.9.2 and 4.2.1. Some additional examples are discussed below.

Example 8.1.1: *NIST ENSO data.* We apply the Whittaker-Henderson (WH) smoothing method to the ENSO data which are another benchmark example in the NIST Statistical Reference Dataset Archives, and original reference [1240]. The data file `ENSO.dat` is available online from the NIST web sites:

```
http://www.itl.nist.gov/div898/strd/nls/nls_main.shtml
http://www.itl.nist.gov/div898/strd/nls/data/enso.shtml
```

The data represent the monthly averaged atmospheric pressure differences between Easter Island and Darwin, Australia. In the nonlinear NIST fit, the data are fitted to three sinusoids of unknown amplitudes and frequencies, except that one of the sinusoids is kept at the annual frequency. There are three significant cycles at 12, 26, and 44 months.

The upper-left graph of Fig. 8.1.1 compares the NIST fit with the Whittaker-Henderson method. We used $s = 3$ and smoothing parameter $\lambda_{\mathrm{opt}} = 6.6$. which was determined by the GCV function `whgcv`.

The lower-left graph shows the corresponding periodogram spectra plotted versus period in units of months/cycle. The three dominant peaks are evident. In the spectrum graphs, the digital frequency is $\omega = 2\pi f$ rads/month, with f measured in cycles/month, and with the corresponding period $p = 1/f$ measured in months/cycle.

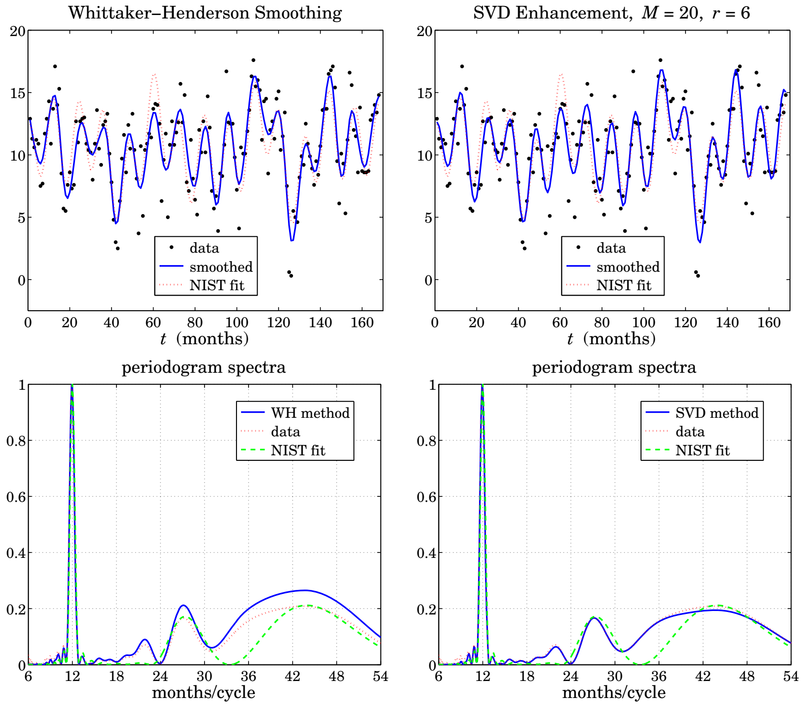


Fig. 8.1.1 Smoothed ENSO signal and spectra.

The time-domain WH signal agrees fairly well with the NIST fit. We note that in the places where the two disagree, the WH fit appears to be a better representation of the noisy data.

The upper-right graph shows the application of the SVD enhancement method, which typically works well for sinusoids in noise. The embedding dimension was $M = 20$ and the assumed rank $r = 6$ (three real sinusoids are equivalent to six complex ones.) The lower-right graph shows the corresponding spectral peaks. The following MATLAB code illustrates the generation of the four graphs:

```
Y = loadfile('ENSO.dat');                          % data file in OSP toolbox
y = Y(:,1); t = Y(:,2);                            % extract data signal

b1 = 1.0510749193E+01; b2 = 3.0762128085E+00; b3 = 5.3280138227E-01;
b4 = 4.4311088700E+01; b5 =-1.6231428586E+00; b6 = 5.2554493756E-01;
b7 = 2.6887614440E+01; b8 = 2.1232288488E-01; b9 = 1.4966870418E+00;

yf = b1 + b2*cos(2*pi*t/12) + b3*sin(2*pi*t/12) + b5*cos(2*pi*t/b4) ...
    + b6*sin(2*pi*t/b4) + b8*cos(2*pi*t/b7) + b9*sin(2*pi*t/b7);

s=3; la = linspace(2,10,100);                      % search range for λ
[gcv,lopt]=whgcv(y,la,s);                          % λopt = 6.6

yw = whsm(y,lopt,s);                               % WH smoothing method
M=20; r=6; ye = svdenh(y,M,r);                     % SVD enhancement method

figure; plot(t,y,'.', t,yw,'-', t,yf,':');         % upper-left graph
figure; plot(t,y,'.', t,ye,'-', t,yf,':');         % upper-right graph

p = linspace(6,54, 481); w = 2*pi./p;              % period in months/cycle

Sy = abs(freqz(zmean(y), 1, w)).^2;    Sy = Sy/max(Sy);        % spectra
Sf = abs(freqz(zmean(yf), 1, w)).^2;   Sf = Sf/max(Sf);
Sw = abs(freqz(zmean(yw), 1, w)).^2;   Sw = Sw/max(Sw);
Se = abs(freqz(zmean(ye), 1, w)).^2;   Se = Se/max(Se);

figure; plot(p,Sw, p,Sy,':', p,Sf,'--');           % lower-left graph
figure; plot(p,Se, p,Sy,':', p,Sf,'--');           % lower-right graph
```

The b_i parameters and the signal `yf` represent the NIST fit. Anticipating the three relevant peaks, the spectra were computed only over the period range $6 \le p \le 54$ months. The function `zmean` removes the mean of the signal so that the spectrum is not masked by the DC component. □

8.2 Regularization Filters

Most of the results of the spline smoothing case carry over to the discrete case. For example, we may obtain an equivalent digital filter by taking the signals to be double-sided and infinite. Using the Parseval identity, the performance index (8.1.3) becomes:

$$\begin{aligned}\mathcal{J} &= \sum_{n=-\infty}^{\infty} |y_n - x_n|^2 + \lambda \sum_{n=-\infty}^{\infty} |d_n * x_n|^2 = \\ &= \int_{-\pi}^{\pi} \big|Y(\omega)-X(\omega)\big|^2 \frac{d\omega}{2\pi} + \lambda \int_{-\pi}^{\pi} \big|D(\omega)X(\omega)\big|^2 \frac{d\omega}{2\pi} = \min\end{aligned} \tag{8.2.1}$$

where $D(\omega)$ is the frequency response[†] of the filter d_n, and we assumed unity weights, $w_n = 1$. The vanishing of the functional derivative of $\mathcal{J}$ with respect to $X^*(\omega)$,

$$\frac{\delta\mathcal{J}}{\delta X^*(\omega)} = X(\omega) - Y(\omega) + \lambda |D(\omega)|^2 X(\omega) = 0 \tag{8.2.2}$$

gives the effective equivalent smoothing filter $H(\omega) = X(\omega)/Y(\omega)$:

$$H(\omega) = \frac{1}{1 + \lambda|D(\omega)|^2} \tag{8.2.3}$$

The corresponding z-domain transfer function is obtained by noting that for real-valued d_n, we have $|D(\omega)|^2 = D(z)D(z^{-1})$, where $z = e^{j\omega}$, so that,

$$H(z) = \frac{1}{1 + \lambda D(z)D(z^{-1})} \tag{8.2.4}$$

Such "recursive regularization filters" have been considered in [440]. In particular, for the s-difference filter $D_s(z) = (1 - z^{-1})^s$, we have:

$$\boxed{H(z) = \frac{1}{1 + \lambda(1 - z^{-1})^s(1 - z)^s}} \quad \text{(Whittaker-Henderson filter)} \tag{8.2.5}$$

Similarly, Eq. (8.2.2) can be written in the z-domain and converted back to the time domain. Noting that $D_s(z)D_s(z^{-1}) = (1-z^{-1})^s(1-z)^s = (-1)^s z^s (1-z^{-1})^{2s} = (-1)^s z^s D_{2s}(z)$, we have:

$$X(z) - Y(z) + \lambda(-1)^s z^s (1 - z^{-1})^{2s} X(z) = 0\,, \qquad \text{or,}$$

$$(-1)^s z^s D_{2s}(z) X(z) = \lambda^{-1}\big(Y(z) - X(z)\big)\,,$$

resulting in the time-domain $(2s)$-difference equation:

$$(-1)^s \nabla^{2s} x_{n+s} = \lambda^{-1}(y_n - x_n) \tag{8.2.6}$$

In the early years, some ingenious methods were developed for solving this type of equation [407–416]. Noting that $|D_s(\omega)| = |1 - e^{-j\omega}|^s = 2^s \big|\sin(\omega/2)\big|^s$, we obtain the frequency response of (8.2.5):

$$H(\omega) = \frac{1}{1 + \lambda\left|2\sin\dfrac{\omega}{2}\right|^{2s}} \tag{8.2.7}$$

The complementary highpass filter $H_c(z) = 1 - H(z)$ extracts the error residual component from the observations y_n, that is, $e_n = y_n - x_n$, or in the z-domain, $E(z) = Y(z) - X(z) = Y(z) - H(z)Y(z) = H_c(z)Y(z)$. Its transfer function and frequency response are given by:

$$H_c(z) = \frac{\lambda(1 - z^{-1})^s(1 - z)^s}{1 + \lambda(1 - z^{-1})^s(1 - z)^s}\,, \quad H_c(\omega) = \frac{\lambda\left|2\sin\dfrac{\omega}{2}\right|^{2s}}{1 + \lambda\left|2\sin\dfrac{\omega}{2}\right|^{2s}} \tag{8.2.8}$$

Fig. 8.2.1 shows a plot of $H(\omega)$ and $H_c(\omega)$ for $s = 1, 2, 3$ and the two values of the smoothing parameter $\lambda = 5$ and $\lambda = 50$. Increasing λ narrows the response of the lowpass filter and widens the response of the highpass one.

[†] Here, ω is the digital frequency in units of radians per sample.

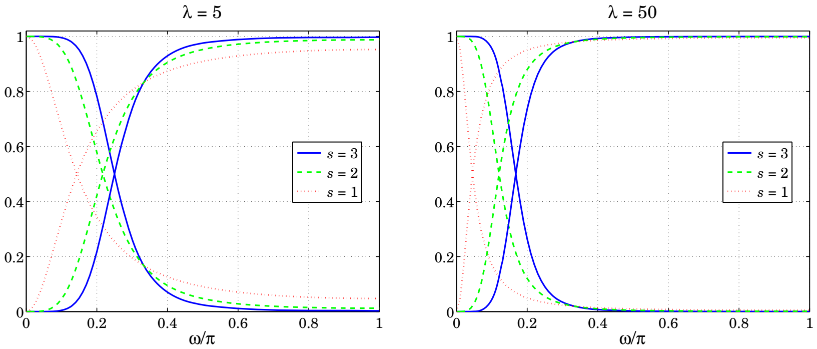


Fig. 8.2.1 Frequency responses of Whittaker-Henderson filters.

8.3 *Hodrick-Prescott Filters*

In macroeconomic applications such as extracting business cycles from GDP data, the standard signal model $y_n = x_n + v_n$ is interpreted to consist of a long-term trend represented by x_n and a shorter-term cyclical component v_n. The filters $H(z)$ and $H_c(z)$ extract the trend and cyclical components, respectively.

The use of Whittaker-Henderson smoothing with $s = 2$ has been advocated by Hodrick and Prescott [439] and has become standard in such applications. The Whittaker-Henderson filters are referred to as *Hodrick-Prescott filters* and there is a very large literature on the subject and on the use of other types of bandpass filters for extracting business cycles, a subset of which is [439–465].

As is the case in typical filter design, the filter parameter λ can be fixed by specifying a desired value for the filter's cutoff frequency ω_c corresponding to some standardized value of the gain. For the lowpass filter we have for general s, the condition:

$$\frac{1}{1+\lambda\left|2\sin\dfrac{\omega_c}{2}\right|^{2s}} = G_c \tag{8.3.1}$$

where G_c is desired value of the gain. The 3-dB cutoff frequency ω_c corresponds to $G_c = 1/\sqrt{2}$. In macroeconomic applications, the 6-dB frequency is often used, corresponding to the choice $G_c = 1/2$. For the highpass case, measuring the gain G_c relative to that at the Nyquist frequency $\omega = \pi$, we have the condition:

$$\frac{\lambda\left|2\sin\dfrac{\omega_c}{2}\right|^{2s}}{1+\lambda\left|2\sin\dfrac{\omega_c}{2}\right|^{2s}} = G_c\,\frac{2^{2s}\lambda}{1+2^{2s}\lambda} \tag{8.3.2}$$

Typically, *business cycles* are defined [446] as having frequency components with periods between 6 and 32 quarters (1.5 to 8 years). A bandpass filter with a passband $[\omega_1, \omega_2] = [2\pi/32,\ 2\pi/6]$ radians/quarter would extract such cycles.

The Hodrick-Prescott highpass filter $H_c(\omega)$ must therefore have a cutoff frequency of about $\omega_c = \omega_1 = 2\pi/32$. Hodrick-Prescott advocate the use of $\lambda = 1600$ for quarterly data. Interestingly, the values of $\lambda = 1600$ and $\omega_c = 2\pi/32$ rads/quarter, correspond to almost a 3-dB gain. Indeed, the gain calculated from Eq. (8.3.2) with $s = 2$ turns out to be $G_c = 0.702667 \equiv -3.065$ dB.

Using the same ω_c and G_c, but different values of s requires adjusting the value of λ. For example, solving Eq. (8.3.2) for λ with $s = 1, 2, 3$ gives:

$$\lambda = 1600\ (s=2),\quad \lambda = 60.654\ (s=1),\quad \lambda = 41640\ (s=3) \tag{8.3.3}$$

Similarly, the value of λ must be adjusted if the sampling frequency is changed. For example, the same cutoff frequency expressed in different units is:

$$\omega_c = \frac{2\pi}{32}\,\frac{\text{radians}}{\text{quarter}} = \frac{2\pi}{8}\,\frac{\text{radians}}{\text{year}} = \frac{2\pi}{96}\,\frac{\text{radians}}{\text{month}} \tag{8.3.4}$$

The above value $G_c = 0.702667$ used in (8.3.2) with $s = 2$ then gives the following values of λ for quarterly, yearly, and monthly sampled data:

$$\lambda = 1600\ (\text{quarterly}),\quad \lambda = 6.677\ (\text{yearly}),\quad \lambda = 128878\ (\text{monthly}) \tag{8.3.5}$$

Similarly, using the slightly more exact value $G_c = 1/\sqrt{2}$ and $s = 2$ gives:

$$\lambda = 1634.5\ (\text{quarterly}),\quad \lambda = 6.822\ (\text{yearly}),\quad \lambda = 131659\ (\text{monthly}) \tag{8.3.6}$$

There is not much agreement as to the values of λ to be used for annual and monthly data. Two other sets of values are as follows, with the first being used by the European Central Bank and the second recommended by [456,463],

$$\begin{array}{ll} \lambda = 1600/4^2 = 100\ (\text{yearly}), & \lambda = 1600\times 3^2 = 14400\ (\text{monthly}) \\ \lambda = 1600/4^4 = 6.25\ (\text{yearly}), & \lambda = 1600\times 3^4 = 129600\ (\text{monthly}) \end{array} \tag{8.3.7}$$

The latter choice is essentially the same as that of Eq. (8.3.5) based on the criterion (8.3.2). Indeed, for small ω_c, we may make the approximation $2\sin(\omega_c/2)\approx\omega_c$. Since $2^{2s}\lambda$ is typically much larger than unity, the right-hand side of Eq. (8.3.2) can be replaced by G_c, resulting in the following approximate solution, which turns out to be valid up to about $\omega_c \le 0.3\pi$,

$$\frac{\lambda\omega_c^{2s}}{1+\lambda\omega_c^{2s}} = G_c \quad\Rightarrow\quad \lambda = \frac{G_c}{(1-G_c)\,\omega_c^{2s}} \tag{8.3.8}$$

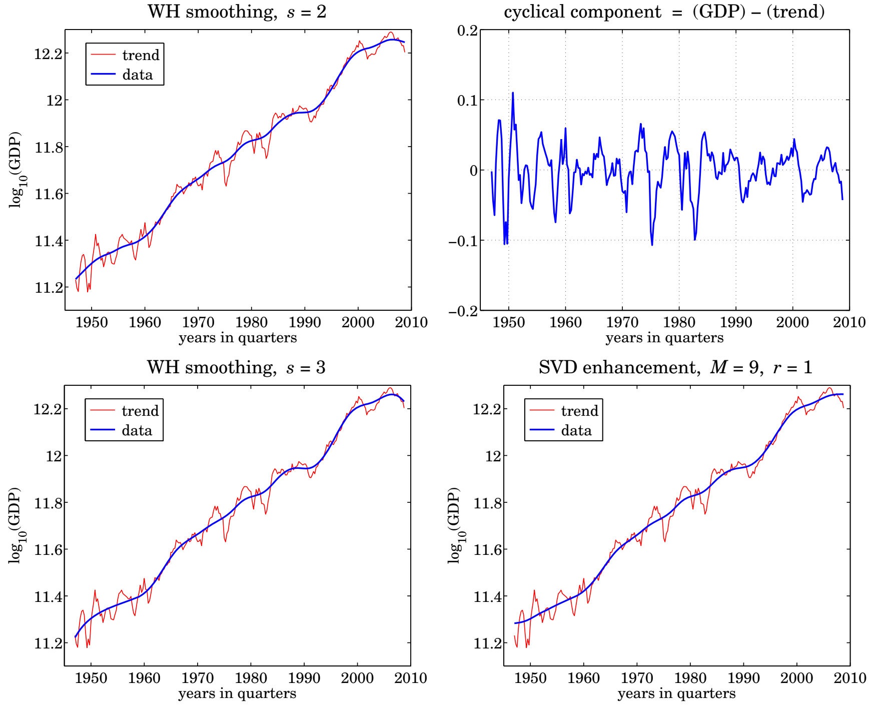
If in this formula, we adjust G_c to get $\lambda = 1600$ at $\omega_c = 2\pi/32$, we find $G_c = 0.70398 \equiv -3.049$ dB, which in turn generates the second set of values in Eq. (8.3.7).

Example 8.3.1: *US GDP for investment.* A protypical example is the application of the Hodrick-Prescott filter to the US GDP. Fig. 8.3.1 shows the real gross domestic product in chained (2000) dollars from private domestic investment, seasonally adjusted at annual rates. The data can be retrieved (as Table 1.1.6) from the BEA web sites:

```
http://www.bea.gov/
http://www.bea.gov/national/nipaweb/Index.asp
```

The signal to be smoothed is the log of the GDP, that is, $y = \log_{10}(\text{GDP})$, and the ordinate units are such that $y = 12$ corresponds to GDP $= 10^{12}$, or, one trillion dollars. The data are quarterly and span the years 1947–2008.

The upper-left graph shows the raw data and the WH-smoothed signal computed with $s = 2$ and $\lambda = 1600$, which as we mentioned above correspond to an approximate 3-dB cutoff frequency of 32 quarters. The upper-right graph shows the residual cyclical component. Its deviations above or below zero indicate the business cycles.


WH smoothing, $s = 2$

cyclical component = (GDP) – (trend)

WH smoothing, $s = 3$

SVD enhancement, $M = 9$, $r = 1$

Fig. 8.3.1 U.S. quarterly GDP in private investment, 1947–2008.

For comparison, the left-bottom graph shows the WH-smoothed signal with $s = 3$ and $\lambda = 41640$ adjusted to match the same 3-dB cutoff frequency as the $s = 2$ case, see Eq. (8.3.3). The lower-right graph shows the smoothed trend from the SVD enhancement method applied with embedding dimension $M = 9$ and rank $r = 1$. The following MATLAB code illustrates the generation of the four graphs:

```
Y = loadfile('USGDP_Inv.dat');                       % data file in OSP toolbox
y = log10(Y(:,2) * 1e9);                             % Y was in billions
t = taxis(y,4,1947);                                 % t-axis in quarters since 1947

s = 2; la = 1600; yt = whsm(y,la,s);                 % WH smoothing with s = 2
figure; plot(t,yt,'-', t,y,'--');                    % upper-left graph
```

```
yc = y-yt;                                           % cyclical component
figure; plot(t,yc, '-');                             % upper-right graph

s = 3; la = 41640.16; yt = whsm(y,la,s);             % WH smoothing with s = 3
figure; plot(t,yt,'-', t,y,'--');                    % bottom-left graph

M=9; r=1; ye = svdenh(y,M,r);                        % SVD enhancement method
figure; plot(t,ye,'-', t,y,'--');                    % bottom-right graph
```

Except near the end-points, the smoothed trend for $s = 3$ is virtually indistinguishable from the $s = 2$ case, and therefore, it would lead to the same prediction of business cycles. The SVD trend is also very comparable. □

8.4 Poles and Impulse Response

Because of the invariance under the substitution $z \to z^{-1}$, the $2s$ poles of the filter $H(z)$ of Eq. (8.2.5) come in two groups with s poles inside the unit circle and their reciprocals outside. It follows that $H(z)$ can be expressed in the factored form:

$$H(z) = \frac{1}{1+\lambda(1-z^{-1})^s(1-z)^s} = \prod_{k=1}^{s}\left[\frac{(1-z_k)^2}{(1-z_k z^{-1})(1-z_k z)}\right] \tag{8.4.1}$$

where z_k, $k = 1, 2, \dots, s$, denote the s poles inside the unit circle. The numerator factors $(1-z_k)^2$ ensure that the right-hand side has unity gain at DC $(z = 1)$, as does the left-hand side. The stable impulse response is double-sided and can be obtained by performing an inverse z-transform with the unit circle as the inversion contour [12]:

$$h_n = \oint_{\text{u.c.}} H(z)\, z^n \frac{dz}{2\pi j z}\,, \quad -\infty < n < \infty \tag{8.4.2}$$

Inserting the factored form (8.4.1) into (8.4.2), we find

$$h_n = \sum_{k=1}^{s} A_k z_k^{|n|}\,, \quad -\infty < n < \infty \tag{8.4.3}$$

where the coefficients A_k are given by

$$A_k = \left(\frac{1-z_k}{1+z_k}\right)\prod_{\substack{i=1\\ i\neq k}}^{s}\left[\frac{(1-z_i)^2}{(1-z_i z_k^{-1})(1-z_i z_k)}\right], \quad k = 1, 2, \dots, s \tag{8.4.4}$$

The poles can be obtained in the form $z_k = e^{j\omega_k}$, where ω_k are the complex frequencies of the denominator, that is, the frequencies that are solutions of the equation:

$$1 + \lambda\left[2\sin\frac{\omega}{2}\right]^{2s} = 0 \tag{8.4.5}$$

where we note that even though ω is complex, we still have $(1-z^{-1})(1-z) = 4\sin^2(\omega/2)$ for $z = e^{j\omega}$. The solution of Eq. (8.4.5) is straightforward. The s frequencies ω_k that lead to poles z_k that are inside the unit circle can be parametrized as follows:

$$\left[2\sin\frac{\omega}{2}\right]^{2s} = \frac{-1}{\lambda} = \frac{e^{j\pi(2k-1)}}{\lambda} \quad\Rightarrow\quad \sin\frac{\omega_k}{2} = \frac{e^{j\pi(2k-1)/(2s)}}{2\lambda^{1/(2s)}}\,, \quad k = 1, 2, \dots, s$$

Thus, the desired set of poles are:

$$z_k = e^{j\omega_k}\,, \quad \omega_k = 2\arcsin\left[\frac{e^{j\theta_k}}{2\lambda^{1/(2s)}}\right], \quad \theta_k = \frac{\pi(2k-1)}{2s}\,, \quad k = 1, 2, \dots, s \tag{8.4.6}$$

If s is even, then the z_k (and the coefficients A_k) come in conjugate pairs. If s is odd, then the zero at $k = (s+1)/2$ is real and the rest come in conjugate pairs. In either case, h_n given by (8.4.3) is real-valued and decays exponentially from either side of the time axis. The MATLAB function `whimp` calculates h_n at any vector of ns and also produces the poles and residues z_k, A_k, $k = 1, 2, \dots, s$,

```
[h,z,A] = whimp(lambda,s,n);        % Whittaker-Henderson impulse response and poles
```

Fig. 8.4.1 shows the impulse responses for $s = 1, 2, 3$ with λs chosen as in (8.3.3) so that the (complementary) filters have the same 3-dB cutoff frequency. The impulse response of the complementary filter is $h_c(n) = \delta(n) - h(n)$. Therefore, the three responses will have roughly the same time width.

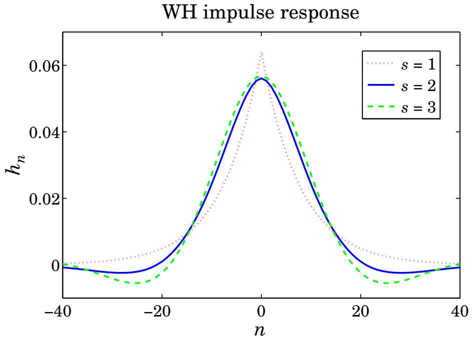


Fig. 8.4.1 Impulse responses of Whittaker-Henderson filters.

8.5 Wiener Filter Interpretation

Finally, we note that, as in the spline smoothing case, the equivalent filter $H(z)$ can be regarded as the optimum unrealizable Wiener filter for estimating x_n from the observations y_n in the signal model:

$$y_n = x_n + v_n\,, \quad \nabla^s x_n = w_n \tag{8.5.1}$$

where v_n, w_n are mutually uncorrelated white noise signals with variances σ_v^2, σ_w^2. Indeed, the transformed signals

$$\bar{x}_n = \nabla^s x_n = w_n\,, \quad \bar{y}_n = \nabla^s y_n = w_n + \nabla^s v_n$$

are stationary and it follows from the results of [643–649] that the optimum Wiener filter will be in this case:

$$H(z) = \frac{S_{\bar{x}\bar{y}}(z)}{S_{\bar{y}\bar{y}}(z)} = \frac{S_{ww}(z)}{S_{ww}(z) + D_s(z)D_s(z^{-1})S_{vv}(z)} = \frac{\sigma_w^2}{\sigma_w^2 + \sigma_v^2 D_s(z)D_s(z^{-1})} \tag{8.5.2}$$

Thus, we may identify $\lambda = \sigma_v^2/\sigma_w^2$. Using such a model, the optimum estimation filter based on a finite, length-N, set of observations y_n can be implemented efficiently as a Kalman filter smoother requiring $O(N)$ operations. On the other hand, the solution of the matrix equation (8.1.10) is just as efficient due to the sparse and banded nature of the linear system.

8.6 Regularization and Kernel Machines

Regularization was initially invented as a method for solving ill-posed, inconsistent, overdetermined, and ill-conditioned inverse problems. Recently it has been applied also to support-vector machines and kernel methods for machine learning. There is a huge literature on the subject, a small subset of which is [480–519]. Here, we present a short discussion with particular emphasis on deconvolution and kernel regression methods.

Both spline and Whittaker-Henderson smoothing are examples of regularization. The performance index Eq. (8.1.3) can be generalized further to cover the case of deconvolution, or inverse filtering,

$$\mathcal{J} = \sum_n |y_n - f_n * x_n|^2 + \lambda \sum_n |d_n * x_n|^2 = \min \tag{8.6.1}$$

where f_n and d_n are FIR filters. This attempts to solve $y_n = f_n * x_n$ for x_n by deconvolving the effect of f_n. We may write (8.6.1) in a compact matrix form using the convolution matrices F, D of the two filters:

$$\mathcal{J} = \|\mathbf{y} - F\mathbf{x}\|^2 + \lambda\|D\mathbf{x}\|^2 = (\mathbf{y} - F\mathbf{x})^T(\mathbf{y} - F\mathbf{x}) + \lambda\mathbf{x}^T(D^T D)\mathbf{x} = \min \tag{8.6.2}$$

The solution is obtained from the gradient,

$$\frac{\partial\mathcal{J}}{\partial\mathbf{x}} = -2F^T(\mathbf{y} - F\mathbf{x}) + 2\lambda D^T D\mathbf{x} = 0\,, \quad \text{or,}$$

$$\hat{\mathbf{x}} = (F^T F + \lambda D^T D)^{-1} F^T \mathbf{y} \tag{8.6.3}$$

The problem (8.6.2) is of course much more general than inverse filtering. The method is known as *Tikhonov regularization* and as *ridge regression*. The linear system $\mathbf{y} = F\mathbf{x}$ may in general be overdetermined, or underdetermined, or rank defective. We discuss such cases in Chap. 15. To simplify the discussion, we assume here that the linear system is either square and invertible or overdetermined but F having full rank. For $\lambda = 0$, we obtain $\hat{\mathbf{x}} = (F^T F)^{-1} F^T \mathbf{y}$, which is recognized as the unique pseudoinverse least-squares solution. In the square case, we have $\hat{\mathbf{x}} = F^{-1}\mathbf{y}$. We are envisioning a signal model of the form $\mathbf{y} = F\mathbf{x} + \mathbf{v}$ and the objective is to determine an estimate of $\mathbf{x}$. We have then,

$$\hat{\mathbf{x}} = F^{-1}\mathbf{y} = F^{-1}(F\mathbf{x} + \mathbf{v}) = \mathbf{x} + F^{-1}\mathbf{v} \tag{8.6.4}$$

A potential problem with this estimate is that if F is ill-conditioned with a large condition number—a common occurrence in practice—the resulting inverse-filtered noise component $\mathbf{u} = F^{-1}\mathbf{v}$ may be magnified to such an extent that it will mask the desired term $\mathbf{x}$, rendering the estimate $\hat{\mathbf{x}}$ useless. The same can happen in the overdetermined case. The presence of the regularization term helps in this regard by providing a more well-conditioned inverse. For the deconvolution problem, one typically selects D to be the unit matrix, $D = I$, leading to the solution,

$$\hat{\mathbf{x}} = (F^T F + \lambda I)^{-1} F^T \mathbf{y} \tag{8.6.5}$$

To see how regularization improves the condition number, let $\lambda_{\max}, \lambda_{\min}$ be the maximum and minimum eigenvalues of $F^T F$. Then, the condition numbers of $F^T F$ and $F^T F + \lambda I$ are $\lambda_{\max}/\lambda_{\min}$ and $(\lambda_{\max} + \lambda)/(\lambda_{\min} + \lambda)$. A highly ill-conditioned problem would have $\lambda_{\min} \ll \lambda_{\max}$. It is straightforward to verify that the larger the λ, the more the condition number of the regularized matrix is reduced:

$$\lambda \gg 1 \quad \Rightarrow \quad \frac{\lambda_{\max} + \lambda}{\lambda_{\min} + \lambda} \ll \frac{\lambda_{\max}}{\lambda_{\min}}$$

For example, if $\lambda_{\min} = 10^{-3}$ and $\lambda_{\max} = 10^{3}$, we have $\lambda_{\max}/\lambda_{\min} = 10^{6}$, but the regularized version $(\lambda_{\max} + \lambda)/(\lambda_{\min} + \lambda)$ takes approximately the values 11, 2, 1.1, for $\lambda = 10^2, 10^3, 10^4$, respectively.

Regularization is not without problems. For noisy data the basic tradeoff is that improving the condition number by increasing λ causes more distortion and smoothing of the desired signal component $\mathbf{x}$. As usual, choosing the proper value of λ is more of an art than science and requires some trial-and-error experimentation. The method of cross-validation can also be applied [368] as a guide.

Other choices for D, for example differencing matrices, are used in applications such as edge-preserving deblurring of images. We discuss deconvolution and inverse filter design further in Sections 12.14 and 15.11.

Next, we consider briefly the connection of regularization to machine learning and reproducing kernel Hilbert spaces. We recall that the objective of the spline smoothing performance index,

$$\mathcal{J} = \sum_{n=0}^{N-1} \big[y_n - f(t_n)\big]^2 + \lambda \int_{t_a}^{t_b} [\ddot{f}(t)]^2 dt = \min \tag{8.6.6}$$

was to “learn” the unknown function $f(t)$ from a finite subset of N noisy observations $y_n = f(t_n) + v_n$, $n = 0, 1, \dots, N-1$. The concept can be generalized from functions of time to multivariable functions of some independent variable, say $\mathbf{x}$, such as three-dimensional space. The observed data samples are of the form $y_n = f(\mathbf{x}_n) + v_n$, and the objective is to learn the unknown function $f(\mathbf{x})$. The performance index is replaced by,

$$\mathcal{J} = \sum_{n=0}^{N-1} \big[y_n - f(\mathbf{x}_n)\big]^2 + \lambda \|f\|^2 = \min \tag{8.6.7}$$

where $\|f\|$ is an appropriate norm that depends on the approach one takes to the minimization problem. One possible and very successful approach is to use a *neural network* to model the unknown function $f(\mathbf{x})$. In this case the regularization norm depends on the parameters of the neural network and its assumed structure (typically a single-hidden layer is sufficient.)

The reproducing kernel approach that we discuss here is to assume that $f(\mathbf{x})$ can be represented as a linear combination of a finite or infinite set of nonlinear basis functions $\phi_i(\mathbf{x})$, $i = 1, 2, \dots, M$, where for now we will assume that M is finite,

$$f(\mathbf{x}) = \sum_{i=1}^{M} \phi_i(\mathbf{x}) c_i = \boldsymbol{\phi}^T(\mathbf{x})\mathbf{c}\,, \qquad \boldsymbol{\phi}(\mathbf{x}) = \begin{bmatrix} \phi_1(\mathbf{x}) \\ \phi_2(\mathbf{x}) \\ \vdots \\ \phi_M(\mathbf{x}) \end{bmatrix}, \quad \mathbf{c} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_M \end{bmatrix} \tag{8.6.8}$$

This is analogous to the approach of Chap. 3 where $f(t)$ was modeled as a polynomial in t and expanded in the monomial basis functions $s_i(t) = t^i$. Here, we define the regularization norm in terms of the coefficients c_i as follows:

$$\|f\|^2 = \sum_{i=1}^{M} \frac{c_i^2}{\lambda_i} = \mathbf{c}^T \Lambda^{-1} \mathbf{c} \tag{8.6.9}$$

where λ_i is a set of some given positive coefficients, and $\Lambda = \text{diag}([\lambda_1, \lambda_2, \dots, \lambda_M])$. The function values at the N observation points are $f(\mathbf{x}_n) = \boldsymbol{\phi}^T(\mathbf{x}_n)\mathbf{c}$, and can be arranged into an N-dimensional column vector:

$$\mathbf{f} = \begin{bmatrix} f(\mathbf{x}_0) \\ \vdots \\ f(\mathbf{x}_n) \\ \vdots \\ f(\mathbf{x}_{N-1}) \end{bmatrix} = \Phi\mathbf{c}\,, \qquad \Phi = \begin{bmatrix} \boldsymbol{\phi}^T(\mathbf{x}_0) \\ \vdots \\ \boldsymbol{\phi}^T(\mathbf{x}_n) \\ \vdots \\ \boldsymbol{\phi}^T(\mathbf{x}_{N-1}) \end{bmatrix} \tag{8.6.10}$$

where Φ has dimension $N{\times}M$. Thus, the performance index can be written compactly,

$$\boxed{\mathcal{J} = (\mathbf{y} - \Phi\mathbf{c})^T(\mathbf{y} - \Phi\mathbf{c}) + \lambda\mathbf{c}^T \Lambda^{-1}\mathbf{c} = \min} \tag{8.6.11}$$

The solution for the optimum coefficients $\mathbf{c}$ is obtained by setting the gradient to zero:

$$\frac{\partial \mathcal{J}}{\partial \mathbf{c}} = -2\Phi^T(\mathbf{y} - \Phi\mathbf{c}) + 2\lambda\,\Lambda^{-1}\mathbf{c} = 0 \quad \Rightarrow \quad (\lambda\Lambda^{-1} + \Phi^T\Phi)\mathbf{c} = \Phi^T\mathbf{y} \tag{8.6.12}$$

$$\mathbf{c} = (\lambda\Lambda^{-1} + \Phi^T\Phi)^{-1}\Phi^T\mathbf{y} \tag{8.6.13}$$

Using the matrix-inversion lemma, we have:

$$(\lambda\Lambda^{-1} + \Phi^T\Phi)^{-1} = \frac{1}{\lambda}\big[\Lambda - \Lambda\Phi^T(\lambda I + \Phi\Lambda\Phi^T)^{-1}\Phi\Lambda\big] \tag{8.6.14}$$

from which it follows that:

$$(\lambda\Lambda^{-1} + \Phi^T\Phi)^{-1}\Phi^T = \Lambda\Phi^T(\lambda I + \Phi\Lambda\Phi^T)^{-1} \tag{8.6.15}$$

where I is the $N\times N$ identity matrix. Thus, the optimal coefficients are given by

$$\mathbf{c} = \Lambda\Phi^T(\lambda I + \Phi\Lambda\Phi^T)^{-1}\mathbf{y} \tag{8.6.16}$$

The observation vector $\mathbf{f} = \Phi\mathbf{c}$ and estimated function value $f(\mathbf{x}) = \boldsymbol{\phi}^T(\mathbf{x})\mathbf{c}$ are then,

$$\boxed{\begin{aligned}&\mathbf{f} = \Phi\Lambda\Phi^T(\lambda I + \Phi\Lambda\Phi^T)^{-1}\mathbf{y}\\ &f(\mathbf{x}) = \boldsymbol{\phi}^T(\mathbf{x})\Lambda\Phi^T(\lambda I + \Phi\Lambda\Phi^T)^{-1}\mathbf{y}\end{aligned}} \tag{8.6.17}$$

The appearance of the bilinear products of the basis functions suggests that we define the *kernel* function:

$$\boxed{K(\mathbf{x},\mathbf{x}') = \boldsymbol{\phi}^T(\mathbf{x})\Lambda\boldsymbol{\phi}(\mathbf{x}') = \sum_{i=1}^{M}\lambda_i\phi_i(\mathbf{x})\phi_i(\mathbf{x}')} \tag{8.6.18}$$

Let us also define the $N\times N$ symmetric positive-definite kernel matrix K and N-dimensional coefficient vector $\mathbf{a} = [a_0, a_1, \dots, a_{N-1}]^T$ by

$$\boxed{\begin{aligned}&K = \Phi\Lambda\Phi^T\\ &\mathbf{a} = (\lambda I + K)^{-1}\mathbf{y}\end{aligned}} \tag{8.6.19}$$

so that $\mathbf{c} = \Lambda\Phi^T\mathbf{a}$ and $\mathbf{f} = \Phi\mathbf{c} = \Phi\Lambda\Phi^T\mathbf{a} = K\mathbf{a}$ and $f(\mathbf{x}) = \boldsymbol{\phi}^T(\mathbf{x})\Lambda\Phi^T\mathbf{a}$. The matrix elements of K can be expressed in terms of the kernel function:

$$K_{nm} = (\Phi\Lambda\Phi^T)_{nm} = \boldsymbol{\phi}^T(\mathbf{x}_n)\Lambda\boldsymbol{\phi}(\mathbf{x}_m) = K(\mathbf{x}_n, \mathbf{x}_m) \tag{8.6.20}$$

for $n, m = 0, 1, \dots, N-1$. Similarly, we have for $f(\mathbf{x})$,

$$\begin{aligned}f(\mathbf{x}) &= \boldsymbol{\phi}^T(\mathbf{x})\Lambda\Phi^T\mathbf{a} = \boldsymbol{\phi}^T(\mathbf{x})\Lambda[\boldsymbol{\phi}(x_0), \dots, \boldsymbol{\phi}(\mathbf{x}_n), \dots, \boldsymbol{\phi}(x_{N-1})]\mathbf{a}\\ &= [K(\mathbf{x},\mathbf{x}_1), \dots, K(\mathbf{x},\mathbf{x}_n), \dots, K(\mathbf{x},\mathbf{x}_{N-1})]\mathbf{a} = \sum_{n=0}^{N-1}K(\mathbf{x},\mathbf{x}_n)a_n\end{aligned} \tag{8.6.21}$$

Thus, we may express (8.6.17) directly in terms of the kernel function and the coefficient vector $\mathbf{a}$,

$$\boxed{\mathbf{f} = K\mathbf{a}, \quad f(\mathbf{x}) = \sum_{n=0}^{N-1}K(\mathbf{x},\mathbf{x}_n)a_n} \tag{8.6.22}$$

Moreover, since $\mathbf{c} = \Lambda\Phi^T\mathbf{a}$, the norm $\|f\|^2$ can also be expressed in terms of K and the vector $\mathbf{a}$ as follows, $\|f\|^2 = \mathbf{c}^T\Lambda^{-1}\mathbf{c} = (\mathbf{a}^T\Phi\Lambda)\Lambda^{-1}(\Lambda\Phi^T\mathbf{a}) = \mathbf{a}^T(\Phi\Lambda\Phi^T)\mathbf{a}$, or,

$$\|f\|^2 = \mathbf{a}^T K\mathbf{a} \tag{8.6.23}$$

Thus, the knowledge of the kernel function $K(\mathbf{x},\mathbf{x}')$—rather than the knowledge of the possibly infinite set of basis functions $\phi_i(\mathbf{x})$—is sufficient to formulate and solve the regularization problem. Indeed, an equivalent optimization problem to (8.6.11) is the following, with the performance index to be minimized with respect to $\mathbf{a}$:

$$\boxed{\mathcal{J} = (\mathbf{y} - K\mathbf{a})^T(\mathbf{y} - K\mathbf{a}) + \lambda\mathbf{a}^T K\mathbf{a} = \min} \tag{8.6.24}$$

The vanishing of gradient with respect to $\mathbf{a}$ leads to the same solution as (8.6.19),

$$\frac{\partial\mathcal{J}}{\partial\mathbf{a}} = -2K^T(\mathbf{y} - K\mathbf{a}) + 2\lambda K\mathbf{a} = 0 \quad\Rightarrow\quad (\lambda K + K^T K)\mathbf{a} = K^T\mathbf{y} \quad\Rightarrow\quad \mathbf{a} = (\lambda I + K)^{-1}\mathbf{y}$$

where we used the symmetry property $K^T = K$ and assumed that K was invertible.

The linear vector space of functions of the form $f(\mathbf{x}) = \boldsymbol{\phi}^T(\mathbf{x})\mathbf{c}$, spanned by the set of basis functions $\{\phi_i(\mathbf{x}),\ i = 1, 2, \dots, M\}$, can be turned into an inner-product space (a Hilbert space if $M = \infty$) by endowing it with the inner product induced by the norm (8.6.9). That is, for any two functions $f_1(\mathbf{x}) = \boldsymbol{\phi}^T(\mathbf{x})\mathbf{c}_1$ and $f_2(\mathbf{x}) = \boldsymbol{\phi}^T(\mathbf{x})\mathbf{c}_2$, we define the inner product:

$$\langle f_1, f_2\rangle = \mathbf{c}_1^T\Lambda^{-1}\mathbf{c}_2 \tag{8.6.25}$$

The resulting vector space, say $\mathcal{H}$, is referred to as a *reproducing kernel Hilbert space*. By writing the kernel function in the form, $K(\mathbf{x},\mathbf{x}') = \boldsymbol{\phi}^T(\mathbf{x})\Lambda\boldsymbol{\phi}(\mathbf{x}') \equiv \boldsymbol{\phi}^T(\mathbf{x})\mathbf{c}(\mathbf{x}')$, with $\mathbf{c}(\mathbf{x}') = \Lambda\boldsymbol{\phi}(\mathbf{x}')$, we see that, as a function of $\mathbf{x}$ for each fixed $\mathbf{x}'$, it lies in the space $\mathcal{H}$, and satisfies the two reproducing-kernel properties:

$$f(\mathbf{x}') = \langle f(\cdot), K(\cdot,\mathbf{x}')\rangle, \quad K(\mathbf{x},\mathbf{x}') = \langle K(\cdot,\mathbf{x}), K(\cdot,\mathbf{x}')\rangle \tag{8.6.26}$$

These follow from the definition (8.6.25). Indeed, given $f(\mathbf{x}) = \boldsymbol{\phi}^T(\mathbf{x})\mathbf{c}$, we have,

$$\langle f(\cdot), K(\cdot,\mathbf{x}')\rangle = \mathbf{c}^T\Lambda^{-1}\mathbf{c}(\mathbf{x}') = \mathbf{c}^T\Lambda^{-1}\Lambda\,\boldsymbol{\phi}(\mathbf{x}') = \mathbf{c}^T\boldsymbol{\phi}(\mathbf{x}') = f(\mathbf{x}')$$

$$\langle K(\cdot,\mathbf{x}), K(\cdot,\mathbf{x}')\rangle = \mathbf{c}(\mathbf{x})^T\Lambda^{-1}\mathbf{c}(\mathbf{x}') = \boldsymbol{\phi}^T(\mathbf{x})\Lambda\Lambda^{-1}\Lambda\,\boldsymbol{\phi}(\mathbf{x}') = \boldsymbol{\phi}^T(\mathbf{x})\Lambda\,\boldsymbol{\phi}(\mathbf{x}') = K(\mathbf{x},\mathbf{x}')$$

One can re-normalize the basis functions by defining $\bar{\phi}_i(\mathbf{x}) = \lambda_i^{-1/2}\phi_i(\mathbf{x})$, or, vectorially $\bar{\boldsymbol{\phi}}(\mathbf{x}) = \Lambda^{-1/2}\boldsymbol{\phi}(\mathbf{x})$, which imply the renormalized basis matrix $\bar{\Phi} = \Phi\Lambda^{1/2}$ and coefficient vector $\bar{\mathbf{c}} = \Lambda^{-1/2}\mathbf{c}$. We obtain then the alternative expressions:

$$\begin{aligned}&\bar{\mathbf{c}} = \bar{\Phi}^T\mathbf{a}\\ &\mathbf{f} = \Phi\mathbf{c} = \bar{\Phi}\bar{\mathbf{c}}\\ &K = \Phi\Lambda\Phi^T = \bar{\Phi}\bar{\Phi}^T\\ &\|f\|^2 = \mathbf{c}^T\Lambda^{-1}\mathbf{c} = \bar{\mathbf{c}}^T\bar{\mathbf{c}}\end{aligned} \tag{8.6.27}$$

and kernel function,

$$K(\mathbf{x},\mathbf{x}') = \boldsymbol{\phi}^T(\mathbf{x})\Lambda\boldsymbol{\phi}(\mathbf{x}') = \bar{\boldsymbol{\phi}}^T(\mathbf{x})\bar{\boldsymbol{\phi}}(\mathbf{x}') \tag{8.6.28}$$

Eq. (8.6.28) expresses the kernel function as the dot product of two vectors and is known as the *kernel trick*. Given a kernel function $K(\mathbf{x},\mathbf{x}')$ that satisfies certain positive-definiteness conditions, the existence of basis functions satisfying Eq. (8.6.28) is guaranteed by Mercer's theorem [512]. The remarkable property of the kernel regularization approach is Eq. (8.6.22), which is known as the *representer theorem* [512],

$$f(\mathbf{x}) = \sum_{n=0}^{N-1}K(\mathbf{x},\mathbf{x}_n)a_n \tag{8.6.29}$$

It states that even though the original least-squares problem (8.6.11) was formulated in a possibly infinite-dimensional Hilbert space, the resulting solution is represented by a finite number of terms in Eq. (8.6.29). This property is more general than the above case and it applies to a performance index of the form:

$$\mathcal{J} = L(\mathbf{y} - \Phi\mathbf{c}) + \lambda\mathbf{c}^T\Lambda^{-1}\mathbf{c} = \min \tag{8.6.30}$$

where $L(\mathbf{z})$ is an arbitrary (convex, increasing, and differentiable) scalar function that replaces the quadratic norm $L(\mathbf{z}) = \mathbf{z}^T\mathbf{z}$. Indeed, the vanishing of the gradient gives,

$$\frac{\partial\mathcal{J}}{\partial\mathbf{c}} = -\Phi^T\frac{\partial L}{\partial\mathbf{y}} + 2\lambda\Lambda^{-1}\mathbf{c} = 0 \quad\Rightarrow\quad \mathbf{c} = \Lambda\Phi^T\frac{1}{2\lambda}\frac{\partial L}{\partial\mathbf{y}}$$

which implies for $\mathbf{f} = \Phi\mathbf{c}$ and $f(\mathbf{x}) = \boldsymbol{\phi}^T(\mathbf{x})\mathbf{c}$,

$$\mathbf{f} = K\mathbf{a}\,,\quad f(\mathbf{x}) = \sum_{n=0}^{N-1} K(\mathbf{x},\mathbf{x}_n)\,a_n\,,\quad \text{with}\quad \mathbf{a} = \frac{1}{2\lambda}\frac{\partial L(\mathbf{y}-K\mathbf{a})}{\partial\mathbf{y}} \tag{8.6.31}$$

where the last equation is a nonlinear equation for the N-vector $\mathbf{a}$. Of course, in the quadratic-norm case, $L(\mathbf{z}) = \mathbf{z}^T\mathbf{z}$, we obtain the equivalent of (8.6.19),

$$\mathbf{a} = \frac{1}{\lambda}(\mathbf{y} - K\mathbf{a})$$

Kernels and the above representation property are used widely in machine learning applications, such as support vector machines [499]. Some typical kernels that satisfy the representation property (8.6.28) are polynomial and gaussian of the type:

$$\begin{aligned} K(\mathbf{x},\mathbf{x}') &= (c + \mathbf{x}\cdot\mathbf{x}')^p \\ K(\mathbf{x},\mathbf{x}') &= \exp\Big(-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{2\sigma^2}\Big) \end{aligned} \tag{8.6.32}$$

By mapping nonlinear problems into linear ones, kernel methods offer a new paradigm for solving many of the classical problems of estimation and classification, including the "kernelization" of methods such as principal component analysis [513], canonical correlation analysis, array processing [516], and adaptive filtering [519]. Some accessible overviews of kernel methods with emphasis on regularization are [504,510,515]. For more details, the reader may consult the references [480–519].

8.7 Sparse Whittaker-Henderson Methods

Several variations of the Whittaker-Henderson method have been proposed in the literature that use different norms for the two terms of Eq. (8.1.1), such as the following criterion based on the ℓ_q and the ℓ_p norms, and using unity weights w_n for simplicity,

$$\mathcal{J}_{qp} = \sum_{n=0}^{N-1}|y_n - x_n|^q + \lambda\sum_{n=s}^{N-1}|\nabla^s x_n|^p = \min \tag{8.7.1}$$

Such criteria are capable of handling outliers in the data more effectively. Eq. (8.7.1) can be written vectorially with the help of the s-differencing matrix D of Eq. (8.1.8),

$$\mathcal{J}_{qp} = \|\mathbf{y}-\mathbf{x}\|_q^q + \lambda\|D\mathbf{x}\|_p^p = \min \tag{8.7.2}$$

where $\|\mathbf{x}\|_p$ denotes the ℓ_p norm of the vector $\mathbf{x} = [x_0, x_1, \cdots, x_{N-1}]^T$,

$$\|\mathbf{x}\|_p = \Big[\sum_{n=0}^{N-1}|x_n|^p\Big]^{\frac{1}{p}} \quad\Rightarrow\quad \|\mathbf{x}\|_p^p = \sum_{n=0}^{N-1}|x_n|^p$$

For $p = \infty$, we have instead,

$$\|\mathbf{x}\|_\infty = \max_{0\le n\le N-1}|x_n|$$

For $p = 0$, we define $\|\mathbf{x}\|_0$ as the *cardinality* of the vector $\mathbf{x}$, that is, the number of *nonzero* elements of $\mathbf{x}$. We note that $\|\mathbf{x}\|_p$ is a proper norm only for $p \ge 1$, however, the cases $0 \le p < 1$ have also been considered.

The case $\mathcal{J}_{11}$ was studied in [422,426] and formulated as a linear programming problem, the case $\mathcal{J}_{pp}$, including the ℓ_∞ norm case, $p = \infty$, was studied in [424], and the more general case, $\mathcal{J}_{qp}$, in [425]. More recently, the case $\mathcal{J}_{21}$, called ℓ_1 *trend filtering*, has been considered in [468] and has received a lot of attention [469–478].

Generally, the cases $\mathcal{J}_{2p}$ are examples of so-called ℓ_p-regularized least-squares problems, which have been studied very extensively in inverse problems, with renewed interest in sparse modeling, statistical learning, compressive sensing applications—a small and very incomplete set of references on regularization and sparse regularization methods is [479–590]. We discuss such regularization issues in more detail in Chap. 15. Next, we concentrate on the original $\mathcal{J}_{22}$ criterion, and the $\mathcal{J}_{21}$ and $\mathcal{J}_{20}$ criteria,

$$\begin{aligned} \mathcal{J}_{22} &= \|\mathbf{y}-\mathbf{x}\|_2^2 + \lambda\|D\mathbf{x}\|_2^2 = \min \\ \mathcal{J}_{21} &= \|\mathbf{y}-\mathbf{x}\|_2^2 + \lambda\|D\mathbf{x}\|_1 = \min \\ \mathcal{J}_{20} &= \|\mathbf{y}-\mathbf{x}\|_2^2 + \lambda\|D\mathbf{x}\|_0 = \min \end{aligned} \tag{8.7.3}$$

The $\mathcal{J}_{21}$ and $\mathcal{J}_{20}$ criteria tend to promote the *sparsity* of the regularizing term $D\mathbf{x}$, that is, $D\mathbf{x}$ will be a *sparse* vector consisting mostly of zeros with a few nonzero entries. Since $D\mathbf{x}$ represents the s-differenced signal, $\nabla^s x_n$, its piecewise vanishing implies that the trend x_n will be a piecewise polynomial of order $s-1$, with the polynomial pieces joining continuously at few break (or, kink) points where $\nabla^s x_n$ is nonzero.

This is similar to the spline smoothing case, except here the locations of the break points are determined dynamically by the solution of the optimization problem, whereas in the spline case they are at prescribed locations.

For differencing order $s = 2$, used in Hodrick-Prescott and ℓ_1-trend-filtering cases, the trend signal x_n will be a piecewise linear function of n, with a sparse number of slope changes. The case $s = 3$, used originally by Whittaker and Henderson, would correspond to piecewise parabolic segments in n. The case $s = 1$, corresponding to the original Bohlmann choice, results in a piecewise constant trend signal x_n. This case is known also as *total variation* minimization method and has been applied widely in image processing.

The $\mathcal{J}_{21}$ problem can be implemented easily in MATLAB with the CVX package.[†] The $\mathcal{J}_{20}$ problem, which produces the sparsest solution, can be solved by an *iterative reweighted ℓ_1-regularized* method [468], or alternatively, by an *iterative reweighted least-squares* method, and can also be used to solve the $\mathcal{J}_{21}$ and the $\mathcal{J}_{2p}$ problems.

There are several variants of the iterative reweighted least-squares (IRLS) method, [520–528,532,553,557,560,565,566], but the basic idea is to replace the ℓ_p norm with a weighted ℓ_2 norm, which can be solved iteratively. Given any real number $0 \le p \le 2$, let $q = 2 - p$, and note that for any real number $x \neq 0$, we can write,

$$|x|^p = \frac{|x|^2}{|x|^q} \approx \frac{|x|^2}{|x|^q + \varepsilon}$$

where ε is a sufficiently small positive number needed to also allow the case $x = 0$. Similarly, we can write for the ℓ_p-norm of a vector $\mathbf{x} \in \mathbb{R}^N$,

$$\begin{aligned} \|\mathbf{x}\|_p^p &= \sum_{i=0}^{N-1} |x_i|^p \approx \sum_{i=0}^{N-1} \frac{|x_i|^2}{|x_i|^q + \varepsilon} = \mathbf{x}^T W(\mathbf{x})\mathbf{x} \\ W(\mathbf{x}) &= \text{diag}\left[\frac{1}{|\mathbf{x}|^q + \varepsilon}\right] = \text{diag}\left[\frac{1}{|x_0|^q + \varepsilon}, \frac{1}{|x_1|^q + \varepsilon}, \dots, \frac{1}{|x_{N-1}|^q + \varepsilon}\right] \end{aligned} \tag{8.7.4}$$

Then, the ℓ_p-regularized problem $\mathcal{J}_{2p}$ can be written in the form,

$$\mathcal{J} = \|\mathbf{y} - \mathbf{x}\|_2^2 + \lambda\|D\mathbf{x}\|_p^p = \|\mathbf{y} - \mathbf{x}\|_2^2 + \lambda\,\mathbf{x}^T D^T W(D\mathbf{x})\,D\mathbf{x} = \min \tag{8.7.5}$$

which leads to to the following iterative algorithm,

```
for $k = 1, 2, \dots, K$, do:
    $W_{k-1} = W(D\mathbf{x}^{(k-1)})$
    $\mathbf{x}^{(k)} = \arg\min_{\mathbf{x}} \|\mathbf{y} - \mathbf{x}\|_2^2 + \lambda\,\mathbf{x}^T D^T W_{k-1} D\mathbf{x}$
```

(IRLS) (8.7.6)

with the algorithm initialized to the ordinary least-squares solution of criterion $\mathcal{J}_{22}$,

$$\mathbf{x}^{(0)} = (I + \lambda D^T D)^{-1}\mathbf{y}$$

The solution of the optimization problem in (8.7.6) at the kth step is:

$$\mathbf{x}^{(k)} = (I + \lambda D^T W_{k-1} D)^{-1}\mathbf{y}$$

Thus, the choices $p = 0$ and $p = 1$ should resemble the solutions of the ℓ_0 and ℓ_1 regularized problems.

Example 8.7.1: *Global Warming Trends.* This is a continuation of Example 3.9.2 in which we compared several smoothing methods. Fig. 8.7.1 compares the Whittaker-Henderson trends for the ℓ_2, ℓ_1, and ℓ_0 cases, with $s = 2$, as well as the corresponding regularizing differenced signals, $\nabla^s x_n$.

[†] http://cvxr.com/cvx/

The ℓ_1 case was computed with the CVX package. The corresponding IRLS implementation is not shown since it produces virtually indistinguishable graphs from CVX.

The ℓ_0 case was implemented with the IRLS method and produced slightly sparser differenced signals as can be observed in the graphs. The MATLAB code used to generate these graphs is summarized below.

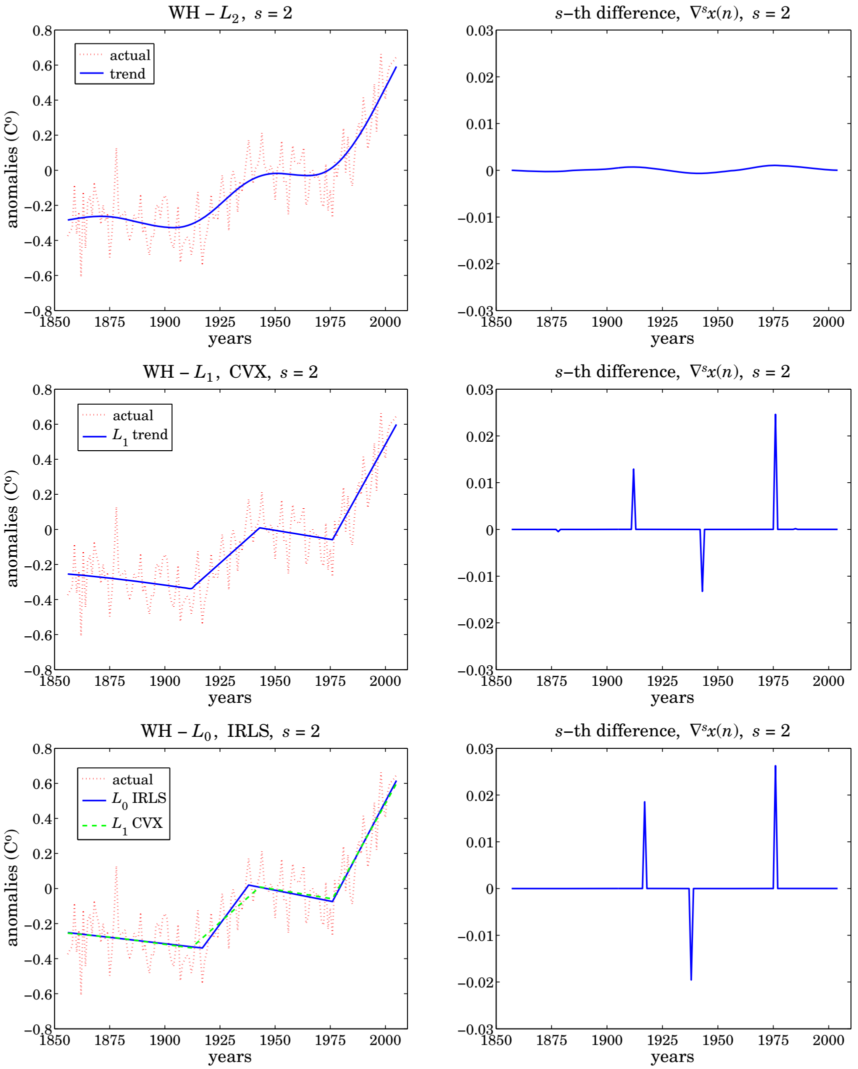


Fig. 8.7.1 Comparison of ℓ_2, ℓ_1, and ℓ_0 trends, and differenced signals for $s = 2$.

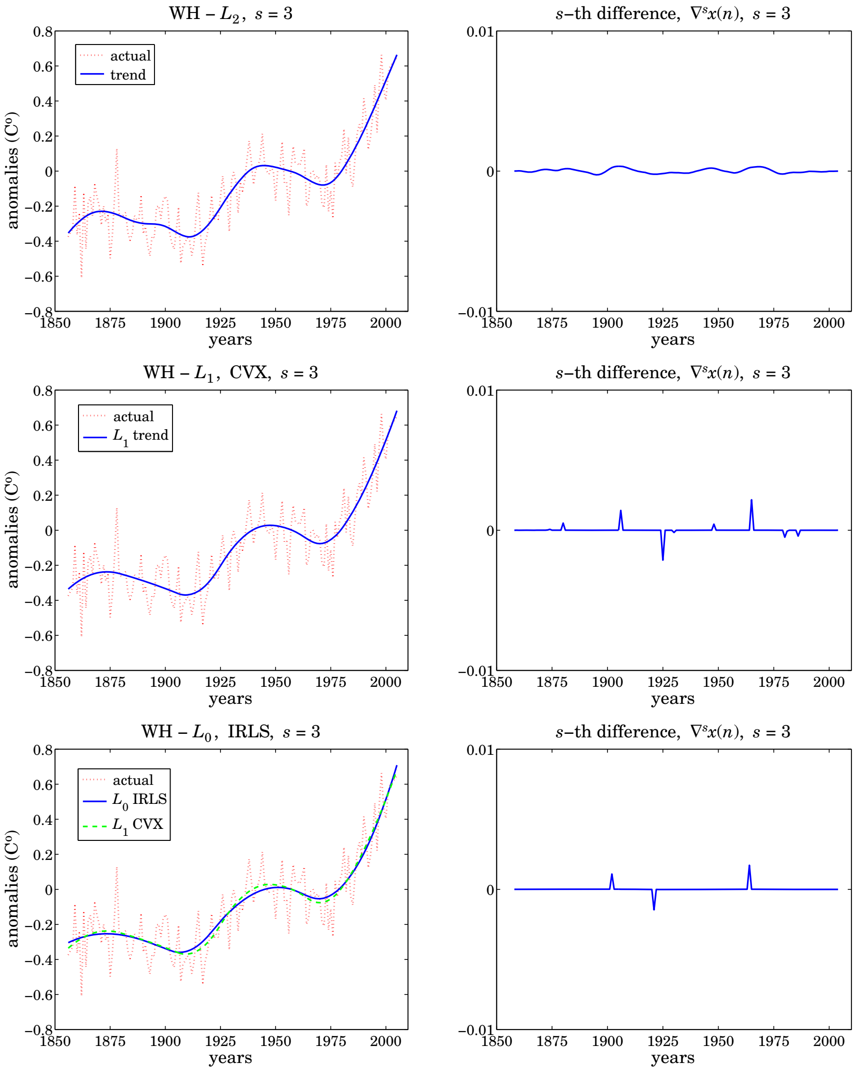


Fig. 8.7.2 Comparison of ℓ_2, ℓ_1, and ℓ_0 trends, and differenced signals for $s = 3$.

```
Y = loadfile(’tavenh2v.dat’);             % load temperature data file

n = Y(:,1); y = Y(:,14); N = length(y);   % extract dates and data

s = 2; Ds = diff(speye(N),s);             % (N-s)xN differencing matrix
ns = n(s:end-1);
```

```
la = 10000; x = whsm(y,la,s);             % Whittaker-Henderson with L2 norm

figure; plot(n,y,’r:’, n,x,’b-’);         % plot trend
figure; plot(ns, Ds*x,’b-’);              % plot differenced trend

la = 10;                                  % Whittaker-Henderson with L1 norm
cvx_quiet(true);                          % CVX package, http:/cvxr.com/cvx/
cvx_begin
    variable x(N)
    minimize( sum_square(y-x) + la * norm(Ds*x,1) )
cvx_end

figure; plot(n,y,’r:’, n,x,’b-’);         % plot trend
figure; plot(ns, Ds*x,’b-’);              % plot differenced trend

p = 0; q = 2 - p; epsilon = 1e-8;         % Whittaker-Henderson with L0 norm
I = speye(N); K = 10;                     % using K=10 IRLS iterations
la = 0.05;

x  = (I + la*Ds’*Ds) \ y;                 % initialize iteration

for k=1:K,                                % IRLS iteration
   W = diag(1./(abs(Ds*x).^q + epsilon));
   xk = (I + la*Ds’*W*Ds) \ y;
   x = xk;
end

figure; plot(n,y,’r:’, n,x,’b-’);         % plot trend
figure; plot(ns, Ds*x,’b-’);              % plot differenced trend
```

Fig. 8.7.2 compare the ℓ_2, ℓ_1, ℓ_0 cases for $s = 3$, which fits piecewise quadratic polynomials to the data. The ℓ_0 case is again the sparsest. (Color graphs online). □

8.8 Computer Project – US GDP Macroeconomic Data

In this project you will study the *Whittaker-Henderson smoothing method* formulated with the L_2 and L_1 norms, and apply it to the US GDP macroeconomic data. For a length-N signal, y_n, $0 \le n \le N-1$, the optimization criteria for determining a length-N smoothed signal x_n are,

$$
\begin{aligned}
(L_2):\quad & \mathcal{J} = \sum_{n=0}^{N-1} \left|y_n - x_n\right|^2 + \lambda \sum_{n=s}^{N-1} \left|\nabla^s x_n\right|^2 = \|\mathbf{y}-\mathbf{x}\|_2^2 + \lambda\,\|D_s\mathbf{x}\|_2^2 = \min \\
(L_1):\quad & \mathcal{J} = \sum_{n=0}^{N-1} \left|y_n - x_n\right|^2 + \lambda \sum_{n=s}^{N-1} \left|\nabla^s x_n\right| \;\; = \|\mathbf{y}-\mathbf{x}\|_2^2 + \lambda\,\|D_s\mathbf{x}\|_1 = \min
\end{aligned}
\tag{8.8.1}
$$

where D_s is the $(N-s)\times N$ convolution matrix corresponding the s-difference operator ∇^s. It can be constructed in MATLAB by,

```
Ds = diff(eye(N),s);       % or, in sparse form, Ds = diff(speye(N),s);
```

The solution of problem (L_2) is straightforward:

$$
\mathbf{x} = (I + \lambda D_s^T D_s)^{-1}\mathbf{y} \tag{8.8.2}
$$

The solution of problem (L_1) can be obtained with the CVX package as follows:

```
cvx_begin
   variable x(N)
   minimize( sum_square(x-y) + lambda * norm(Ds*x,1) );
cvx_end
```

It can also be solved with the *iterative reweighted least-squares* (IRLS) algorithm, as discussed in Sec. 8.7.

The second column of the OSP data file, `USGDP_Inv.dat`, represents the quarterly US GDP for private investment in billions of dollars. Read this column with the help of the function `loadfile` and then take its log:

```
Y = loadfile('USGDP_Inv.dat');
y = log10(Y(:,2) * 1e9);
```

These data represent a prototypical example for the application of Whittaker-Henderson filters, referred to in this context as Hodrick-Prescott filters.

a. Choose difference order $s = 2$ and regularization parameter $\lambda = 1600$. Solve the Whittaker-Henderson problem (L_2) and plot the solution $\mathbf{x}$ together with the actual data $\mathbf{y}$.

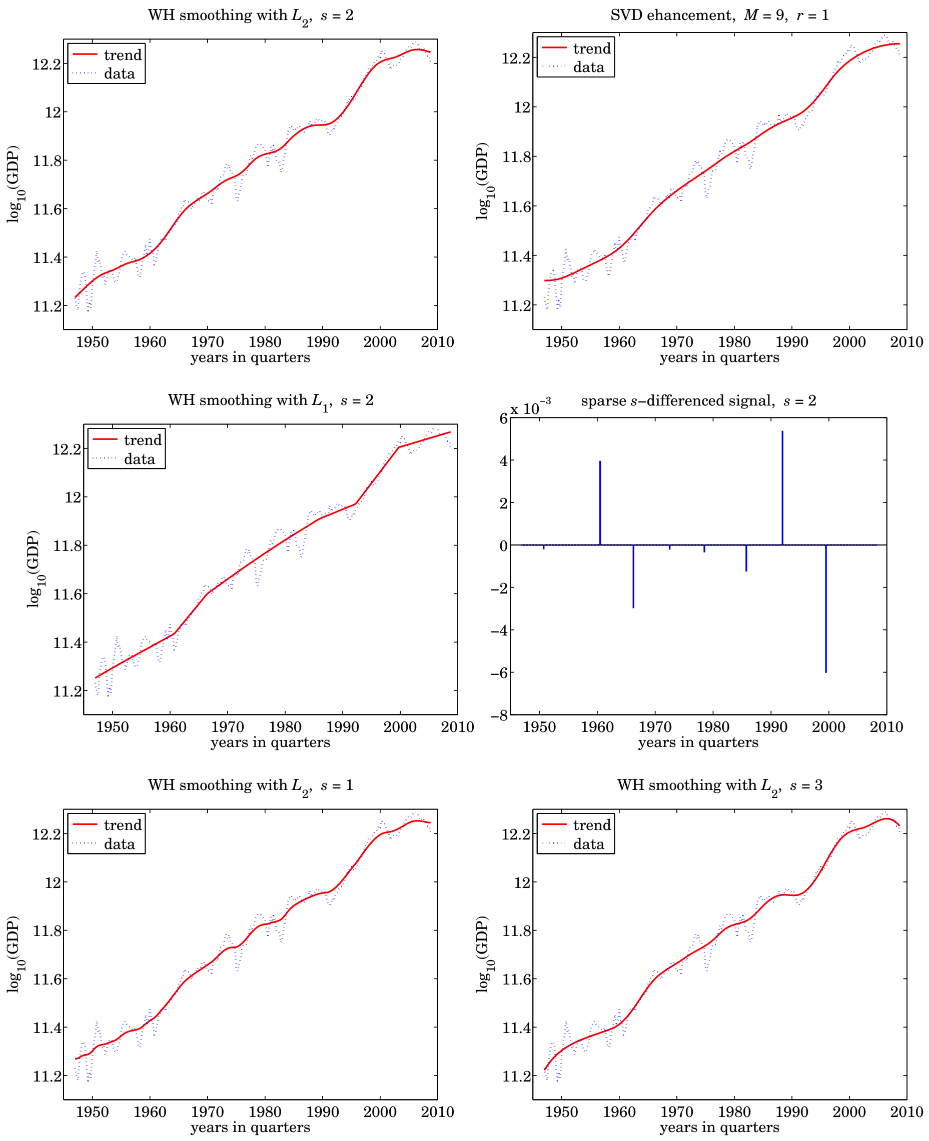
b. Calculate the SVD enhanced version of $\mathbf{y}$, by the following steps: (i) remove and save the mean from $\mathbf{y}$, (ii) form its forward/backward data matrix using an embedding order of $M = 9$, (iii) subject it to $K = 8$ SVD enhancement iterations using rank $r = 1$, (iv) extract the enhanced signal from the enhanced data matrix, and (v) add the mean that was removed. Plot the resulting enhanced signal together with $\mathbf{y}$. The MATLAB steps are summarized in Sec. 15.17 (use, `type=2`, for the F/B case).

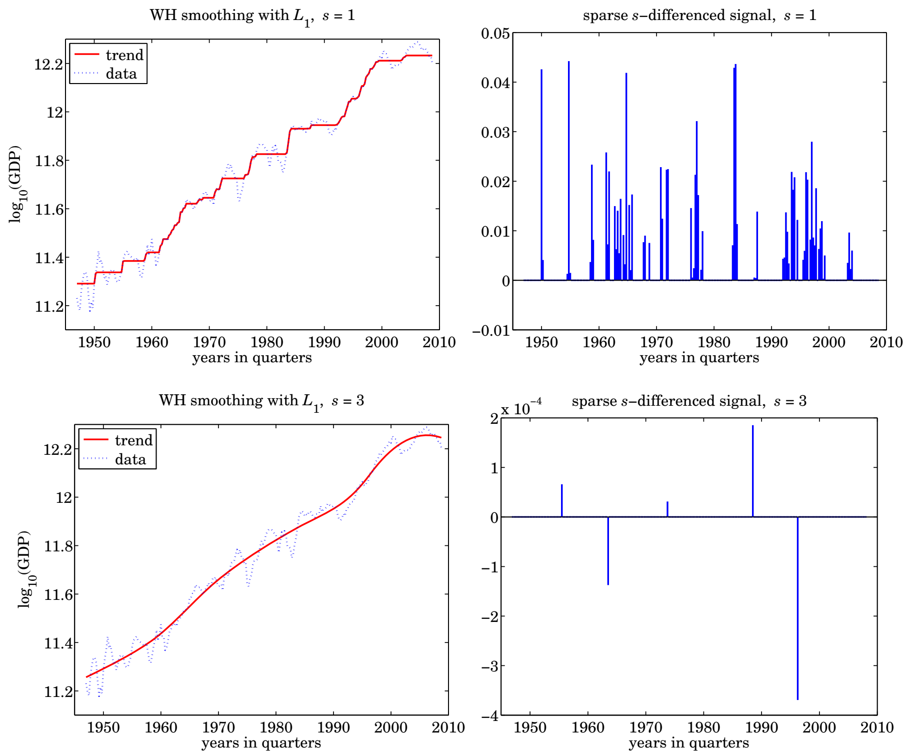
c. For the value $\lambda_1 = \lambda/480 = 1600/480$ and difference order $s = 2$, solve problem (L_1), and plot the solution $\mathbf{x}$ together with the actual data $\mathbf{y}$. Moreover, on a separate graph, plot the differenced signal $D_s\mathbf{x}$ using a stem plot and observe its sparseness, which means that $\mathbf{x}$ is piece-wise linear. The particular choice for λ_1 was made in order for the (L_2) and (L_1) problems to have comparable RMS errors.

d. Repeat parts (a) and (c) for $s = 1$ and regularization parameter $\lambda = 60.65$ for the (L_2) problem (justified in Sec. 8.3), and $\lambda_1 = 1$ for the (L_1) problem, chosen to achieve comparable RMS errors. Notice how the (L_1) problem results in a piece-wise constant fit. But $D_s\mathbf{x}$ is not as sparse because $s = 1$ is not really a good choice.

   The $s = 1$ case is an example of the so-called *total-variation minimization* method, used widely in image processing.

e. Repeat parts (a) and (c) for $s = 3$ and regularization parameter $\lambda = 41640.16$ for the (L_2) problem (justified in Sec. 8.3), and $\lambda_1 = \lambda/1000$ for the (L_1) problem, chosen to achieve comparable RMS errors. Here, the (L_1) problem will result in piece-wise quadratic polynomial fits. Some example graphs are shown below.





8.9 Problems

8.1 For the case $s = 1$, show that the Whittaker-Henderson filter has poles z_1, $1/z_1$, where

$$z_1 = e^{-\alpha}, \quad \alpha = 2\,\mathrm{asinh}\left(\frac{1}{2\sqrt{\lambda}}\right)$$

For the case $s = 2$, show that the filter has poles $\{z_1,\, z_1^*,\, 1/z_1,\, 1/z_1^*\}$, where

$$z_1 = \left(\sqrt{1-ja^2} + jae^{j\pi/4}\right)^2 = \frac{1}{2}D^2\left(1-\frac{a}{D}\right)^2\left(1+j\frac{a}{D}\right)^2, \quad D = \sqrt{1+\sqrt{1+a^4}}, \quad a = \frac{1}{2\lambda^{1/4}}$$

Show that in both cases $|z_1| < 1$.

8.2 Determine explicit expressions in terms of λ for the quantities σ^2 and z_1 that appear in the factorization of the denominator of the Hodrick-Prescott filter:

$$1+\lambda(1-z^{-1})^2(1-z)^2 = \sigma^2(1-z_1z^{-1})(1-z_1^*z^{-1})(1-z_1z)(1-z_1^*z)$$

What are the numerical values of σ^2, z_1 for $\lambda = 1600$? What are the values of the coefficients of the second-order filter $(1-2\,\mathrm{Re}(z_1)z^{-1}+|z_1|^2z^{-2})$?

8.3 Consider the performance index (8.6.1) for a regularized deconvolution problem. Making the enough assumptions, show that the performance index can be written in terms of frequency responses as follows,

$$\begin{aligned}\mathcal{J} &= \sum_n |y_n - f_n * x_n|^2 + \lambda\sum_n |d_n * x_n|^2 \\ &= \int_{-\pi}^{\pi}\Big[\,|Y(\omega)-F(\omega)X(\omega)|^2 + \lambda\,|D(\omega)X(\omega)|^2\Big]\frac{d\omega}{2\pi} = \min\end{aligned}$$

Determine the optimum $X(\omega)$ that minimizes this index. Then, show that the corresponding optimum deconvolution filter $H(z) = X(z)/Y(z)$ is given by:

$$H(z) = \frac{F(z^{-1})}{F(z)F(z^{-1})+\lambda D(z)D(z^{-1})}$$

What would be the stochastic state-space model for x_n, y_n that has this $H(z)$ as its optimum double-sided (unrealizable) Wiener filter for estimating x_n from y_n?

9

Periodic Signal Extraction

Many physical, financial, and social time series have a natural periodicity in them, such as daily, monthly, quarterly, yearly. The observed signal can be regarded as having three components: a periodic (or nearly periodic) seasonal part s_n, a smooth trend t_n, and a residual irregular part v_n that typically represents noise,

$$y_n = s_n + t_n + v_n$$

The model can also be assumed to be multiplicative, $y_n = s_n t_n v_n$. The signal processing task is to extract both the trend and the seasonal components, t_n and s_n, from the observed signal y_n.

For example, many climatic signals, such as CO_2 emissions, are characterized by an annual periodicity. Government agencies routinely estimate and remove the seasonal component from business and financial data and only the "seasonally-adjusted" signal $a_n = t_n + v_n$ is available, such as the US GDP that we considered in Example 8.3.1. Further processing of the deseasonalized signal a_n, using for example a trend extraction filter such as the Hodrick-Prescott filter, can reveal additional information, such as business cycles.

Periodic signals appear also in many engineering applications. Some examples are: (a) Electrocardiogram recordings are subject to power frequency interference (e.g., 60 Hz and its higher harmonics) which must be removed by appropriate filters. (b) All biomedical signals require some sort of signal processing for their enhancement. Often weak biomedical signals, such as brain signals from visual responses or muscle signals, can be evoked periodically with the responses accumulated (averaged) to enhance their SNR; (c) TV video signals have two types of periodicities in them, one due to line-scanning and one due to the frame rate. In the pre-HDTV days, the chrominance (color) TV signals were put on a subcarrier signal and added to the luminance (black & white) signal, and the composite signal was then placed on another carrier for transmission. The subcarrier's frequency was chosen carefully so as to shift the line- and frame-harmonics of the chrominance signal relative to those of the luminance so that at the receiving end the two could be separated by appropriately designed comb filters [30]. (d) GPS signals contain a repetitive code word that repeats with a period of one millisecond. The use of comb filters can enhance their reception. (e) Radars send out repetitive pulses so that

the returns from slowly moving targets have a quasi-periodic character. By accumulating these return, the SNR can be enhanced. As we see below, signal averaging is a form of comb filtering.

In this chapter, we discuss the design of comb and notch filters for extracting periodic signals or canceling periodic interference. We discuss also the specialized comb filters, referred to as "seasonal filters," that are used by standard seasonal decomposition methods, such as the census X-11 method, and others.

9.1 Notch and Comb Filters for Periodic Signals

To get started, we begin with the signal plus interference model $y_n = s_n + v_n$ in which either the signal or the noise is periodic, but not both.

If the noise v_n is periodic, its spectrum will be concentrated at the harmonics of some fundamental frequency, say ω_1. The noise reduction filter must be an ideal *notch filter* with notches at the harmonics $k\omega_1$, $k = 0, 1, \dots$, as shown in Fig. 9.1.1. If the filter notches are narrow, then the distortion of the desired signal s_n will be minimized.

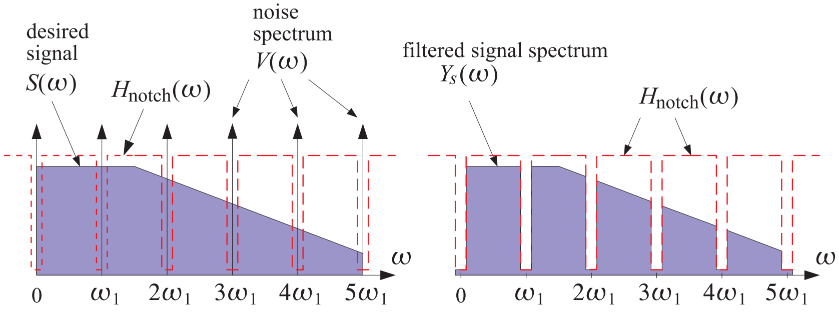


Fig. 9.1.1 Notch filter for reducing periodic interference.

On the other hand, if the desired signal s_n is periodic and the noise is a wideband signal, the signal enhancement filter for extracting s_n must be an ideal *comb filter* with peaks at the harmonics of the desired signal, as shown in Fig. 9.1.2. If the comb peaks are narrow, then only a minimal amount of noise will pass through the filter (that is, the portion of the noise whose power lies within the narrow peaks.)

A discrete-time periodic signal s_n with a period of D samples admits the following finite D-point DFT and inverse DFT representation [29] in terms of the D harmonics that lie within the Nyquist interval, $\omega_k = 2\pi k/D = k\omega_1$, for $k = 0, 1, \dots, D-1$,

$$\begin{aligned} &\text{(DFT)} \qquad S_k = \sum_{n=0}^{D-1} s_n e^{-j\omega_k n}\,, \quad k = 0, 1, \dots, D-1 \\ &\text{(IDFT)} \qquad s_n = \frac{1}{D}\sum_{k=0}^{D-1} S_k e^{j\omega_k n}\,, \quad n = 0, 1, \dots, D-1 \end{aligned} \tag{9.1.1}$$

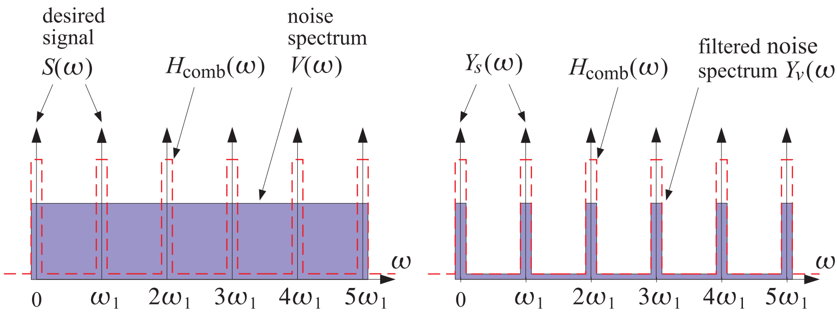


Fig. 9.1.2 Comb filter for enhancing a periodic signal.

where S_k is the D-point DFT of one period $[s_0, s_1, \dots, s_{D-1}]$ of the time signal. Because of the periodicity, the IDFT formula is actually valid for all n in the interval $-\infty < n < \infty$.

We note that a periodic continuous-time signal $s(t)$ does not necessarily result into a periodic discrete-time signal when sampled at some arbitrary rate. For the sampled signal $s_n = s(nT)$ to be periodic in n with a period of D samples, where T is the sampling interval, the sampling rate $f_s = 1/T$ must be D times the fundamental harmonic f_1, that is, $f_s = Df_1$, or equivalently, one period $T_{\rm per} = 1/f_1$ must contain D samples, $T_{\rm per} = DT$. This implies periodicity in n,

$$s_{n+D} = s\big((n+D)T\big) = s(nT + DT) = s(nT + T_{\rm per}) = s(nT) = s_n$$

The assumed periodicity of s_n implies that the sum of any D successive samples, $(s_n + s_{n-1} + \cdots + s_{n-D+1})$, is a constant independent of n. In fact, it is equal to the DFT component S_0 at DC $(\omega_k = 0)$,

$$s_n + s_{n-1} + \cdots + s_{n-D+1} = S_0\,, \quad -\infty < n < \infty \tag{9.1.2}$$

In a seasonal + trend model such as $y_n = s_n + t_n + v_n$, we may be inclined to associate any DC term with the trend t_n rather with the periodic signal s_n. Therefore, it is common to assume that the DC component of s_n is absent, that is, the sum (9.1.2) is zero, $S_0 = 0$. In such cases, the comb filter for extracting s_n must be designed to have peaks only at the non-zero harmonics, $\omega_k = k\omega_1$, $k = 1, 2, \dots, D-1$. Similarly, the notch filter for removing periodic noise must not have a notch at DC.

The typical technique for designing notch and comb filters for periodic signals is by frequency scaling, that is, the mapping of frequencies $\omega \to \omega D$, or equivalently, the mapping of the z-domain variable

$$z \to z^D \tag{9.1.3}$$

The effect of the transformation is to shrink the spectrum by a factor of D and then replicate it D times to fill the new Nyquist interval. An example is shown in Fig. 9.1.3 for $D = 4$. Starting with a lowpass filter $H_{\rm LP}(\omega)$, the frequency-scaled filter will be a comb filter, $H_{\rm comb}(\omega) = H_{\rm LP}(\omega D)$. Similarly, a highpass filter is transformed into a notch filter $H_{\rm notch}(\omega) = H_{\rm HP}(\omega D)$.

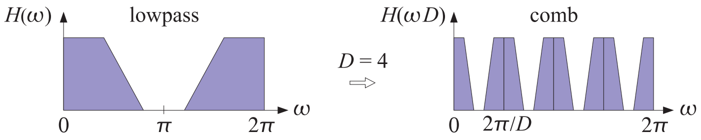


Fig. 9.1.3 Mapping of a lowpass filter to a comb filter by frequency scaling.

In the z-domain, we have the following simple prescriptions for turning lowpass and highpass filters into comb and notch filters:

$$\boxed{\begin{aligned} H_{\rm comb}(z) &= H_{\rm LP}(z^D) \\ H_{\rm notch}(z) &= H_{\rm HP}(z^D) \end{aligned}} \tag{9.1.4}$$

For example, the simplest comb and notch filters are generated by,

$$\begin{aligned} H_{\rm LP}(z) &= \frac{1}{2}(1 + z^{-1}) & & H_{\rm comb}(z) = \frac{1}{2}(1 + z^{-D}) \\ & & \Rightarrow & \\ H_{\rm HP}(z) &= \frac{1}{2}(1 - z^{-1}) & & H_{\rm notch}(z) = \frac{1}{2}(1 - z^{-D}) \end{aligned} \tag{9.1.5}$$

Their magnitude responses are shown in Fig. 9.1.4 for $D = 10$. The harmonics $\omega_k = 2\pi k/D = 2\pi k/10$, $k = 0, 1, \dots, 9$ are the peaks/notches of the comb/notch filters. The original lowpass and highpass filter responses are shown as the dashed lines. The factors $1/2$ in Eq. (9.1.5) normalize the peak gains to unity. The magnitude responses of the two filters are:

$$|H_{\rm comb}(\omega)|^2 = \cos^2(\omega D/2)\,, \quad |H_{\rm notch}(\omega)|^2 = \sin^2(\omega D/2) \tag{9.1.6}$$

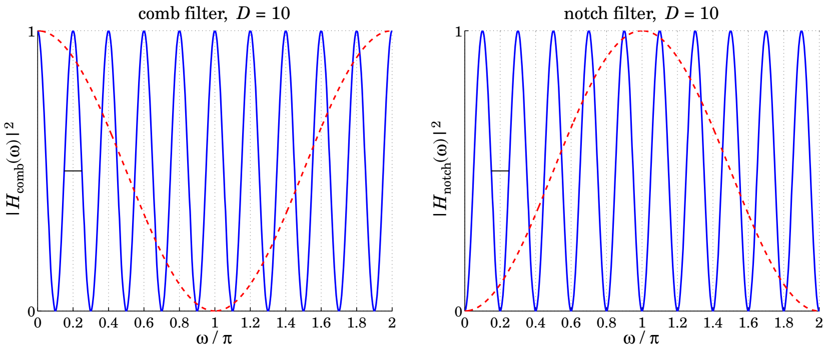


Fig. 9.1.4 Simple comb and notch filters with $D = 10$.

The filters are complementary, as well as power-complementary, in the sense,

$$H_{\rm comb}(z) + H_{\rm notch}(z) = 1\,, \quad |H_{\rm comb}(\omega)|^2 + |H_{\rm notch}(\omega)|^2 = 1 \tag{9.1.7}$$

The 3-dB widths $\Delta\omega$ of the comb peaks or the notch dips are fixed by the period D. Indeed, they are defined by the condition $\sin^2(D\Delta\omega/4) = 1/2$, which gives $\Delta\omega = \pi/D$. They are indicated on Fig. 9.1.4 as the short horizontal lines at the half-power level.

In order to control the width, we must consider IIR or higher order FIR filters. For example, we may start with the lowpass filter given in Eq. (2.3.5), and its highpass version,

$$H_{\text{LP}}(z) = b\,\frac{1+z^{-1}}{1-az^{-1}}\,,\quad b = \frac{1-a}{2}\,,\qquad H_{\text{HP}}(z) = b\,\frac{1-z^{-1}}{1-az^{-1}}\,,\quad b = \frac{1+a}{2} \tag{9.1.8}$$

where $0 < a < 1$. The transformation $z \to z^D$ gives the comb and notch filters [30]:

$$\begin{aligned} H_{\text{comb}}(z) &= b\,\frac{1+z^{-D}}{1-az^{-D}}\,,\quad b = \frac{1-a}{2} \\ H_{\text{notch}}(z) &= b\,\frac{1-z^{-D}}{1-az^{-D}}\,,\quad b = \frac{1+a}{2} \end{aligned} \tag{9.1.9}$$

The filters remain complementary, and power-complementary, with magnitude responses:

$$\begin{aligned} |H_{\text{comb}}(\omega)|^2 &= \frac{\beta^2}{\beta^2 + \tan^2(\omega D/2)}\,,\quad \beta \equiv \frac{1-a}{1+a} \\ |H_{\text{notch}}(\omega)|^2 &= \frac{\tan^2(\omega D/2)}{\beta^2 + \tan^2(\omega D/2)} \end{aligned} \tag{9.1.10}$$

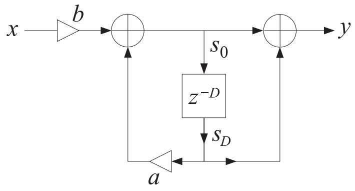
Their 3-dB width $\Delta\omega$ is controlled by the pole parameter a through the relation [30]:

$$\tan\left(\frac{D\Delta\omega}{4}\right) = \frac{1-a}{1+a} = \beta \tag{9.1.11}$$

The noise reduction ratio of the comb filter is the same as that of the lowpass filter $H_{\text{LP}}(z)$, which was calculated in Chap. 2,

$$\mathcal{R} = \frac{1-a}{2} = \frac{\beta}{1+\beta} \tag{9.1.12}$$

and can be made as small as desired by increasing a towards unity, but at the expense of also increasing the time constant of the filter. The canonical (direct-form II) realization of the comb filter and its sample processing algorithm using a circular buffer implementation of the multiple delay z^{-D} is as follows in the notation of [30], where p is the circular pointer,



```
for each $x$ do:
    $s_D = *(p+D)$
    $s_0 = bx + as_D$
    $y = s_0 + s_D$
    $*p = s_0$
    $p--$
```

Example 9.1.1: Fig. 9.1.5 shows two examples designed with $D = 10$ and 3-dB widths $\Delta\omega = 0.05\pi$ and $\Delta\omega = 0.01\pi$. By comparison, the simple designs had $\Delta\omega = \pi/D = 0.1\pi$. For $\Delta\omega = \pi/D$, we have $\tan(D\Delta\omega/4) = \tan(\pi/4) = 1$, which implies that $a = 0$ and $b = 1/2$, reducing to the simple designs of Eq. (9.1.5). □

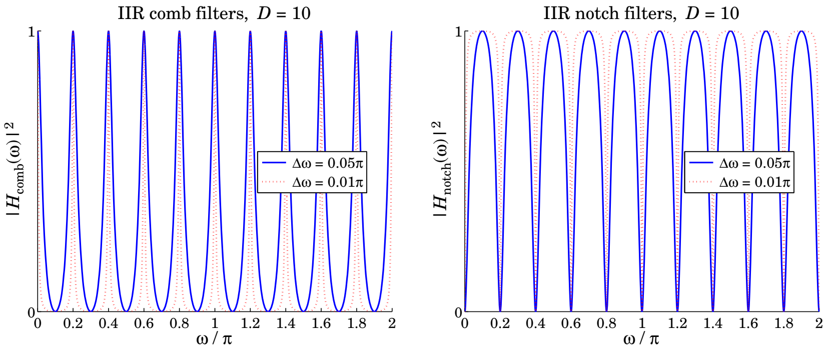


Fig. 9.1.5 Recursive comb and notch filters with $D = 10$.

Example 9.1.2: Fig. 9.1.6 shows on the left a simulated electrocardiogram (ECG) signal corrupted by 60 Hz power frequency interference and its harmonics. On the right, it shows the result of filtering by an IIR notch filter. The underlying ECG is recovered well after the initial transients die out.

The sampling rate was $f_s = 600$ Hz and the fundamental frequency of the noise, $f_1 = 60$ Hz. This gives for the period $D = f_s/f_1 = 10$. The ECG beat was taken to be 1 sec and therefore there were 600 samples in each beat for a total of 1200 samples in the two beats shown in the figure.

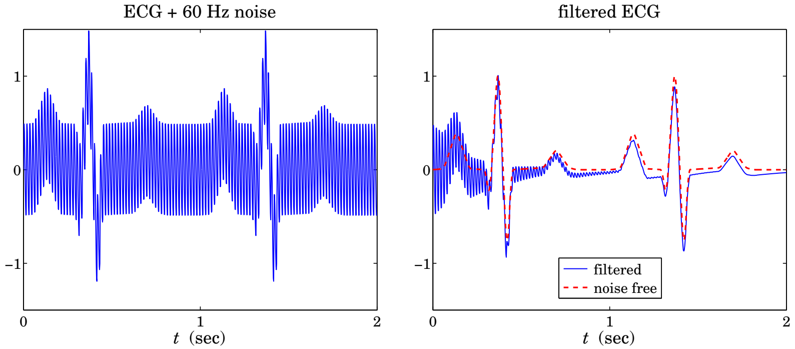


Fig. 9.1.6 Eliminating 60 Hz harmonics from ECG signal.

The IIR notch filter was designed to achieve a 3-dB width of $\Delta f = f_1/50$, that is, a Q-factor of $Q = f_1/\Delta f = 50$. Therefore, in units of rads/sample, the notch width is $\Delta\omega = 2\pi\Delta f/f_s = 2\pi/(DQ) = 0.004\pi$, which results in the filter parameters $a = 0.9391$ and $b = (1+a)/2 = 0.9695$. Thus, the designed notch filter was:

$$H_{\text{notch}}(z) = 0.9695\,\frac{1-z^{-10}}{1-0.9391z^{-10}}$$

The noise was simulated by adding the following harmonic components,

$$v_n = \sum_{k=1}^{D/2-1} A_k \sin(\omega_k n)\,, \quad \text{with } \omega_k = \frac{2\pi k}{D}\,, \quad A_k = \frac{1}{2k^2}$$

where the amplitudes A_k were arbitrarily chosen. Note that only the non-zero harmonics that lie in the interval $0 < \omega < \pi$ were used.

The 40-dB time-constant of the notch filter was $n_{\text{eff}} = D\,\ln(0.01)\,/\,\ln(a) = 732$ samples or equivalently, $\tau = n_{\text{eff}}/f_s = 732/600 = 1.22$ sec. It is evident from the figure that beyond this time, the transients essentially die out. The MATLAB code for generating these graphs was as follows:

```
nbeats = 2; L = 600; M = 15;                   % 600 samples per beat
s = ecgsim(nbeats,L,M);                        % simulated ECG
n = (0:length(s)-1)'; t = n/L;                 % time in seconds

D = 10;  v = 0;
for k=1:D/2-1,
   v = v + (0.5/k^2) * sin(2*pi*k*n/D);        % generate noise
end

y = s + v;                                     % noisy ECG

Q = 50; beta = tan(pi/2/Q);
a = (1-beta)/(1+beta); b = (1+a)/2;            % filter parameters
aD = up([1,-a],D);                             % upsampled denominator coefficients
bD = up([b,-b],D);                             % upsampled numerator coefficients

x = filter(bD,aD,y);                           % filtered ECG

figure; plot(t,y);                             % left graph
figure; plot(t,x, t,s,':');                    % right graph
```

The MATLAB function `ecgsim`, which is part of the OSP toolbox, was used to generate the simulated ECG. It is based on the function `ecg` from [30]. The function `up` is used to upsample the highpass filter's coefficient vectors by a factor of D, generating the coefficient vectors of the notch filter, so that the built-in filtering function `filter` can be used. □

The upsampling operation used in the previous example is the time-domain equivalent of the transformation $z \to z^D$ and it amounts to inserting $D-1$ zeros between any two original filter coefficients. For example, applied to the vector $[h_0, h_1, h_2, h_3]$ with $D = 4$, it generates the upsampled vector:

$$[h_0, h_1, h_2, h_3] \to [h_0, 0, 0, 0, h_1, 0, 0, 0, h_2, 0, 0, 0, h_3]$$

The function `up` implements this operation,

```
g = up(h,D);          % upsampled vector
```

It is similar to MATLAB's built-in function `upsample`, except it does not append $D-1$ zeros at the end. The difference is illustrated by the following example,

```
      up([1,2,3,4],4) = [1  0  0  0  2  0  0  0  3  0  0  0  4]
upsample([1,2,3,4],4) = [1  0  0  0  2  0  0  0  3  0  0  0  4  0  0  0]
```

In addition to enhancing periodic signals or removing periodic interference, comb and notch filters have many other applications. The transformation $z \to z^D$ is widely used in audio signal processing for the design of reverberation algorithms emulating the delays arising from reflected signals within rooms or concert halls or in other types of audio effects [30]. The mapping $z \to z^D$ is also used in multirate signal processing applications, such as decimation or interpolation [30]. The connection to multirate applications can be seen by writing the frequency mapping $\omega' = \omega D$ in terms of the physical frequency f in Hz and sampling rate f_s,

$$\frac{2\pi f'}{f_s'} = \frac{2\pi f}{f_s} D$$

In the signal enhancement context, the sampling rates are the same $f_s' = f_s$, but we have frequency scaling $f' = fD$. On the other hand, in multirate applications, the frequencies remain the same $f' = f$ and the above condition implies the sampling rate change $f_s' = f_s/D$, which can be thought of as decimation by a factor of D from the high rate f_s to the low rate f_s', or interpolation from the low to the high rate.

9.2 *Notch and Comb Filters with Fractional Delay*

The implementation of comb and notch filters requires that the sampling rate be related to the fundamental harmonic by $f_s = Df_1$ with D an integer, so that z^{-D} represents a D-fold multiple delay. In some applications, one may not have the freedom of choosing the sampling rate and the equation $D = f_s/f_1$ may result into a non-integer number.

One possible approach, discussed at the end of this section, is simply to design individual comb/notch filters for each desired harmonic $f_k = kf_1 = kf_s/D$, $k = 1, 2, \dots$, that lies within the Nyquist interval, and then either cascade the filters together in the notch case, or add them in parallel in the comb case.

Another approach is to approximate the desired non-integer delay z^{-D} by an FIR filter and then use the IIR comb/notch structures of Eq. (9.1.9). Separating D into its integer and fractional parts, we may write:

$$D = D_{\text{int}} + d \tag{9.2.1}$$

where $D_{\text{int}} = \text{floor}(D)$ and $0 < d < 1$. The required multiple delay can be written then as $z^{-D} = z^{-D_{\text{int}}} z^{-d}$. The fractional part z^{-d} can be implemented by replacing it with an FIR filter that approximates it, that is, $H(z) \approx z^{-d}$, so that $z^{-D} \approx z^{-D_{\text{int}}} H(z)$. Then, the corresponding IIR comb/notch filters (9.1.9) will be approximated by

$$\begin{aligned} H_{\text{comb}}(z) &= b\,\frac{1 + z^{-D_{\text{int}}} H(z)}{1 - a\,z^{-D_{\text{int}}} H(z)}\,, \quad b = \frac{1-a}{2} \\ H_{\text{notch}}(z) &= b\,\frac{1 - z^{-D_{\text{int}}} H(z)}{1 - a\,z^{-D_{\text{int}}} H(z)}\,, \quad b = \frac{1+a}{2} \end{aligned} \tag{9.2.2}$$

Fig. 9.2.1 shows a possible realization. There exist many design methods for such approximate fractional delay filters [162]. We encountered some in Sec. 3.6. For example,

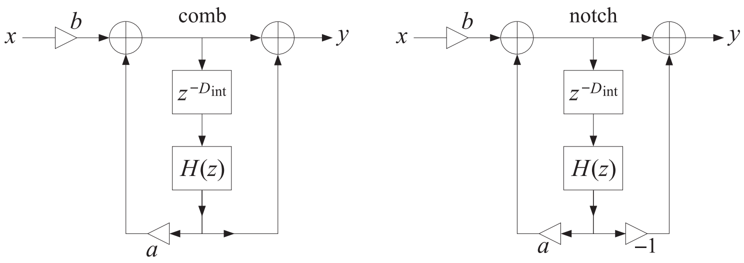


Fig. 9.2.1 Comb and notch filters with fractional delay.

the transfer functions of the causal Lagrange interpolation filters of orders 1 and 2 approximating the required non-integer delay d can be obtained from Eq. (3.6.30),

$$\begin{aligned} H(z) &= d + (1-d)z^{-1} \\ H(z) &= \frac{1}{2}(d-1)(d-2) - d(d-2)z^{-1} + \frac{1}{2}d(d-1)z^{-2} \end{aligned} \tag{9.2.3}$$

Such interpolation filters accurately cover only a fraction, typically 10–20%, of the Nyquist interval, and therefore, would be appropriate only if the first few harmonics are significant. A more effective approach suggested by [168] is to impose linear constraints on the design of $H(z)$ that preserve the required filter response at all the harmonics.

For integer delay D, the comb filter peaks or the notch filter nulls occur at the D-th roots of unity $z_k = e^{j\omega_k}$, $\omega_k = 2\pi k/D$, which satisfy $z_k^D = 1$.

For non-integer D, we require the same constraints for the delay filter $z^{-D_{\text{int}}}H(z)$, that is, $z_k^{-D_{\text{int}}}H(z_k) = 1$, or in terms of the frequency response, $e^{-j\omega_k D_{\text{int}}}H(\omega_k) = 1$, where again $z_k = e^{j\omega_k}$, $\omega_k = 2\pi k/D$. Since $e^{-j\omega_k D} = 1$, we have,

$$e^{-j\omega_k D_{\text{int}}}H(\omega_k) = 1 = e^{-j\omega_k D} = e^{-j\omega_k(D_{\text{int}}+d)} \quad\Rightarrow\quad H(\omega_k) = e^{-j\omega_k d}$$

These are the constraints to be imposed on the design of $H(z)$. In order to obtain a real-valued impulse response for this filter, we must work with the harmonics that lie in the symmetric Nyquist interval, that is, $-\pi \le \omega_k \le \pi$, or,

$$-\pi \le \frac{2\pi k}{D} \le \pi \quad\Rightarrow\quad -\frac{D}{2} \le k \le \frac{D}{2}$$

Writing $D_{\text{int}} = 2p + q$ and $D = D_{\text{int}} + d = 2p + q + d$, with integer p and $q = 0, 1$, the above condition reads:

$$-p - \frac{1}{2}(q+d) \le k \le p + \frac{1}{2}(q+d)$$

Since $0 < d < 1$ and k must be an integer, we obtain,

$$-p \le k \le p \tag{9.2.4}$$

Thus, the design problem is to determine an FIR filter $H(z)$ such that $H(\omega) \approx e^{-j\omega d}$, and subject to the constraints:

$$H(\omega_k) = e^{-j\omega_k d}, \quad -p \le k \le p \tag{9.2.5}$$

When $q = d = 0$, we must choose $-p \le k \le p-1$, because $k = \pm p$ both are mapped onto the Nyquist frequency $\omega = \pm\pi$ and need be counted only once. In this case, of course, we expect the design method to produce the identity filter $H(z) = 1$.

Following [168], we use a constrained least-squares design criterion with the following performance index into which the constraints have been incorporated by means of complex-valued Lagrange multipliers λ_k:

$$\mathcal{J} = \int_{-\alpha\pi}^{\alpha\pi} \left|H(\omega) - e^{-j\omega d}\right|^2 \frac{d\omega}{2\pi} + \sum_{k=-p}^{p}\left[e^{-j\omega_k d} - H(\omega_k)\right]\lambda_k^* + \text{c.c.} = \min \tag{9.2.6}$$

where "c.c." denotes the complex conjugate of the second term. The approximation $H(\omega) \approx e^{-j\omega d}$ is enforced in the least-squares sense over a portion of the Nyquist interval, $[-\alpha\pi, \alpha\pi]$, where typically, $0.9 \le \alpha \le 1$, with $\alpha = 1$ covering the full interval. Assuming an Mth order filter $\mathbf{h} = [h_0, h_1, \dots, h_M]^T$, we can write the frequency response in terms of the $(M+1)$-dimensional vectors,

$$H(\omega) = \sum_{n=0}^{M} h_n e^{-jn\omega} = \mathbf{s}_\omega^\dagger \mathbf{h}, \quad \mathbf{s}_\omega = \begin{bmatrix} 1 \\ e^{j\omega} \\ \vdots \\ e^{jM\omega} \end{bmatrix}, \quad \mathbf{h} = \begin{bmatrix} h_0 \\ h_1 \\ \vdots \\ h_M \end{bmatrix} \tag{9.2.7}$$

Similarly, we can express the gain constraints in the vector form,

$$S^\dagger \mathbf{h} = \mathbf{g} \tag{9.2.8}$$

where S is an $(M+1)\times(2p+1)$ matrix and $\mathbf{g}$ a $(2p+1)$-dimensional column vector defined component-wise by

$$\begin{aligned} S_{nk} &= e^{jn\omega_k}, \quad 0 \le n \le M, \quad -p \le k \le p \\ g_k &= e^{-j\omega_k d}, \quad -p \le k \le p \end{aligned} \tag{9.2.9}$$

that is,

$$S = [\dots, \mathbf{s}_{\omega_k}, \dots], \quad \mathbf{g} = \begin{bmatrix} \vdots \\ e^{-j\omega_k d} \\ \vdots \end{bmatrix} \tag{9.2.10}$$

It follows that the performance index can be written compactly as,

$$\mathcal{J} = \int_{-\alpha\pi}^{\alpha\pi} \left|\mathbf{s}_\omega^\dagger \mathbf{h} - e^{-j\omega d}\right|^2 \frac{d\omega}{2\pi} + \boldsymbol{\lambda}^\dagger(\mathbf{g} - S^\dagger\mathbf{h}) + (\mathbf{g} - S^\dagger\mathbf{h})^\dagger\boldsymbol{\lambda} = \min \tag{9.2.11}$$

where $\boldsymbol{\lambda} = [\dots, \lambda_k, \dots]^T$ is the vector of Lagrange multipliers. Expanding the first term of $\mathcal{J}$, we obtain,

$$\mathcal{J} = \mathbf{h}^\dagger R\mathbf{h} - \mathbf{h}^\dagger\mathbf{r} - \mathbf{r}^\dagger\mathbf{h} + \alpha + \boldsymbol{\lambda}^\dagger(\mathbf{g} - S^\dagger\mathbf{h}) + (\mathbf{g}^\dagger - \mathbf{h}^\dagger S)\boldsymbol{\lambda} = \min \tag{9.2.12}$$

where the matrix R and vector $\mathbf{r}$ are defined by,

$$R = \frac{1}{2\pi}\int_{-\alpha\pi}^{\alpha\pi} \mathbf{s}_\omega \mathbf{s}_\omega^\dagger\, d\omega\,, \quad \mathbf{r} = \frac{1}{2\pi}\int_{-\alpha\pi}^{\alpha\pi} \mathbf{s}_\omega e^{-j\omega d}\, d\omega \tag{9.2.13}$$

and component-wise,

$$\begin{aligned} R_{nm} &= \int_{-\alpha\pi}^{\alpha\pi} e^{j\omega(n-m)}\,\frac{d\omega}{2\pi} = \frac{\sin\big(\alpha\pi(n-m)\big)}{\pi(n-m)}\,, \quad n,m = 0,1,\dots,M \\ r_n &= \int_{-\alpha\pi}^{\alpha\pi} e^{j\omega(n-d)}\,\frac{d\omega}{2\pi} = \frac{\sin\big(\alpha\pi(n-d)\big)}{\pi(n-d)}\,, \quad n = 0,1,\dots,M \end{aligned} \tag{9.2.14}$$

We note that for $\alpha = \pi$, R reduces to the identity matrix. The optimal solution for $\mathbf{h}$ is obtained by setting the gradient of $\mathcal{J}$ to zero:

$$\frac{\partial\mathcal{J}}{\partial\mathbf{h}^*} = R\mathbf{h} - \mathbf{r} - S\boldsymbol{\lambda} = 0 \quad\Rightarrow\quad \mathbf{h} = R^{-1}\mathbf{r} + R^{-1}S\,\boldsymbol{\lambda} = \mathbf{h}_{\text{u}} + R^{-1}S\boldsymbol{\lambda}$$

where $\mathbf{h}_{\text{u}} = R^{-1}\mathbf{r}$ is the unconstrained solution of the least-squares problem. The Lagrange multiplier $\boldsymbol{\lambda}$ can be determined by multiplying both sides by $S^\dagger$ and using the constraint (9.2.8):

$$\mathbf{g} = S^\dagger\mathbf{h} = S^\dagger\mathbf{h}_{\text{u}} + S^\dagger R^{-1}S\,\boldsymbol{\lambda} \quad\Rightarrow\quad \boldsymbol{\lambda} = (S^\dagger R^{-1}S)^{-1}(\mathbf{g} - S^\dagger\mathbf{h}_{\text{u}})$$

Finally, substituting $\boldsymbol{\lambda}$ into the solution for $\mathbf{h}$, we obtain,

$$\boxed{\mathbf{h} = \mathbf{h}_{\text{u}} + R^{-1}S(S^\dagger R^{-1}S)^{-1}(\mathbf{g} - S^\dagger\mathbf{h}_{\text{u}})} \tag{9.2.15}$$

This type of constrained least-squares problem appears in many applications. We will encounter it again in the context of designing linearly constrained minimum variance beamformers for interference suppression, and in the problem of optimum stock portfolio design.

The MATLAB function `combfd` implements the above design method. Its inputs are the fractional period D, the order M of the filter $H(z)$, the comb/notch pole parameter a, and the Nyquist factor α,

```
[bD,aD,h,zmax] = combfd(D,M,a,alpha);     % comb/notch filter design with fractional delay
```

Entering the parameter a as negative indicates the design of a notch filter. The outputs `bD,aD` are the coefficients of the numerator and denominator polynomials of the comb/notch filters (9.2.2):

$$\begin{aligned} B_D(z) &= b\big[1 \pm z^{-D_{\text{int}}}H(z)\big] \\ A_D(z) &= 1 - a\,z^{-D_{\text{int}}}H(z) = 1 - az^{-D_{\text{int}}}(h_0 + h_1z^{-1} + \cdots + h_Mz^{-M}) \end{aligned} \tag{9.2.16}$$

The output `h` is the impulse response vector $\mathbf{h}$, and `zmax` is the maximum pole radius of the denominator filter $A_D(z)$, which can be used to monitor the stability of the designed comb/notch filter. The pole parameter a can be fixed using the bandwidth equation (9.1.11), which is still approximately valid.

Fig. 9.2.2 shows a design example with fractional period $D = 9.1$, so that $D_{\text{int}} = 9$ and $d = 0.1$. The other parameters were $M = 8$, $a = R^D$ with $R = 0.95$, and $\alpha = 1$.

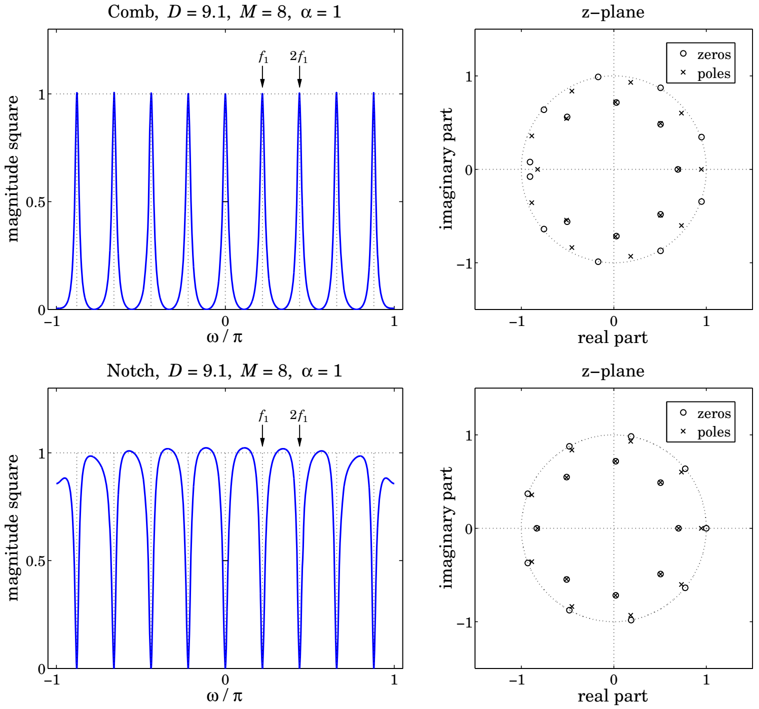


Fig. 9.2.2 Comb and notch filters with $D = 9.1$, and their pole/zero patterns.

The 3-dB widths obtained from Eq. (9.1.11) are indicated on the graphs by the two short horizontal lines at the half-power levels. The frequency plots are over the symmetric interval $-\pi \le \omega \le \pi$. The comb peaks have unity gain at the harmonics. For the notch case, the response between the notch dips is not very flat, but can be made flatter by decreasing the bandwidth, i.e., increasing the parameter a towards unity.

The right graphs depict the pole/zero patterns of the polynomials $B_D(z)$ and $A_D(z)$. These polynomials have orders $D_{\text{int}} + M = 9 + 8 = 17$. For the comb filter, we observe how the $D_{\text{int}} = 9$ poles arrange themselves around the unit circle at the harmonic frequencies, while the remaining 8 poles lie inside the unit circle.

The zeros of $B_D(z)$ also arrange themselves in two groups, 8 of them lying on the unit circle halfway between the comb peak poles, and the remaining 9 lying inside the unit circle, with a group of 7 poles and 7 zeros almost falling on top of each other, almost canceling each other.

A similar pattern occurs for the notch filter, except now the notch zeros at the harmonics have poles lying almost behind them in order to sharpen the notch widths, while the remaining pole/zero pairs arrange themselves inside the unit-circle as in the comb case.

Generally, this design method tends to work well whenever D is near an odd integer, such as in the above example and in the top graphs of Fig. 9.2.4, which have $D = 9.1$ and $D = 8.9$. The method has some difficulty when D is near an even integer, such as $D = 9.9$ or $D = 8.1$, as shown in Figs. 9.2.3 and the bottom of 9.2.4.

In such cases, the method tends to place a pole or pole/zero pair on the real axis near $z = -1$ resulting in an unwanted peak or dip at the Nyquist frequency $\omega = \pi$. Such poles are evident in the pole/zero plots of Fig. 9.2.3. If D were exactly an even integer, then such pole/zero pair at Nyquist would be present, but for non-integer D, the Nyquist frequency is not one of the harmonics. Removing that pole/zero pair from the design, does not improve the problem.

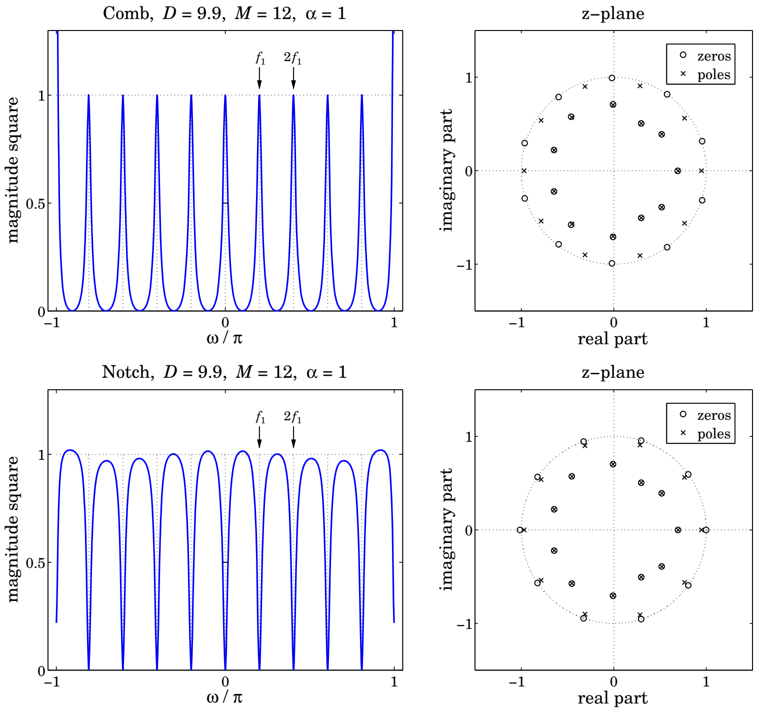


Fig. 9.2.3 Comb and notch filters with $D = 9.9$, and their pole/zero patterns.

The MATLAB code for generating the magnitude responses and pole/zero plots is the same for all three figures. In particular, Fig. 9.2.2 was generated by,

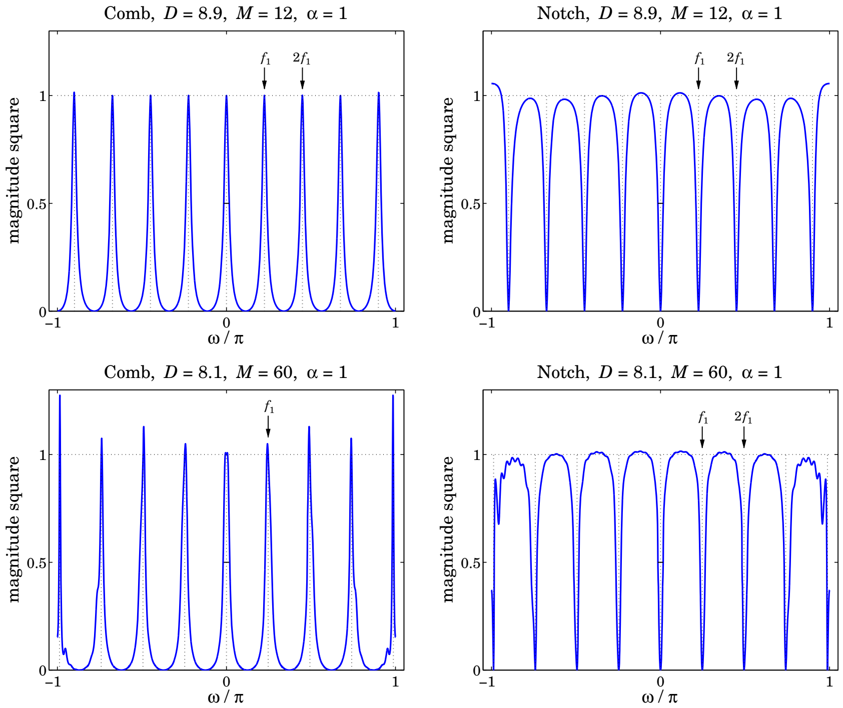


Fig. 9.2.4 Comb and notch filters with $D = 8.9$ and $D = 8.1$.

```
f = linspace(-1,1,4001); w = pi*f;                     % frequency range

D=9.1; R=0.95; a=R^D; M=8; alpha=1;                    % design parameters
beta = (1-a)/(1+a); Dw = 4/D * atan(beta);             % bandwidth calculation

[bD,aD,h,zmax] = combfd(D,M,a,alpha);                  % comb, param a entered as positive
Hcomb = abs(freqz(bD,aD,w)).^2;                        % comb's magnitude rresponse

figure; plot(w/pi,Hcomb); figure; zplane(bD,aD);       % upper two graphs

[bD,aD,h,zmax] = combfd(D,M,-a,alpha);                 % notch, param a entered as negative
Hnotch = abs(freqz(bD,aD,w)).^2;

figure; plot(w/pi,Hnotch); figure; zplane(bD,aD);      % lower two graphs
```

Parallel and Cascade Realizations

As we mentioned in the beginning of the previous section, an alternative approach is to design individual peak or notch filters at the harmonics and then combine the filters in parallel for the comb case, and in cascade for the notch case. Fig. 9.2.5 illustrates this type of design for the two “difficult” cases of $D = 9.9$ and $D = 8.1$ using second-order peak/notch filters designed to have the same bandwidth as in Fig. 9.2.3.

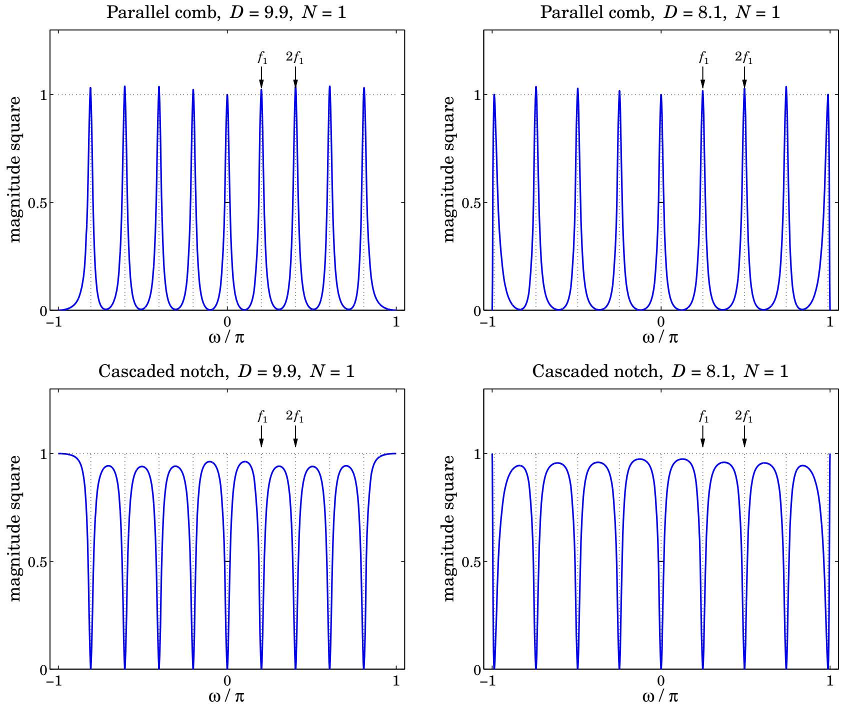


Fig. 9.2.5 Second-order parallel comb and cascaded notch filters.

Let $H_k(z)$ be the peak/notch filter for the kth harmonic $\omega_k = k\omega_1 = 2\pi k/D$, $k = 0, 1, \dots, p$ and its negative $-\omega_k$. Then, the transfer functions of the comb and notch filters will be:

$$H_{\text{comb}}(z) = \sum_{k=0}^{p} H_k(z)\,, \qquad H_{\text{notch}}(z) = \prod_{k=0}^{p} H_k(z) \tag{9.2.17}$$

In their simplest form, the individual filters $H_k(z)$ are second-order and can be obtained from the lowpass and highpass filters (9.1.8) by the lowpass-to-bandpass z-domain transformation [30,595]:

$$z \;\to\; z' = \frac{z(\cos\omega_k - z)}{1 - z\cos\omega_k} \tag{9.2.18}$$

The resulting second-order peaking and notch filters are [30], for $k = 1, 2, \dots, p$:

$$\begin{aligned} \text{peak:}\quad & H_k(z) = b\,\frac{1 - z^{-2}}{1 - (1+a)\cos\omega_k\, z^{-1} + a z^{-2}}\,, \qquad b = \frac{1-a}{2} \\ \text{notch:}\quad & H_k(z) = b\,\frac{1 - 2\cos\omega_k\, z^{-1} + z^{-2}}{1 - (1+a)\cos\omega_k\, z^{-1} + a z^{-2}}\,, \qquad b = \frac{1+a}{2} \end{aligned} \tag{9.2.19}$$

The filter parameter a is fixed in terms of the 3-dB width of the peak or the notch by,

$$\tan\left(\frac{\Delta\omega}{2}\right) = \frac{1-a}{1+a} = \beta \tag{9.2.20}$$

For $k = 0$, we may use the first-order lowpass/highpass filters of Eq. (9.1.8) without any z-domain transformation. But in order for their 3-dB frequency to match the specified 3-dB width $\Delta\omega$, their parameter a must be redefined as follows:

$$\tan\left(\frac{\Delta\omega}{4}\right) = \frac{1-a}{1+a} = \beta \tag{9.2.21}$$

To clarify the construction, we give below the MATLAB code for generating the left graphs of Fig. 9.2.5,

```
f = linspace(-1,1,4001); w = pi*f;                % frequency range −π ≤ ω ≤ π

D = 9.9; p = floor(D/2); w1 = 2*pi/D;             % design parameters
R = 0.95; a = R^2;

beta = (1-a)/(1+a); dw = 2*atan(beta);            % 3-dB width calculation
beta0 = tan(dw/4);  a0 = (1-beta0)/(1+beta0);     % bandwidth parameter for k = 0 section

A = [1, -a0, 0];                                  % denominator coefficients for k = 0
Bcomb =  [1, 1, 0] * (1-a0)/2;                    % numerator coefficients for k = 0
Bnotch = [1,-1, 0] * (1+a0)/2;

Hcomb  = freqz(Bcomb,A,w);                        % k = 0 section, H0(ω)
Hnotch = freqz(Bnotch,A,w);

for k=1:p,                                        % non-zero harmonics
   A = [1, -(1+a)*cos(k*w1), a];                  % denominator of Hk(z)
   Bcomb =  [1, 0, -1] * (1-a)/2;                 % numerator of peak Hk(z)
   Bnotch = [1, -2*cos(k*w1), 1] * (1+a)/2;       % numerator of notch Hk(z)
   Hcomb =  Hcomb + freqz(Bcomb,A,w);             % add in parallel for comb
   Hnotch = Hnotch .* freqz(Bnotch,A,w);          % cascade for notch
end

figure; plot(w/pi, abs(Hcomb).^2, '-');           left graphs
figure; plot(w/pi, abs(Hnotch).^2,'-');
```

It is evident from Fig. 9.2.5 that this design method is flexible enough to correctly handle any values of the fractional period D. However, because of the mutual interaction between the individual filters, the peaks of the comb do not quite have unity gains, and the segments between the nulls of the notch filter are not quite flat.

This behavior can be fixed by decreasing the width $\Delta\omega$. However, for a fixed value of $\Delta\omega$, the only way to improve the response is by using higher-order filters. For example,

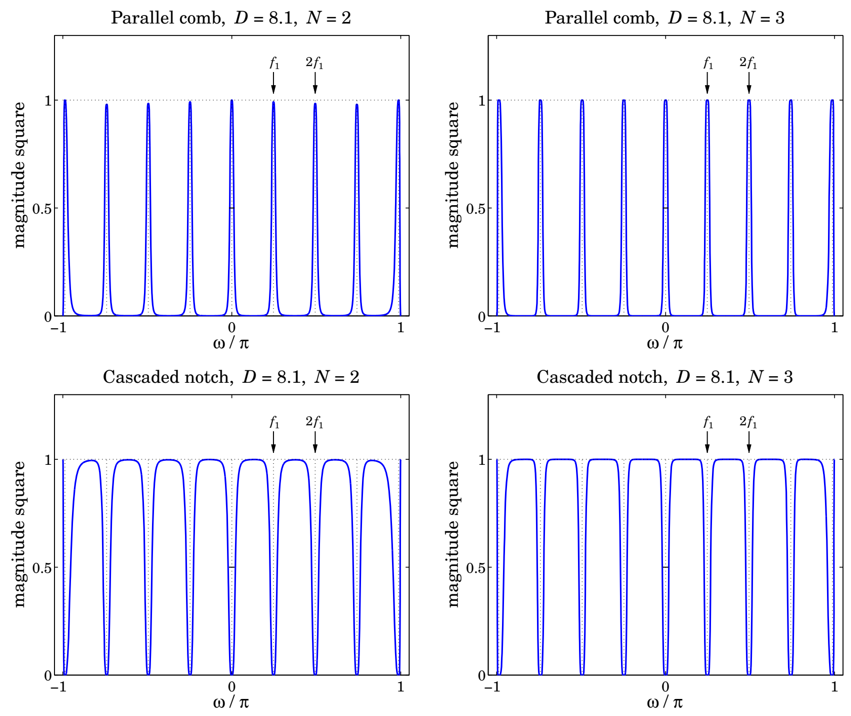


Fig. 9.2.6 High-order Butterworth parallel comb and cascaded notch filters.

Fig. 9.2.6 illustrates the cases of designing the individual filters using Butterworth filter prototypes of orders $N = 2$ and $N = 3$, whereas Fig. 9.2.5 corresponds to $N = 1$.

The following MATLAB code illustrates the generation of the left graphs in Fig. 9.2.6, and uses the functions `hpeq` and `frespc` from the high-order equalizer design toolbox in [595], which is also included in the OSP toolbox:

```
f = linspace(-1,1,4001); w = pi*f;

D = 8.1; p = floor(D/2); w1 = 2*pi/D;
R = 0.95; a = R^2;

beta = (1-a)/(1+a); dw = 2*atan(beta);          % 3-dB width

N = 2; GB = -20*log10(2);                        % Butterworth order and bandwidth gain

[B0,A0] = hpeq(N, -inf, 0, GB, 0, dw/2);         % k = 0 for comb, cutoff = half-bandwidth
Hcomb = frespc(B0,A0,w);

for k=1:p
   [B,A] = hpeq(N, -inf, 0, GB, k*w1, dw);       % non-zero harmonics
    Hcomb = Hcomb + frespc(B,A,w);               % add in parallel
end
```

```
figure; plot(w/pi,abs(Hcomb).^2,'-');            upper-left graph

[B0,A0] = hpeq(N, 0, -inf, GB, 0, dw/2);         % k = 0 for notch
Hnotch = frespc(B0,A0,w);

for k=1:p
   [B,A] = hpeq(N, 0, -inf, GB, k*w1, dw);
    Hnotch = Hnotch .* frespc(B,A,w);            % cascade in series
end

figure; plot(w/pi,abs(Hnotch).^2,'-');           % lower-left graph
```

The higher-order designs can also be based on Chebyshev or elliptic filters. In all cases, the starting point is a lowpass (or highpass) analog prototype filter $H_a(s)$, which is transformed into a peaking (or notch) filter centered at ω_k using the s-to-z domain bandpass transformation [30,595]:

$$H(z)=H_a(s)\,,\qquad s=\frac{z'-1}{z'+1}=\frac{z^2-2\cos\omega_k\,z+1}{z^2-1} \tag{9.2.22}$$

where z' is given by Eq. (9.2.18). For example, the analog Butterworth prototype filters of orders $N = 1, 2, 3$ are:

$$H_a(s)=\frac{\beta}{\beta+s}\,,\quad H_a(s)=\frac{\beta^2}{\beta^2+\sqrt{2}\beta s+s^2}\,,\quad H_a(s)=\frac{\beta}{\beta+s}\cdot\frac{\beta^2}{\beta^2+\beta s+s^2}$$

Similarly, for the notch filters, the analog prototypes are the highpass filters:

$$H_a(s)=\frac{s}{\beta+s}\,,\quad H_a(s)=\frac{s^2}{\beta^2+\sqrt{2}\beta s+s^2}\,,\quad H_a(s)=\frac{s}{\beta+s}\cdot\frac{s^2}{\beta^2+\beta s+s^2}$$

For arbitrary N, the Butterworth lowpass and highpass filters are:

$$H_a(s)=\left[\frac{\sigma}{\beta+s}\right]^r\prod_{i=1}^{L}\left[\frac{\sigma^2}{\beta^2+2\beta s\sin\phi_i+s^2}\right],\quad \phi_i=\frac{\pi(2i-1)}{2N} \tag{9.2.23}$$

where $N = 2L + r$, with integer L and $r = 0, 1$, and with $\sigma = \beta$ in the lowpass case, and $\sigma = s$ in the highpass one. The parameter β is related to the 3-dB width through $\beta = \tan(\Delta\omega/2)$. The filters of Eq. (9.2.19) are obtained by applying the transformation (9.2.22) to the $N = 1$ case.

Each peaking or notching filter is the cascade of L second-order sections in s or fourth-order sections in z (and possibly a second-order section in z if $r = 1$). The function `frespc` is used to calculate the corresponding frequency responses in such cascaded form. Further details on high-order designs and a description of the function `hpeq` can be found in [595].

9.3 *Signal Averaging*

Signal averaging is a technique for estimating a repetitive signal in noise. Evoked biological signals, GPS, and radar were some applications mentioned at the beginning of this chapter. A variant of the method can also be used to deseasonalize business, social, and climate data—the difference being here that the non-periodic part of the measured signal is not only noise but it can also contain a trend component. The typical assumed noise model in signal averaging has the form:

$$y_n = s_n + v_n \tag{9.3.1}$$

where s_n is periodic with some period D, assumed to be an integer, and v_n is zero-mean white noise. The periodic signal s_n can be extracted by filtering y_n through any comb filter, such as the IIR filter of Eq. (9.1.9).

Signal averaging is equivalent to comb filtering derived by applying the D-fold replicating transformation $z \to z^D$ to an ordinary, length-N, lowpass FIR averaging filter:

$$H_{\text{LP}}(z) = \frac{1}{N}\left[1 + z^{-1} + z^{-2} + \cdots + z^{-(N-1)}\right] = \frac{1}{N}\,\frac{1 - z^{-N}}{1 - z^{-1}} \tag{9.3.2}$$

The definition $H(z) = H_{\text{LP}}(z^D)$, then gives the comb filter:

$$H(z) = \frac{1}{N}\left[1 + z^{-D} + z^{-2D} + \cdots + z^{-(N-1)D}\right] = \frac{1}{N}\,\frac{1 - z^{-ND}}{1 - z^{-D}} \tag{9.3.3}$$

The latter equation shows that $H(z)$ has zeros at all the (ND)-th roots of unity that are not D-th roots of unity. At the latter, the filter has unity-gain peaks.

An example is shown in Fig. 9.3.1, with period $D = 10$ and $N = 5$ and $N = 10$. The comb peaks are at the D-th roots of unity $\omega_k = 2\pi k/D = 2\pi k/10$, $k = 0, 1, \dots, 9$. The 3-dB width of the peaks is indicated on the graphs by the short horizontal lines at the half-power level centered around the first harmonic.

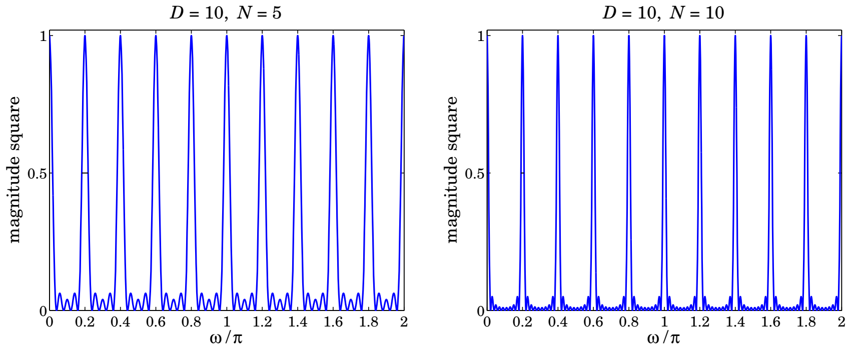


Fig. 9.3.1 Signal averaging filters with $D = 10$ and $N = 5, 10$.

The 3-dB width is given by

$$\Delta\omega = 0.886\,\frac{2\pi}{ND} \tag{9.3.4}$$

which follows from the frequency response of $H(z)$:

$$H(\omega) = \frac{1}{N}\,\frac{1 - e^{-j\omega ND}}{1 - e^{-j\omega D}} = \frac{1}{N}\,\frac{\sin(ND\omega/2)}{\sin(D\omega/2)}\,e^{-j(\omega(N-1)D/2} \tag{9.3.5}$$

Thus, the peaks get narrower with increasing number N of averaging periods. This has the effect of decreasing the noise, while letting through the periodic signal s_n.

The signal averaging interpretation can be seen from the time-domain operation of the filter. The corresponding output is the estimated periodic signal,

$$\hat{s}_n = \frac{1}{N}\left[y_n + y_{n-D} + y_{n-2D} + \cdots + y_{n-(N-1)D}\right] \tag{9.3.6}$$

Inserting $y_n = s_n + v_n$ and using the periodicity property $s_{n-D} = s_n$, we obtain,

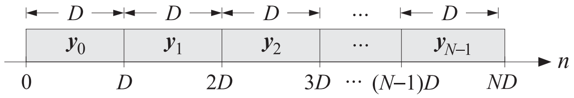
$$\hat{s}_n = s_n + \frac{1}{N}\left[v_n + v_{n-D} + \cdots + v_{n-(N-1)D}\right] \equiv s_n + \hat{v}_n \tag{9.3.7}$$

Because v_n was assumed to be stationary uncorrelated white noise, the variance of the filtered noise $\hat{v}_n$ will be reduced by a factor of N,

$$\sigma_{\hat{v}}^2 = \frac{1}{N^2}\left[\text{var}(v_n) + \text{var}(v_{n-D}] + \cdots + \text{var}(v_{n-(N-1)D})\right] = \frac{1}{N^2}(N\sigma_v^2) = \frac{1}{N}\sigma_v^2 \tag{9.3.8}$$

which implies that the NRR of the comb filter is $\mathcal{R} = 1/N$. Thus, by choosing N sufficiently large, the noise can be reduced, enabling the estimation of s_n.

Let the signal y_n be collected over N periods, that is, $0 \le n \le ND - 1$, and divide the signal into N length-D period segments as shown below,



The filtering operation (9.3.6) can be thought of as the averaging the N subblocks together. Indeed, let $y_i(n) = y_{iD+n}$, for $n = 0, 1, \dots, D-1$, be the samples within the i-th subblock, $i = 0, 1, \dots, N-1$. Then, we have

$$\frac{1}{N}\sum_{i=0}^{N-1} y_i(n) = \frac{1}{N}\sum_{i=0}^{N-1} y_{iD+n} = \frac{1}{N}\sum_{k=0}^{N-1} y_{(N-1)D+n-kD} = \hat{s}_{(N-1)D+n} \tag{9.3.9}$$

or, in words, the last D filter output samples, that is, over the period $[(N-1)D, ND-1]$, are the average of the samples over the last N periods. This can also be seen more simply by writing (9.3.6) in recursive form, which follows from Eq. (9.3.3),

$$\hat{s}_n = \hat{s}_{n-D} + \frac{1}{N}(y_n - y_{n-ND}) = \hat{s}_{n-D} + \frac{1}{N}y_n\,, \qquad 0 \le n \le ND - 1 \tag{9.3.10}$$

where the term y_{n-ND} was dropped because of the causal nature of y_n and the assumed range of n, that is, $0 \le n \le ND - 1$. Thus, Eq. (9.3.10) shows that $\hat{s}_n$ is the accumulation and averaging of the N period segments of y_n.

The MATLAB implementation of signal averaging is straightforward, for example, assuming that the array y has length at least ND,

```
s = 0;
for i=0:N-1,
   yi = y(i*D+1 : i*D+D);          % extract i-th period
   s = s + yi;                     % accumulate i-th period
end
s = s/N;                           % average of N periods
```

So far we have not imposed the constraint $S_0 = s_n + s_{n-1} + \cdots + s_{n-D+1} = 0$. If in addition to the noise component v_n, there is a slowly-varying background or trend present, say, t_n, so that the observation signal is $y_n = s_n + t_n + v_n$, then we may associate the constant S_0 with the trend and assume that $S_0 = 0$. To guarantee this constraint, we may subtract from each block $y_i(n)$ its local average, and compute the estimated periodic component by:

$$\hat{s}_n = \frac{1}{N}\sum_{i=0}^{N-1}\big[y_i(n) - \mu_i\big]\,, \quad \mu_i = \frac{1}{D}\sum_{n=0}^{D-1} y_i(n) \tag{9.3.11}$$

which does satisfy $S_0 = 0$. By replicating the μ_i by D times within the i-th time period $[iD, iD + D - 1]$, and stringing the replicated values together over all the periods, we obtain a step-wise estimate of the trend component t_n. The following MATLAB code illustrates how to do that:

```
y = y(:);
L = length(y);
N = floor(L/D);                      % number of periods in y
r = mod(L,D);                        % L = ND + r

s = 0;
for i=0:N-1,
   yi = y(i*D+1 : i*D+D);            % i-th period
   m(i+1) = mean(yi);                % mean to be removed
   s = s + yi - m(i+1);              % accumulate i-th period
end
s = s / N;                           % estimated period

ys = repmat(s,N,1);                  % replicate N periods
ys(end+1:end+r) = s(1:r);            % extend to length L by appending a portion of s

yt = repmat(m,D,1); yt = yt(:);      % repeat each mean D times within its period
yt(end+1:end+r) = yt(end);           % extend to length L by replicating last mean r times
```

where ys represents the estimated periodic signal, replicated over N periods, and yt is the estimated step-function trend. These above steps have been incorporated into the MATLAB function sigav:

```
[ys,s,yt] = sigav(y,D);     % signal averaging
```

Example 9.3.1: Fig. 9.3.2 shows a simulated signal averaging example. The period is $D = 10$ and the total number of periods $N = 100$. The graphs display only the first 10 periods to improve visibility. The periodic signal was superimposed on a slowly-varying trend and noise was added:

$$y_n = s_n + t_n + v_n\,, \quad s_n = 0.5\sin\left(\frac{4\pi n}{D}\right) + 0.5\sin\left(\frac{6\pi n}{D}\right), \quad t_n = \sin\left(\frac{2\pi n}{10D}\right)$$

where $n = 0, 1, \dots, ND - 1$, and v_n is zero-mean, unit-variance, white noise. The upper row shows the noise-free case (with $v_n = 0$). The upper-right graph shows the periodic signal s_n. The estimated one resulting from the output of sigav is essentially identical to s_n and thus not displayed. The step-function estimated trend is shown on the upper-left.

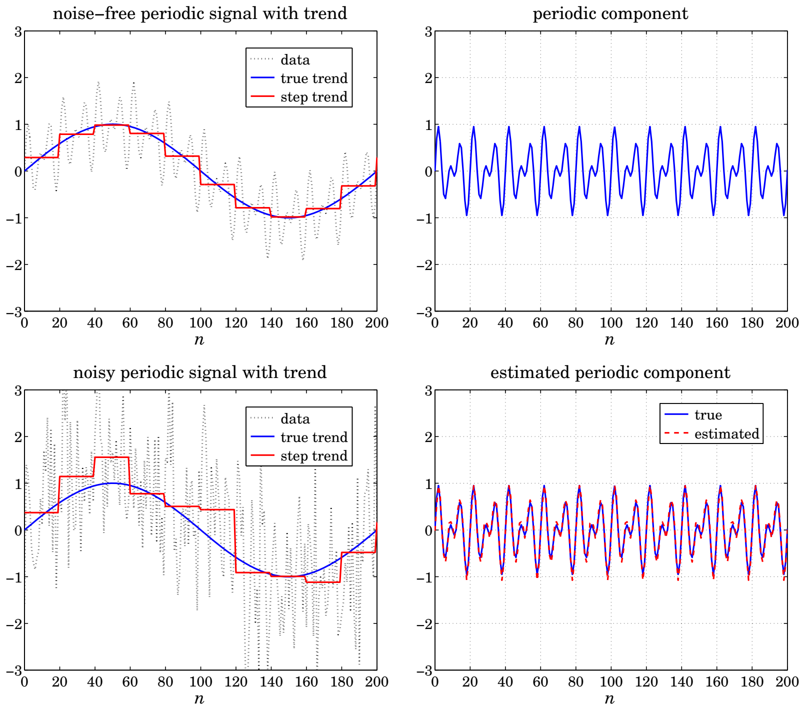


Fig. 9.3.2 Signal averaging of noisy periodic signal with slowly-varying trend.

The lower-left graph shows the noisy case, including the estimated step-trend signal. The lower-right graph shows the estimated periodic signal from the output of sigav. The following MATLAB code illustrates the generation of the bottom graphs:

```
D = 20; N = 100; n = 0:N*D;

s = (sin(4*pi*n/D) + sin(6*pi*n/D))/2;          % periodic component
t = sin(2*pi*n/D/10);                           % trend component

seed = 2008; randn('state',seed);
v = randn(size(n));

y = s + t + v;                                  % noisy observations

[ys,p,yt] = sigav(y,D);                         % signal averaging, p = one period

figure; plot(n,y,'--', n,t,'-.', n,yt,'-');     % yt is the estimated trend
xlim([0,200]);                                  % show only the first 10 periods
figure; plot(n,ys, '-');                        % estimated periodic component
xlim([0,200]);
```

Example 9.3.2: *Housing Starts.* Fig. 9.3.3 shows the application of signal averaging to the monthly, not seasonally adjusted, new privately-owned housing starts, for the 25 year period from January 1984 to December 2008. The data are from the US Census Bureau from the web link: `http://www.census.gov/ftp/pub/const/starts_cust.xls`.

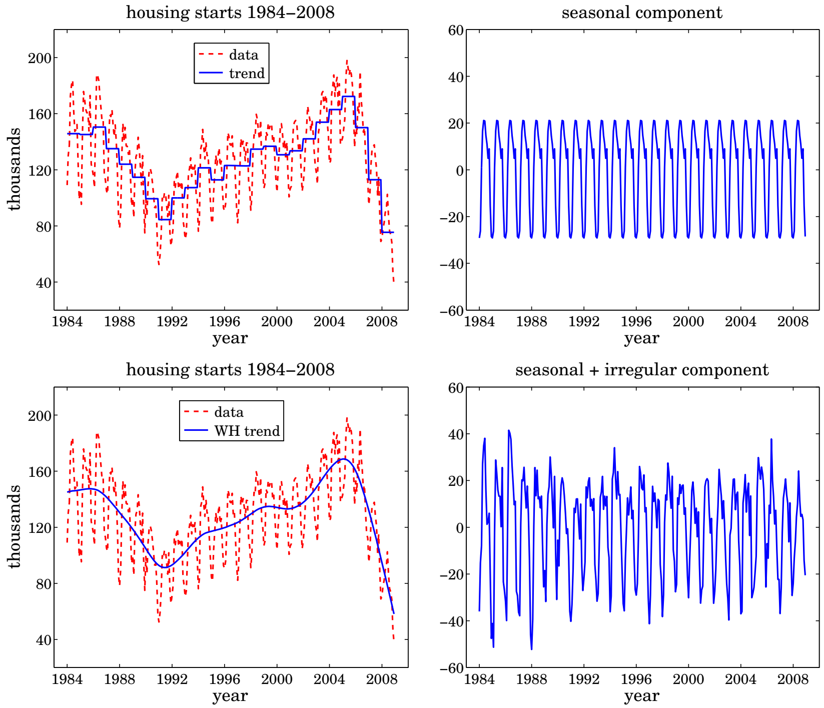


Fig. 9.3.3 Signal averaging and smoothing of monthly housing data.

The upper graphs show the estimated step-wise trend and the seasonal, periodic, component. Although there is clear annual periodicity in the data, the signal averaging method is not the best approach to this application because it does not result into a smooth trend. We consider better methods to deseasonalize such data in the next sections.

As an alternative method, the bottom graphs show the application of the Whittaker Henderson smoothing method to estimate the smooth trend. The optimal smoothing parameter was determined by the GCV criterion to be $\lambda = 6850$ and the smoothing order was $s = 2$.

The difference between the raw data and the estimated trend represents the seasonal plus irregular component and is plotted in the bottom-right graph. Further application of signal averaging to this component will generate an estimate of the seasonal component. It is not plotted because it is essentially identical to that shown in the upper-right graph.

The following MATLAB code illustrates the generation of the four graphs, including, but commented out, the computation of the seasonal part for the bottom graphs:

```
Y = loadfile(’newhouse.dat’);            % data file available in the OSP toolbox
```

```
i = find(Y(:,1)==109.1);                 % finds the beginning of the year 1984

y = Y(i:end-4,1);                        % keep data from Jan.1984 to Dec.2008
t = taxis(y,12,1984);                    % define time axis

[ys,s,yt] = sigav(y,12);                 % signal averaging with period 12

figure; plot(t,y,’--’, t,yt,’-’);        % upper-left graph
figure; plot(t,ys,’-’);                  % upper-right graph

s = 2; la = 6800:2:6900;                 % smoothing order and search-range of λ’s
[gcv,lopt] = whgcv(y,la,s);              % optimum smoothing parameter, λ_opt = 6850

yt = whsm(y,lopt,s);                     % Whittaker-Henderson smoothing
ysi = y-yt;                              % seasonal + irregular component
% ys = sigav(ysi,12);                    % seasonal component, not shown

figure; plot(t,y,’--’, t,yt,’-’);        % bottom-left graph
figure; plot(t,ysi,’-’);                 % bottom-right graph
% figure; plot(t,ys,’-’);                % essentially the same as upper-right graph
```

9.4 *Ideal Seasonal Decomposition Filters*

A possible approach for separating the three components of the signal $y_n = s_n + t_n + v_n$ is to first estimate the trend t_n using a lowpass filter, and then extract the seasonal component s_n by applying a comb filter to the residual $r_n = y_n - t_n = s_n + v_n$, which consists of the seasonal and irregular parts.

The technique assumes of course that the trend is a slowly-varying, low-frequency, signal. Fig. 9.4.1 illustrates some typical frequency spectra for the three components and the ideal filters that might be used to extract them.

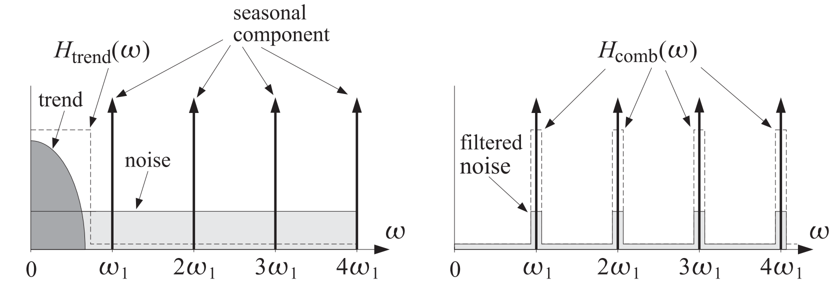


Fig. 9.4.1 Ideal filters for decomposition into trend and seasonal components.

Let $H_{\text{trend}}(z)$ be the trend-extraction filter and $H_{\text{comb}}(z)$ the comb filter with peaks at the seasonal harmonics (excluding the one at DC). Then, the filtering equations for extracting the three components from y_n can be expressed in the z-domain as follows:

$$
\begin{aligned}
T(z) &= H_{\text{trend}}(z)Y(z)\\
R(z) &= S(z)+V(z)= Y(z)-T(z)= \big[1-H_{\text{trend}}(z)\big]Y(z)\\
S(z) &= H_{\text{comb}}(z)R(z)= H_{\text{comb}}(z)\big[1-H_{\text{trend}}(z)\big]Y(z)\equiv H_S(z)Y(z)\\
V(z) &= R(z)-S(z)= \big[1-H_{\text{comb}}(z)\big]\big[1-H_{\text{trend}}(z)\big]Y(z)\equiv H_I(z)Y(z)
\end{aligned}
$$

where $Y(z), S(z), T(z), V(z), R(z)$ are the z-transforms of y_n, s_n, t_n, v_n, r_n. Thus, the filters for extracting the three components are:

$$
\begin{aligned}
H_T(z) &= H_{\text{trend}}(z) && \text{(trend)}\\
H_S(z) &= H_{\text{comb}}(z)\big[1-H_{\text{trend}}(z)\big] && \text{(seasonal)}\\
H_I(z) &= \big[1-H_{\text{comb}}(z)\big]\big[1-H_{\text{trend}}(z)\big] && \text{(irregular)}
\end{aligned}
\tag{9.4.1}
$$

The three filters satisfy the complementarity property:

$$
H_T(z)+H_S(z)+H_I(z)=1 \tag{9.4.2}
$$

In Example 9.3.2, we followed exactly this approach where the trend filter was implemented as a Whittaker-Henderson smoother and the comb filter as a signal averager. Other possibilities exist for these filters and a lot of research has gone into making choices that try to balance a good filter response versus the ability to work well with short data records, including the handling of the end-point problem.

Example 9.4.1: *Housing Starts.* The housing starts signal considered in Example 9.3.2 displays the typical frequency spectra shown in Fig. 9.4.1.

The left graph in Fig. 9.4.2 shows the corresponding magnitude spectrum of the original data signal y_n, normalized to unity maximum and plotted over the symmetric Nyquist interval $[-\pi,\pi]$ in units of the fundamental harmonic $\omega_1 = 2\pi/12$. The spectrum is dominated by the low-frequency trend signal. The right graph shows the spectrum of the seasonal plus irregular component $r_n = y_n - t_n = s_n + v_n$, which displays the harmonics more clearly.

The following MATLAB code illustrates the computation of the spectra:

```
Y = loadfile('newhouse.dat');          % data file available in the OSP toolbox
i = find(Y(:,1)==109.1);               % finds the beginning of the year 1984
y = Y(i:end-4,1);                      % keep data from Jan.1984 to Dec.2008

s = 2; lopt = 6850                     % use optimum λ from Example 9.3.2
yt = whsm(y,lopt,s);                   % Whittaker-Henderson smoothing
ysi = y - yt;                          % seasonal + irregular component

k = linspace(-6,6,1201); w = 2*pi*k/12;               % frequency ω = kω1
L = length(y);
wind = 0.54 - 0.46*cos(2*pi*(0:L-1)/(L-1))';          % Hamming window

Y = abs(freqz(y.*wind,1,w));   Y = Y/max(Y);          % normalized spectrum
Ysi = abs(freqz(ysi.*wind,1,w)); Ysi = Ysi/max(Ysi);

figure; plot(k,Y); figure; plot(k,Ysi);               % left and right graphs
```

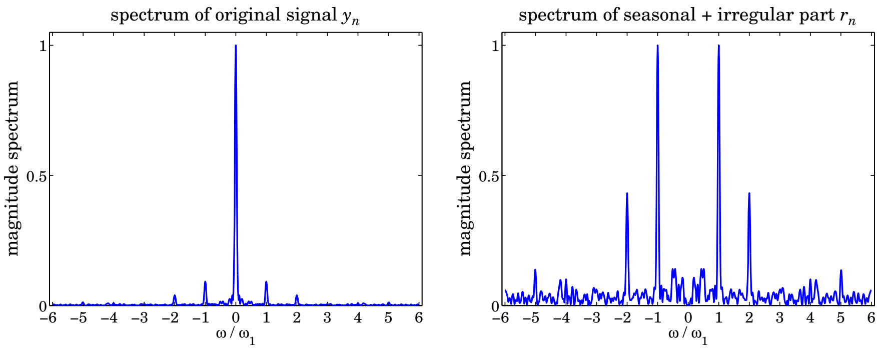


Fig. 9.4.2 Spectra of monthly housing data with and without trend.

The signals were windowed by a Hamming window prior to computing their DTFTs. □

Ideally, it does not matter if $H_{\text{comb}}(z)$ excludes or not the peak at DC because it would be canceled from $H_S(z)$ by the presence of the factor $\big[1-H_{\text{trend}}(z)\big]$. However, in practice because the filters are non-ideal, an extra step is usually taken to ensure that this peak is absent or minimized from s_n. For example, an additional de-trending step may be applied to $S(z)$, that is,

$$
\begin{aligned}
&S_{\text{prelim}}(z)= H_{\text{comb}}(z)R(z)\\
&S(z)= S_{\text{prelim}}(z)-H_{\text{trend}}(z)S_{\text{prelim}}(z)= H_{\text{comb}}(z)\big[1-H_{\text{trend}}(z)\big]^2 Y(z)
\end{aligned}
\tag{9.4.3}
$$

This results in the modified extraction filters, which still satisfy (9.4.2):

$$
\begin{aligned}
H_T(z) &= H_{\text{trend}}(z) && \text{(trend)}\\
H_S(z) &= H_{\text{comb}}(z)\big[1-H_{\text{trend}}(z)\big]^2 && \text{(seasonal)}\\
H_I(z) &= \big[1-H_{\text{trend}}(z)\big]\big\{1-H_{\text{comb}}(z)\big[1-H_{\text{trend}}(z)\big]\big\} && \text{(irregular)}
\end{aligned}
\tag{9.4.4}
$$

Further refinements will be discussed later on.

9.5 Classical Seasonal Decomposition

The classical seasonal decomposition method is the simplest realization of the procedure outlined in the previous section. Consider the following two possible lowpass trend-extraction filters:

$$
\begin{aligned}
H_{\text{trend}}(z) &= \frac{1}{D}\big[1+z^{-1}+z^{-2}+\cdots+z^{-(D-1)}\big]\\
H_{\text{trend}}(z) &= \frac{1}{D}\big[1+z^{-1}+z^{-2}+\cdots+z^{-(D-1)}\big]\cdot\frac{1}{2}(1+z^{-1})
\end{aligned}
\tag{9.5.1}
$$

where D is the period of the seasonal component. The first is typically used when D is odd, and the second, when D is even. They are referred to as the 1×D and 2×D trend filters, the notation $N_1{\times}N_2$ denoting the convolution of a length-N_1 with a length-N_2 averaging filter:

$$\frac{1}{N_1}\big[1+z^{-1}+\cdots+z^{-(N_1-1)}\big]\cdot\frac{1}{N_2}\big[1+z^{-1}+\cdots+z^{-(N_2-1)}\big] \tag{9.5.2}$$

The filters (9.5.1) are not perfect but are widely used. They have the desirable property of having nulls at the non-zero harmonics $\omega_k = k\omega_1 = 2\pi k/D$, $k = 1,2,\dots,D-1$. Their 3-dB cutoff frequency is about one-half the fundamental harmonic ω_1, that is,

$$\omega_c = 0.886\,\frac{\pi}{D} \tag{9.5.3}$$

Eq. (9.5.3) can easily be derived for the $1{\times}D$ case and is a good approximation for the $2{\times}D$ case. Fig. 9.5.1 shows the magnitude response $|H_{\text{trend}}(\omega)|$ versus ω over the symmetric Nyquist interval, $-\pi \le \omega \le \pi$. The 3-dB frequency is indicated on the graph at the $1/\sqrt{2}$ level.

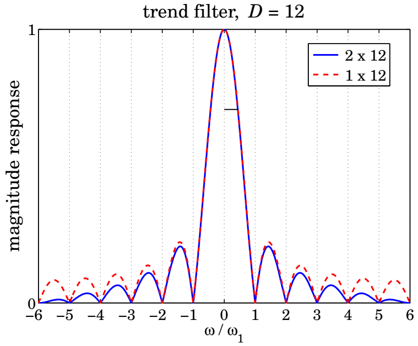


Fig. 9.5.1 Trend-extraction filters with $D = 12$.

In order to avoid delays introduced by the filters, the filters can be made symmetric with respect to the time origin. Let $D = 2p+1$ or $D = 2p$ in the even or odd case. Then, the symmetrized versions of the filters (9.5.1) are obtained by advancing them by p time units, that is, multiplying them by a factor of z^p:

$$\begin{aligned} D &= 2p+1\,, \quad H_{\text{trend}}(z) = z^p\frac{1}{D}\big[1+z^{-1}+\cdots+z^{-(D-1)}\big] \\ D &= 2p\,, \quad\quad\;\; H_{\text{trend}}(z) = z^p\frac{1}{D}\big[1+z^{-1}+\cdots+z^{-(D-1)}\big]\cdot\frac{1}{2}(1+z^{-1}) \end{aligned} \tag{9.5.4}$$

The corresponding frequency responses are obtained by setting $z = e^{j\omega}$:

$$\begin{aligned} D &= 2p+1\,, \quad H_{\text{trend}}(\omega) = \frac{\sin(D\omega/2)}{D\sin(\omega/2)} \\ D &= 2p\,, \quad\quad\;\; H_{\text{trend}}(\omega) = \frac{\sin(D\omega/2)}{D\sin(\omega/2)}\cdot\cos(\omega/2) \end{aligned} \tag{9.5.5}$$

where we used the identity $1+z^{-1}+\cdots+z^{-(D-1)} = (1-z^{-D})/(1-z^{-1})$. The symmetric impulse responses are:

$$\begin{aligned} D &= 2p+1\,, \quad \mathbf{h}_{\text{trend}} = \frac{1}{D}\big[1,\underbrace{1,\dots,1}_{2p-1\text{ ones}},1\big] \\ D &= 2p\,, \quad\quad\;\; \mathbf{h}_{\text{trend}} = \frac{1}{D}\big[0.5,\underbrace{1,\dots,1}_{2p-1\text{ ones}},0.5\big] \end{aligned} \tag{9.5.6}$$

In both cases, the filter length is $2p+1$, and the time-domain operation for calculating the estimated trend is by the symmetric convolutional equation:

$$\hat{t}_n = \sum_{i=-p}^{p} h_{\text{trend}}(i)\,y_{n-i} \tag{9.5.7}$$

The issues of filtering with double-sided filters were discussed in Sec. 3.9. We recall that for a length-L input signal y_n, the steady-state filtered output is over the time range $p \le n \le L-1-p$. The first p and last p output transients can be computed using appropriate asymmetric filters, and there exist many possibilities for these. Musgrave's minimum-revision method, discussed in Sec. 9.8, constructs such asymmetric filters from a given symmetric filter such as $\mathbf{h}_{\text{trend}}$.

The calculation of the trend estimate, incorporating also the end-point asymmetric filters, can be carried out with the MATLAB functions `trendma`, `minrev`, and `lpfilt`,

```
htrend = trendma(D);       % trend filters of Eq. (9.5.6)
B = minrev(htrend,R);      % corresponding smoothing matrix
t_hat = lpfilt(B,y);       % filtering operation
```

where `y` denotes the input data vector, and R is the Musgrave parameter to be explained in Sec. 9.8. The use of asymmetric filters affects only the first p and last p outputs.

In the so-called *classical decomposition method*, we apply the above filtering procedure to calculate the trend, and then apply ordinary signal averaging on the residual $r_n = y_n - t_n$ to calculate the seasonal component. The following computational steps describe the method:

```
B = minrev(trendma(D),R);   % trend moving-average, with minimum-revision end-filters
yt = lpfilt(B,y);           % trend component
yr = y - yt;                % seasonal + irregular components
ys = sigav(yr,D);           % seasonal component
yi = yr - ys;               % irregular component
```

For a multiplicative decomposition, $y_n = s_n t_n v_n$, the last three steps are replaced by,

```
yr = y ./ yt;               % seasonal + irregular components
ys = sigav(yr,D);           % seasonal component
yi = yr ./ ys;              % irregular component
```

The function `cldec` implements the above steps,

```
[yt,ys,yi] = cldec(y,D,R,type);     % classical decomposition method
```

where the string `type` takes on the values `'a'` or `'m'` for additive (the default) or multiplicative decomposition. The default value of R is zero, which simply omits the computation of the first and last p transients and replaces them with the corresponding samples of the input signal y_n.

Example 9.5.1: *Housing Starts.* Fig. 9.5.2 the trend and seasonal components of the housing starts data extracted by the classical decomposition method versus the methods discussed in Example 9.3.2.

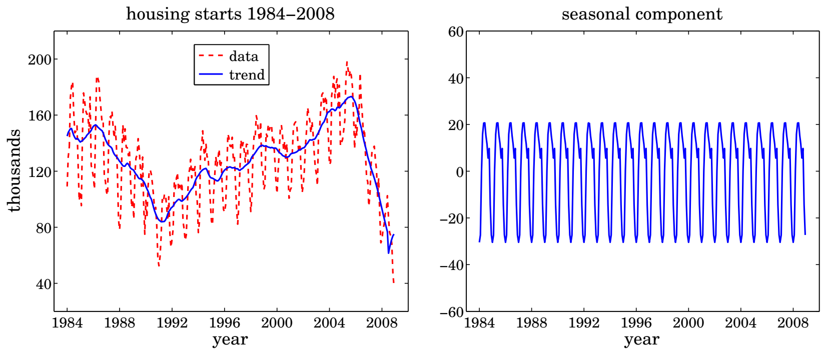


Fig. 9.5.2 Classical decomposition of monthly housing data.

The Musgrave parameter was chosen to be $R = 10$. Since $D = 12$, the value of R affects only the first and last 6 outputs. The MATLAB code for generating these graphs was,

```
Y = loadfile('newhouse.dat');
i = find(Y(:,1)==109.1);
y = Y(i:end-4,1); t = taxis(y,12,1984);

D=12; R=10;
[yt,ys,yi] = cldec(y,D,R);                 % classical decomposition method

figure; plot(t,y,'--', t,yt,'-');          % left graph
figure; plot(t,ys,'-');                    % right graph
```

The estimated trend is not as smooth as that of the Whittaker-Henderson method, but the estimated seasonal component is essentially the same as that of Example 9.3.2. □

Example 9.5.2: *Global Carbon Dioxide Data.* Figure 9.5.3 shows on the upper-left the monthly global CO$_2$ data for the period of January 1980 to March 2009, obtained from the NOAA web site: `http://www.esrl.noaa.gov/gmd/ccgg/trends/`.

The vertical axis is in parts per million (ppm), which represents the dry air mole fraction, that is, the number of CO$_2$ molecules divided by the number of all air molecules, after water vapor has been removed.

The upper graphs show the application of the classical seasonal decomposition method. The upper-left graph shows the trend t_n extracted by a 2×12 moving-average filter, while the right graph shows the seasonal component s_n.

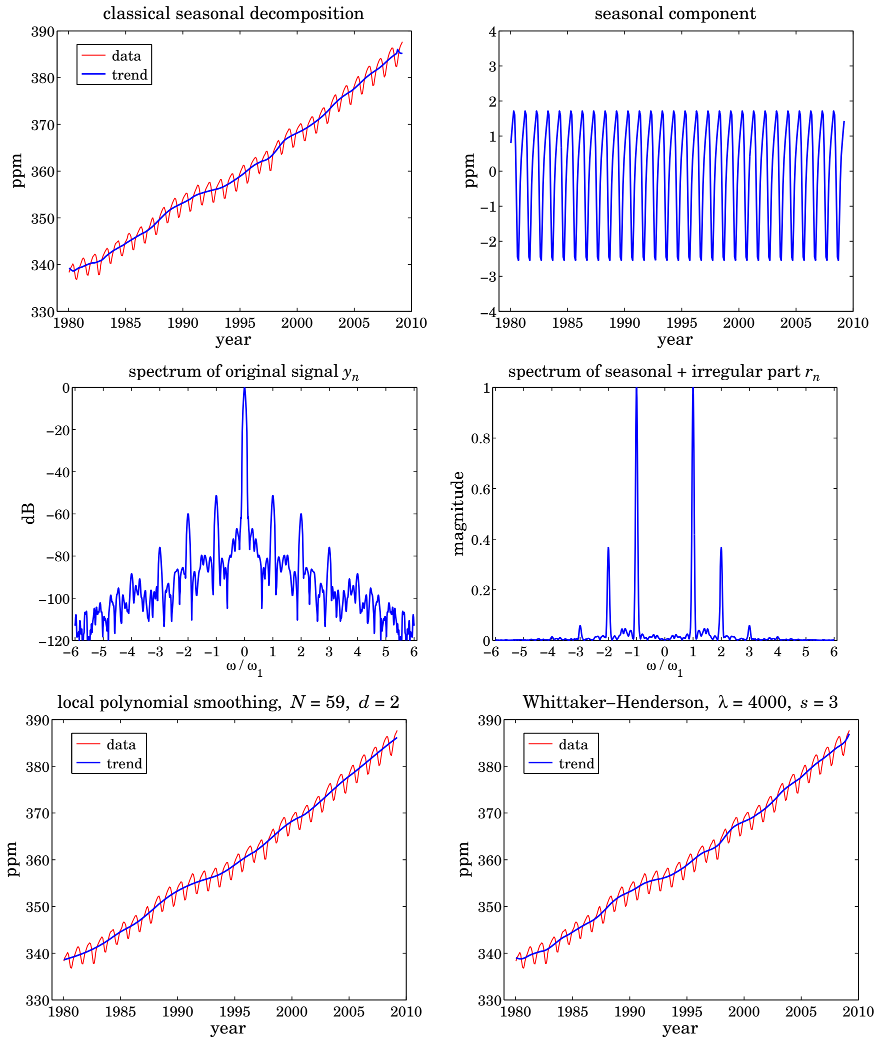


Fig. 9.5.3 Monthly global CO$_2$ data and spectra.

The middle graphs show the spectra of the original data on the left, and of the residual part $r_n = y_n - t_n = s_n + v_n$ on the right. The frequency axis is the symmetric Nyquist interval $[-\pi, \pi]$ in units of the fundamental harmonic $\omega_1 = 2\pi/12$. The trend dominates the spectrum of y_n and swamps the smaller harmonic peaks of the seasonal part. Indeed, the level of the seasonal component relative to the trend can be estimated in dB to be:

$$20\log_{10}\left(\frac{\text{std}(s_n)}{\text{mean}(y_n)}\right) = 20\log_{10}\left(\frac{1.46}{360}\right) = -47.8\text{ dB}$$

Therefore, the spectrum of the seasonal component is too small to be visible if plotted in absolute units. In order to make it visible, a Kaiser window with an 80-dB sidelobe level was applied to y_n prior to computing its spectrum and then plotted in dB. On the other hand, after the trend is removed, the harmonics in the residual component r_n are quite visible if plotted in absolute units as in the middle-right graph.

The bottom two graphs show the trend component t_n extracted by a local polynomial smoothing filter on the left (with length $N = 59$ and length $d = 2$), and by a Whittaker-Henderson smoother on the right (with $\lambda = 4000$ and $s = 3$). The corresponding seasonal components obtained by signal averaging of the residual $r_n = y_n - t_n$ are not shown because they are essentially the same as that of the upper-right graph. The MATLAB code used to generate these six graphs was as follows:

```
Y = loadfile('co2_mm_gl.dat');                   % data file in the OSP toolbox
t = Y(:,3); y = Y(:,4); yt0 = Y(:,5);            % extract times and signals

R = 15; [yt,ys,yi] = cldec(y,12,R);              % classical decomposition

figure; plot(t,y, t,yt);                         % upper-left graph
figure; plot(t,ys);                              upper-right graph

k = linspace(-6,6,1201); w = 2*pi*k/12;          % frequency in units of ω1

L = length(y); Rdb = 80;                         % Kaiser window parameters
wind = kwindow(L,Rdb)';                          % Kaiser window in the OSP toolbox

ysi = y - yt;                                    % seasonal + irregular component

Y =   abs(freqz(y.*wind,1,w));    Y = Y/max(Y);              % DTFT computation
Ysi = abs(freqz(ysi.*wind,1,w)); Ysi = Ysi/max(Ysi);

figure; plot(k, 20*log10(Y));                    % middle-left graph
figure; plot(k, Ysi);                            % middle-right graph

N=59; d=2; yt = lpfilt(lpsm(N,d),y);             % LPSM smoother

figure; plot(t,y, t,yt);                         % bottom-left graph
% ys = sigav(y-yt,12);                           % seasonal part, not shown
% figure; plot(t,ys);

la=4000; s=3; yt = whsm(y,la,s);                 % Whittaker-Henderson smoother

figure; plot(t,y, t,yt);                         % bottom-right graph
```

The signal yt0 extracted from the 5th column of the data file (as in the second line of code above) represents the already de-seasonalized data, and therefore, we can compare it to the trend extracted by the above three methods. It is not plotted because it is virtually identical to the above extracted trends.

The percentage error defined as `100*norm(yt-yt0)/norm(yt0)` is found to be 0.05%, 0.07%, and 0.05% for the classical, LPSM, and WH methods, respectively. □

To gain some further insight into the nature of the filtering operations for the classical decomposition method, we show in Fig. 9.5.4 the magnitude responses of the filters $H_S(\omega)$, $H_T(\omega)$, and $H_I(\omega)$ for extracting the seasonal, trend, and irregular components, as defined by Eq. (9.4.1). The trend filter $H_{\text{trend}}(\omega)$ is given by Eq. (9.5.5), and the comb filter $H_{\text{comb}}(\omega)$ by Eq. (9.3.5) with the phase factor removed to make it symmetric.

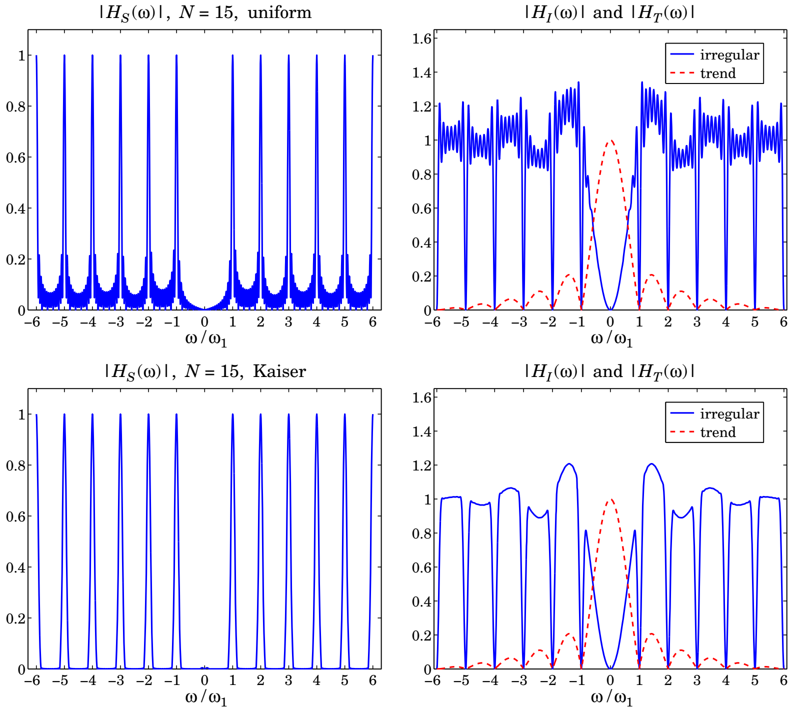


Fig. 9.5.4 Component extraction filters $H_S(\omega)$, $H_T(\omega)$, and $H_I(\omega)$.

The upper graphs in Fig. 9.5.4 show the case of $D = 12$ and $N = 15$. We observe the absence of the harmonic at DC in $H_S(\omega)$. The irregular filter does not quite extract the noise component v_n, but rather a filtered version thereof. Ideally, the irregular filter $H_I(\omega)$ should have zeros at the harmonics, be very small in the passband of $H_T(\omega)$, and be flat between the harmonics. The actual filter $H_I(\omega)$ does approximate these features.

The sidelobe behavior about the harmonics in $H_S(\omega)$, or about the nulls in $H_I(\omega)$, is due to the sidelobes introduced by the signal averaging filter $H_{\text{comb}}(\omega)$ of Eq. (9.3.5), which was obtained by applying the seasonalizing transformation $z \to z^D$ to a length-N FIR filter with uniform weights—the sidelobes being effectively the D-fold replicated versions of the sidelobes of a length-N rectangular window.

Such sidelobes are suppressed only by about 13 dB relative to the main peaks and are quire visible (at the level of $10^{-13/20} = 0.22$). The sidelobes can be suppressed further by replacing the rectangular FIR filter by a length-N windowed version thereof, using for example a Hamming or a Kaiser window. To be precise, the comb filter obtained from a window $w(n)$, $-M \le n \le M$, where $N = 2M + 1$, is defined by

$$W(z) = \sum_{n=-M}^{M} w(n)z^{-n} \quad \Rightarrow \quad H_{\text{comb}}(z) = W(z^D) = \sum_{n=-M}^{M} w(n)z^{-nD} \tag{9.5.8}$$

where $w(n)$ must be normalized to add up to unity. The two lower graphs of Fig. 9.5.4 show the filters obtained from a Kaiser window of length $N = 15$ and sidelobe level $R_{\text{dB}} = 50$ dB. The sidelobes are suppressed to the level of $10^{-50/20} = 0.003$ and are not visible if plotted in absolute scales. The price one pays for suppressing the sidelobes is, of course, the widening of the harmonic peaks. To clarify these ideas, we give below the MATLAB code for generating the graphs in Fig. 9.5.4:

```
D = 12; N = 15;
k = linspace(-6,6,1201); w = 2*pi*k/D;                        % frequency axis

ht = trendma(D);                   Ht = abs(freqz(ht,1,w));   % trend filter Htrend(ω)
hc = up(ones(1,N)/N, D);           Hc = abs(freqz(hc,1,w));   % comb filter Hcomb(ω)
hs = conv(hc, compl(ht));          Hs = abs(freqz(hs,1,w));   % seasonal filter HS(ω)
hi = conv(compl(hs), compl(ht));   Hi = abs(freqz(hi,1,w));   % irregular filter HI(ω)
ha = compl(hs);                    Ha = abs(freqz(ha,1,w));   % seasonal adjustment filter

figure; plot(k, Hs); figure; plot(k, Hi, k,Ht,'r--');         % upper graphs
figure; plot(k, Ha);                                          % left graph in Fig. 9.5.5

Rdb=50; hk = kwindow(N,Rdb); hk = hk/sum(hk);                 % Kaiser window

hc = up(hk, D);                    Hc = abs(freqz(hc,1,w));   % new Hcomb(ω)
hs = conv(hc, compl(ht));          Hs = abs(freqz(hs,1,w));   % new HS(ω)
hi = conv(compl(hs), compl(ht));   Hi = abs(freqz(hi,1,w));   % new HI(ω)
ha = compl(hs);                    Ha = abs(freqz(ha,1,w));   % seasonal adjustment filter

figure; plot(k, Hs); figure; plot(k, Hi, k,Ht,'r--');         % lower graphs
figure; plot(k, Ha);                                          % right graph in Fig. 9.5.5
```

The impulse response definitions in this code implement Eq. (9.4.1) in the time domain. The upsampling function `up` was described in Sec. 9.1. The function `compl` computes the impulse response of the complement of a double-sided symmetric filter, that is, $H(z) \rightarrow 1 - H(z)$, or $h_n \rightarrow \delta_n - h_n$. The function `kwindow` computes the Kaiser window (for spectral analysis) [604] for a given length N and sidelobe level R_{db} in dB, and it is part of the OSP toolbox.

Fig. 9.5.5 illustrates the complementarity property more clearly by showing the seasonal adjustment filter $H_A(\omega) = 1 - H_S(\omega) = H_T(\omega) + H_I(\omega)$, that is, the filter that removes the seasonal component from the data. As expected, the filter has nulls at the harmonics and is essentially flat in-between.

9.6 Seasonal Moving-Average Filters

Signal averaging can be thought of as ordinary filtering by the seasonalized FIR averager filter of Eq. (9.3.3). However, as we saw in Eq. (9.3.9), the averaged period builds up gradually at the filter output and becomes available only as the last D output points. This is so because the filter length ND is essentially the same as the signal length so that the filter operates mostly in its transient state. Indeed, if L is the length of the signal y_n, the number of periods is $N = \text{floor}(L/D)$ so that $L \approx ND$.

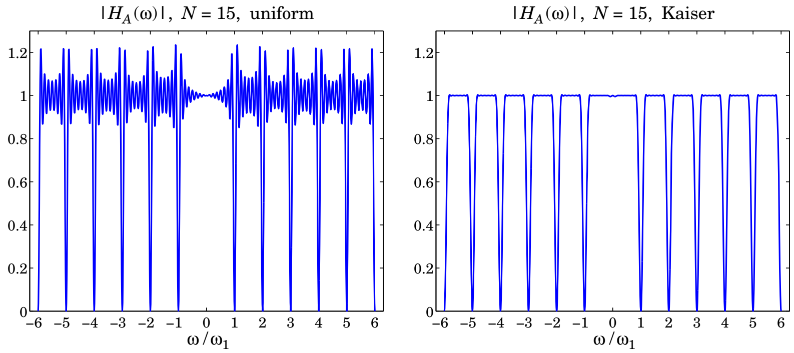


Fig. 9.5.5 Seasonal adjustment filter $H_A(\omega) = 1 - H_S(\omega) = H_T(\omega) + H_I(\omega)$.

In the classical decomposition method, the final accumulated period is replicated N times to make up the seasonal component s_n. This procedure is appropriate only if s_n is truly periodic. However, in many practical applications s_n is only quasi-periodic with slowly changing periods. In order to be able to estimate s_n more accurately we must use a shorter seasonal moving-average filter that tracks the local (i.e., within the filter's moving window) periodic component.

Example 9.6.1: Fig. 9.6.1 illustrates the filtering point of view for extracting the seasonal part s_n. The same CO$_2$ data are used as in Example 9.5.2. The classical decomposition method is applied first to determine the trend t_n, and then the residual signal is formed $r_n = y_n - t_n$. In this example, the number of periods contained in the y_n signal is $N = 29$.

The upper-left graph shows the result of ordinary causal filtering of the residual signal r_n by the signal averaging comb filter (9.3.3) using MATLAB's built-in function `filter`. We observe that the transients eventually build up to the same final period as that obtained by signal averaging (shown as the dotted line.)

In the upper-right graph, the residual r_n was filtered by the double-sided filtering function `filtdbl` discussed in Sec. 3.9, which is ordinary causal convolution followed by advancing the result by $(N-1)D/2$ samples. Again, we observe the input-on and input-off transients and the build-up of the correct period at the middle.

The transient portions of the double-sided filter output can be adjusted by using Musgrave's minimum-revision asymmetric filters for the left and right end points. The resulting filter output is shown in the lower-left graph, in which the Musgrave parameter was chosen to be $R = \infty$ (see Sec. 9.8 for more on that.)

The lower-right graph shows the result of filtering r_n through a so-called 3×3 double-sided seasonal moving-average filter, which is discussed below. The MATLAB code for generating these graphs is as follows:

```
Y = loadfile('co2_mm_gl.dat');
t = Y(:,3); y = Y(:,4);                           % CO2 data
```

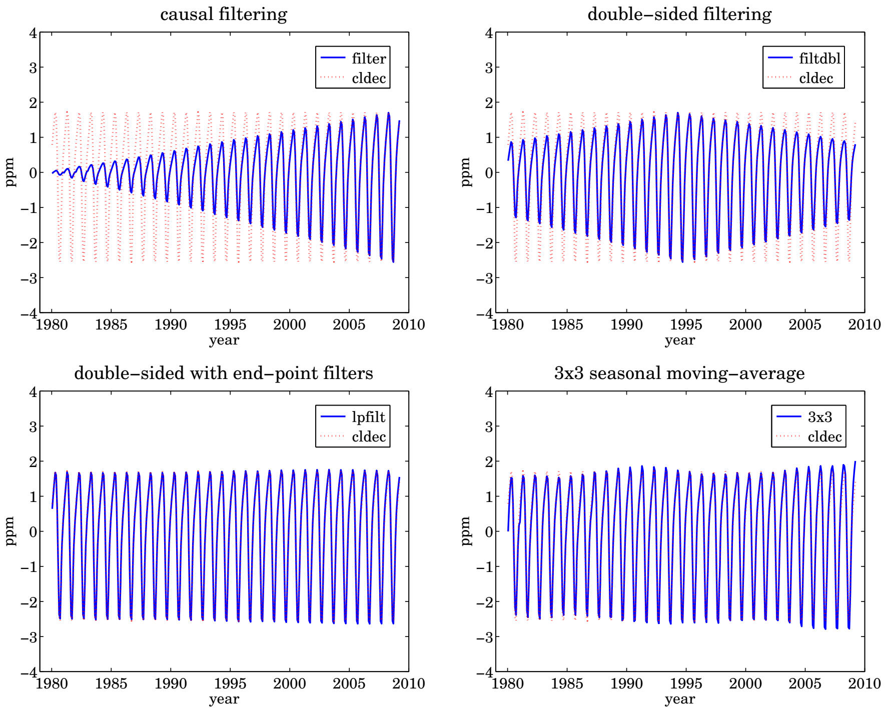


Fig. 9.6.1 Filtering versions of seasonal filter.

```
D=12; N=floor(length(y)/D); M=33; R=inf;      % filter parameters

[yt,ys,yi] = cldec(y,D,R); yr = y - yt;        % yr = residual component r_n

h = ones(1,N)/N;                               % length-N moving-average
hc = up(h, D);                                 % seasonalized comb filter obtained from h
Bc = upmat(minrev(h,R), D);                    % seasonalized minimum-revision filter matrix

ys1 = filter(hc,1,yr);                         % ordinary causal filtering by the comb filter hc
ys2 = filtdbl(hc,yr);                          % double-sided filtering
ys3 = lpfilt(Bc, yr);                          % double-sided filtering and end-point filters
[yt4,ys4] = smadec(y, D, M, R);                % ys4 is the 3×3 moving-average output

figure; plot(t,ys1, t,ys,':'); figure; plot(t,ys2, t,ys,':');     % upper graphs
figure; plot(t,ys3, t,ys,':'); figure; plot(t,ys4, t,ys,':');     % lower graphs
```

The `smadec` function is a simple alternative to `cldec` and is discussed below. The function `upmat` upsamples a filter matrix by a factor of D for its use in comb filtering. It upsamples each row and then each column by D and then, it replaces each group of D columns by the corresponding convolution matrix arising from the first column in each group. It can be passed directly into the filtering function `lpfilt`,

```
Bup = upmat(B,D;          % upsampling a filtering matrix
```

For example, the asymmetric filters associated with the 3×3 seasonal moving-average filter [618] are as follows for $D = 3$, where the middle column is the 3×3 filter and the other columns, the asymmetric filters to be used at the ends of the data record, and the function `smat` is described below:

$$B = \text{smat}(1,33) = \frac{1}{27}\begin{bmatrix} 11 & 7 & 3 & 0 & 0 \\ 11 & 10 & 6 & 3 & 0 \\ 5 & 7 & 9 & 7 & 5 \\ 0 & 3 & 6 & 10 & 11 \\ 0 & 0 & 3 & 7 & 11 \end{bmatrix}$$

$$B_{\text{up}} = \text{upmat}(B,3) = \frac{1}{27}\begin{bmatrix} 11 & 0 & 0 & 7 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 11 & 0 & 0 & 7 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 11 & 0 & 0 & 7 & 0 & 0 & 3 & 0 & 0 & 0 & 0 \\ 11 & 0 & 0 & 10 & 0 & 0 & 6 & 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 11 & 0 & 0 & 10 & 0 & 0 & 7 & 0 & 0 & 5 & 0 & 0 \\ 0 & 0 & 11 & 0 & 0 & 10 & 0 & 0 & 7 & 0 & 0 & 5 & 0 \\ 5 & 0 & 0 & 7 & 0 & 0 & 9 & 0 & 0 & 7 & 0 & 0 & 5 \\ 0 & 5 & 0 & 0 & 7 & 0 & 0 & 10 & 0 & 0 & 11 & 0 & 0 \\ 0 & 0 & 5 & 0 & 0 & 7 & 0 & 0 & 10 & 0 & 0 & 11 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 & 6 & 0 & 0 & 10 & 0 & 0 & 11 \\ 0 & 0 & 0 & 0 & 3 & 0 & 0 & 7 & 0 & 0 & 11 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 7 & 0 & 0 & 11 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 7 & 0 & 0 & 11 \end{bmatrix} \tag{9.6.1}$$

The simple 3×3, 3×5, and 3×9 seasonal moving-average filters are widely used in de-seasonalizing business, government, and census data. They are obtained by symmetrizing the $N_1 \times N_2$ filters of Eq. (9.5.2) and then applying the transformation $z \to z^D$. For example, the resulting 3×3 and 3×5 comb filters are:

$$\begin{aligned} H_{33}(z) &= \frac{1}{3}(z^D + 1 + z^{-D})\cdot\frac{1}{3}(z^D + 1 + z^{-D}) \\ H_{35}(z) &= \frac{1}{3}(z^D + 1 + z^{-D})\cdot\frac{1}{5}(z^{2D} + z^D + 1 + z^{-D} + z^{-2D}) \end{aligned} \tag{9.6.2}$$

with symmetric impulse responses,

$$\begin{aligned} \mathbf{h}_{33} &= \frac{1}{9}[1, \underbrace{0,\dots,0}_{D-1\text{ zeros}}, 2, 0,\dots,0, 3, 0,\dots,0, 2, 0,\dots,0, 1] \\ \mathbf{h}_{35} &= \frac{1}{15}[1, 2, 0,\dots,0, 3, 0,\dots,0, 3, 0,\dots,0, 3, 0,\dots,0, 2, 0,\dots,0, 1] \end{aligned} \tag{9.6.3}$$

The MATLAB function `smav` calculates such impulse responses,

```
h = smav(N1,N2,D);          % seasonal moving-average filters
```

It is simply:

```
h  = up(conv(ones(1,N1), ones(1,N2))/(N1*N2), D);
```

These filters are to be applied to the residual signal $r_n = y_n - t_n$. Their end-point effects can be handled by using Musgrave's minimum-revision filters or by any other appropriate asymmetric filters. In fact, the census X-11/X-12 methods use asymmetric filters that are specially constructed for the 3×3, 3×5, and 3×9 filters, and may be found in Ref. [618]. They have been incorporated into the `smadec` and `x11dec`. For example, Eq. (9.6.1) shows the 3×3 filter matrix before and after it is upsampled.

To summarize, the filtering approach for de-seasonalizing a signal $y_n = s_n + t_n + v_n$ with period D consists of the following two basic steps:

1. Apply a lowpass filter to extract the trend component t_n, incorporating also asymmetric end-point filters. The trend-extraction filter can be a simple $1\times D$ or $2\times D$ moving average, or, any other lowpass filter such as a local-polynomial or Whittaker-Henderson smoother.
2. Apply a comb filter to the de-trended residual signal $r_n = y_n - t_n$ to extract the seasonal part s_n, incorporating asymmetric filters for the end-points. The comb filter can be a simple seasonalized 3×3, 3×5, or 3×9 lowpass filter, or a more general seasonalized filter such as one obtained from a non-rectangular window. The de-seasonalized, or seasonally adjusted, signal is then $a_n = y_n - s_n$.

The MATLAB function `smadec` carries out this program using the simple $1\times D$ or $2\times D$ moving-average filter for de-trending and the 3×3, 3×5, or 3×9 comb filters for the seasonal part. It has usage:

```
[yt,ys,yi] = smadec(y,D,M,R,iter,type);      % seasonal moving-average decomposition
```

where `yt, ys, yi` are the estimated components t_n, s_n, v_n, and `y` is the input data vector. The integer values $M = 33, 35, 39$ select the 3×3, 3×5, or 3×9 seasonal comb filters, other values of M can also be used. The Musgrave parameter defaults to $R = \infty$, the parameter `iter` specifies the number of iterations of the filtering process, which correspond to applying the trend filter `iter` times. The string `type` takes on the values `'a', 'm'` for additive or multiplicative decomposition. To clarify the operations, we give below the essential part of the code in `smadec` for the additive case:

```
F = minrev(trendma(D),R);          % trend moving-average, with minimum-revision end-filters
B = smat(D,M,R);                   % seasonal moving averages, with end-filters

yt = y;                            % initialize iteration
for i=1:iter,
  yt = lpfilt(F,yt);               % T component
  yr = y - yt;                     % S+I component
  ys = lpfilt(B,yr);               % S component
  yi = yr - ys;                    % I component
end
```

The function `smat` generates the filtering matrix of the seasonalized comb filters, including the specific asymmetric filters for the 3×3, 3×5, or 3×9 cases, as well for other cases.

Example 9.6.2: *Unemployment Data 1965–1979.* The data set representing the monthly number of unemployed 16–19 year old men for the period Jan. 1965 to Dec. 1979 has served as a benchmark for comparing seasonal adjustment methods [633,637]. The data set is available from the US Bureau of Labor Statistics web site: `http://www.bls.gov/data/` (series ID: `LNU03000013`, under category: Unemployment > Labor Force Statistics > on-screen data search).

The upper graphs of Fig. 9.6.2 illustrate the application of the `smadec` function using $D = 12$, $M = 35$ (which selects the 3×5 comb), one iteration, Musgrave parameter $R = \infty$ for the $2\times D$ trend filter, and additive decomposition type. The left graph shows the trend t_n and the right, the estimated seasonal component s_n, which is not exactly periodic but exhibits quasi-periodicity. The results are comparable to those of Refs. [633,637].

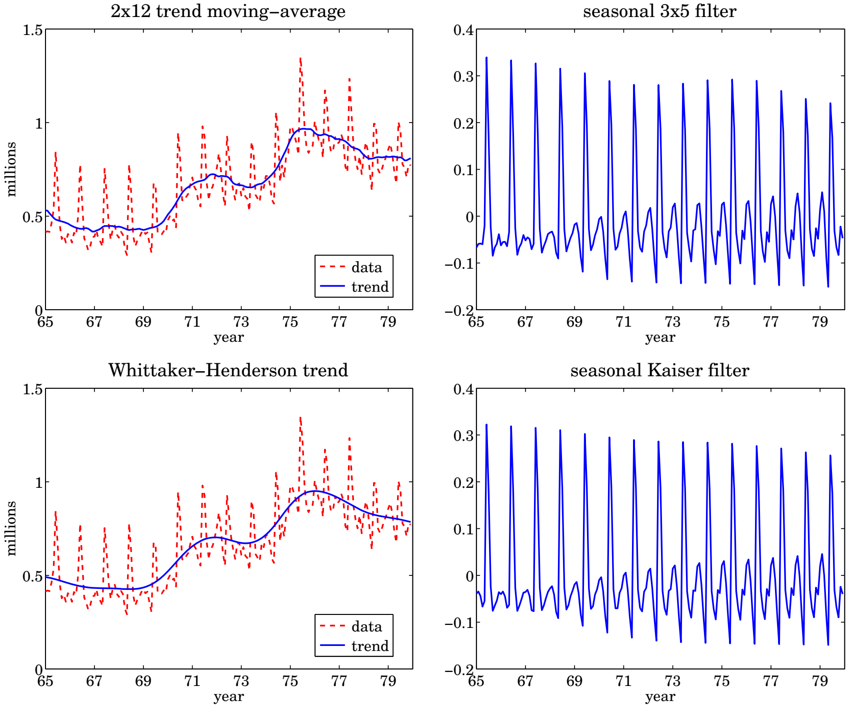


Fig. 9.6.2 Trend/seasonal decomposition of monthly unemployment data for 1965–1979.

The lower graphs show the decomposition obtained by de-trending using a Whittaker-Henderson smoother of order $s = 2$, followed by a Kaiser comb filter. The function `whgcv` was used to determine the optimum smoothing parameter, $\lambda_{\text{opt}} = 2039$. The Kaiser window had length $N = 15$ and relative sidelobe level of $R_{\text{db}} = 50$ dB. The MATLAB code for generating these graphs was as follows:

```
Y = loadfile('unemp-1619-nsa.dat');                  % data file in OSP toolbox
i=find(Y==1965); Y = Y(i:i+14,2:13)';                % extract 1965-1979 data
```

```
y = Y(:)/1000; t = taxis(y,12,65);                   % y units in millions

D=12; M=35; R=inf; iter=1; type='a';                 % smadec input parameters

[yt,ys,yi] = smadec(y,D,M,R,iter,type);              % yt,ys represent tn, sn

figure; plot(t,y, t,yt); figure; plot(t,ys);         % upper graphs

s = 2; la = 2000:2050;                               % search range of λ's
[gcv,lopt] = whgcv(y,la,s);                          % optimum λopt = 2039

yt = whsm(y,lopt,s);                                 % extract tn component
yr = y-yt;                                           % residual S+I component

Rdb=50; N=15; h = kwindow(N,Rdb); hk = h/sum(h);     % Kaiser window
B = upmat(minrev(hk,R), D);                          % Kaiser comb with end-filters

ys = lpfilt(B, yr);                                  % extract sn component

figure; plot(t,y, t,yt); figure; plot(t,ys);         % bottom graphs
```

The Whittaker-Henderson method results in a smoother trend. However, the trend from `smadec` can be made equally smooth by increasing the number of iterations, for example, setting `iter=3`. The frequency responses of the various filters are shown in Eq. (9.6.3).

The filters $H_T(\omega), H_S(\omega)$ for extracting the trend and seasonal components, and the seasonal-adjustment filter $H_A(\omega) = 1 - H_S(\omega)$ are constructed from Eq. (9.4.1). The MATLAB code for generating these graphs was:

```
k = linspace(-6,6,1201); w = 2*pi*k/12;                 % frequency range [−π, π]

ht = trendma(12);           Ht = abs(freqz(ht,1,w));    % 2×12 trend filter
hc = smav(3,5,12);                                      % upsampled 3×5 comb filter
hs = conv(hc,compl(ht));    Hs = abs(freqz(hs,1,w));    % seasonal filter, HS(ω)
ha = compl(hs);             Ha = abs(freqz(ha,1,w));    % adjustment filter, 1 − HS(ω)

figure; plot(k,Hs, k,Ht,'--'); figure; plot(k,Ha);      % upper graphs

Ht = 1 ./ (1 + lopt * (2*sin(w/2)).^(2*s));             % Whittaker-Henderson trend filter
hc = up(hk,12);                                         % Kaiser comb impulse response
Hc = freqz(hc,1,w) .* exp(j*(N-1)*D*w/2);               % Kaiser comb frequency response
Hs = Hc .* (1-Ht); Ha = 1 - Hs;                         % HS(ω) and HA(ω) = 1 − HS(ω)
Hs = abs(Hs); Ha = abs(Ha);

figure; plot(k,Hs, k,Ht,'--'); figure; plot(k,Ha);      % bottom graphs
```

The Whittaker-Henderson trend filter was computed using Eq. (8.2.7). The frequency response of the Kaiser comb filter was multiplied by $e^{j\omega(N-1)D/2}$ to make the filter symmetric. It is evident that the WH/Kaiser filters perform better. □

The steps implementing the Whittaker-Henderson/Kaiser decomposition have been incorporated into the MATLAB function `whkdec`,

```
[yt,ys,yi] = whkdec(y,D,s,la,N,Rdb,R,type);     % WH/Kaiser decomposition
```

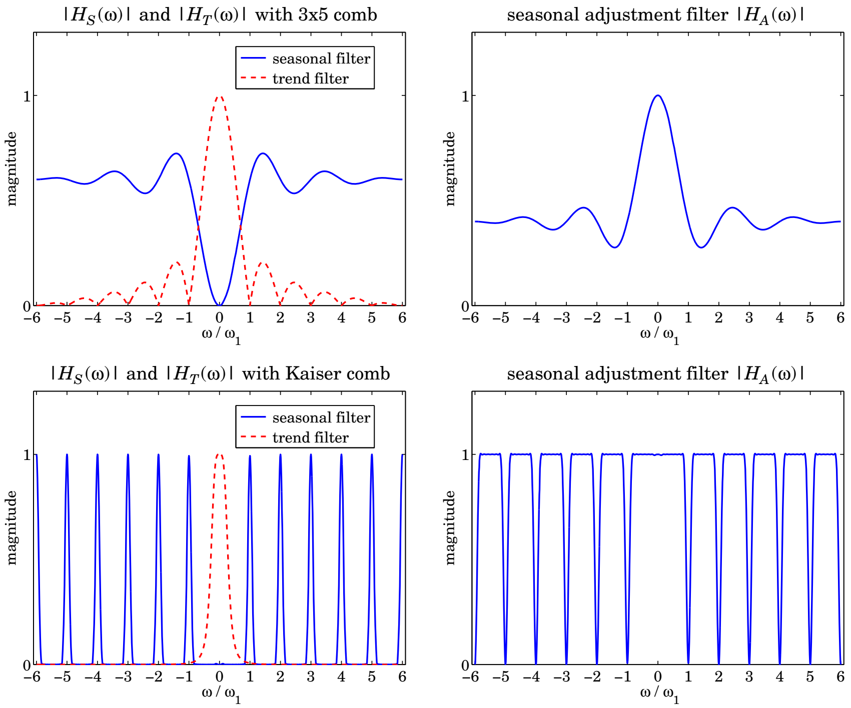


Fig. 9.6.3 Frequency responses of trend, seasonal, and seasonal-adjustment filters.

The WH parameters `s,la` must be selected in advance, for example, λ can be tentatively estimated using the GCV function `whgcv`, but it should be noted that the GCV does not always give a "good" value for λ. The Kaiser window length `N` must be odd and the sidelobe level must be restricted to the range $[13, 120]$ dB. The Musgrave parameter R affects only the Kaiser comb filter because the WH trend already takes into account the end points. The parameter `type` is as in the function `smadec`.

9.7 Census X-11 Decomposition Filters

The Census X-11/X-12 seasonal adjustment procedures have become a standard for deseasonalizing economic data [605–621]. They are based on a series of filtering operations that represent a refined version of the procedures outlined in the previous section.

Here, we only discuss the relevant filtering operations, leaving out details such as adjustments for outliers or calendar effects. The most recent version, X-12-ARIMA, is available from the web site [607]. The web pages [608,609] contain a number of papers on the development of the X-11/X-12 methods.

As outlined in [610,613], the X-11 method involves the repeated application of the 2×12 trend filter of Eq. (9.5.1), the 3×3 and 3×5 comb filters of Eq. (9.6.2), and the Henderson filters of lengths 9, 13, or 25, with polynomial and smoothing orders $d = s = 3$ given by Eq. (4.2.29) of Chap. 4. The basic X-11 filtering steps are as follows, assuming an additive model $y_n = s_n + t_n + v_n$,

1. Apply a 2×12 trend filter to y_n to get a preliminary estimate of the trend t_n.
2. Subtract t_n from y_n to get a preliminary estimate of the residual $r_n = y_n - t_n$.
3. Apply the 3×3 comb filter to r_n to get a preliminary estimate of s_n.
4. Get an improved s_n by removing its filtered version by the 2×12 trend filter.
5. Subtract s_n from y_n to get a preliminary adjusted signal $a_n = y_n - s_n = t_n + v_n$.
6. Filter a_n by a Henderson filter to get an improved estimate of the trend t_n.
7. Subtract t_n from y_n to get an improved residual $r_n = y_n - t_n$.
8. Apply the 3×5 comb filter to r_n to get an improved estimate of s_n.
9. Get the final s_n by removing its filtered version by the 2×12 trend filter.
10. Subtract s_n from y_n to get the final adjusted signal $a_n = y_n - s_n = t_n + v_n$.
11. Filter a_n by a Henderson filter to get the final estimate of the trend t_n.
12. Subtract t_n from a_n to get the final estimate of the irregular component v_n.

These steps are for monthly data. For quarterly data, replace the 2×12 trend filter by a 2×4 filter. The steps can be expressed concisely in the z-domain as follows:

$$\begin{aligned}
&1.\quad Y_T^{\text{pre}} = FY\\
&2.\quad Y_R^{\text{pre}} = Y - Y_T^{\text{pre}} = (1-F)Y\\
&3.\quad Y_S^{\text{pre}} = H_{33}Y_R^{\text{pre}} = H_{33}(1-F)Y\\
&4.\quad Y_S^{\text{imp}} = Y_S^{\text{pre}} - FY_S^{\text{pre}} = H_{33}(1-F)^2Y\\
&5.\quad Y_A^{\text{pre}} = Y - Y_S^{\text{imp}} = \big[1 - H_{33}(1-F)^2\big]Y\\
&6.\quad Y_T^{\text{imp}} = HY_A^{\text{pre}} = H\big[1 - H_{33}(1-F)^2\big]Y\\
&7.\quad Y_R^{\text{imp}} = Y - Y_T^{\text{imp}} = \Big[1 - H\big[1 - H_{33}(1-F)^2\big]\Big]Y\\
&8.\quad Y_S^{\text{imp}} = H_{35}Y_R^{\text{imp}} = H_{35}\Big[1 - H\big[1 - H_{33}(1-F)^2\big]\Big]Y\\
&9.\quad Y_S = Y_S^{\text{imp}} - FY_S^{\text{imp}} = (1-F)H_{35}\Big[1 - H\big[1 - H_{33}(1-F)^2\big]\Big]Y \equiv H_SY\\
&10.\quad Y_A = Y - Y_S = (1-H_S)Y \equiv H_AY\\
&11.\quad Y_T = HY_A = H(1-H_S)Y \equiv H_TY\\
&12.\quad Y_I = Y_A - Y_T = (1-H)(1-H_S)Y \equiv H_IY
\end{aligned} \tag{9.7.1}$$

where $Y_T^{\text{pre}} = FY$ stands for $Y_Y^{\text{pre}}(z) = F(z)Y(z)$, etc., and $F(z)$, $H_{33}(z)$, $H_{35}(z)$, $H(z)$ denote the 2×12 trend filter, the 3×3 and 3×5 comb filters, and the Henderson filter, and the z-transforms of the data, trend, seasonal, adjusted, and irregular components are denoted by $Y(z)$, $Y_T(z)$, $Y_S(z)$, $Y_A(z)$, and $Y_I(z)$.

It follows that the effective filters for extracting the seasonal, seasonally-adjusted, trend, and irregular components are:

$$\begin{aligned}
H_S &= (1-F)H_{35}\Big[1 - H\big[1 - H_{33}(1-F)^2\big]\Big] &&\text{(seasonal)}\\
H_A &= 1 - H_S &&\text{(seasonally-adjusted)}\\
H_T &= H(1-H_S) &&\text{(trend)}\\
H_I &= (1-H)(1-H_S) &&\text{(irregular)}
\end{aligned} \tag{9.7.2}$$

They satisfy the complementarity property $H_T(z) + H_S(z) + H_I(z) = 1$.

Example 9.7.1: *X-11 Filters.* The construction of the time-domain impulse responses of the X-11 decomposition filters (9.7.2) is straightforward. For example, the following MATLAB code evaluates the impulse responses (using a 13-term Henderson filter), as well as the corresponding frequency responses shown in Fig. 9.7.1,

```
k = linspace(-6,6,1201); w = 2*pi*k/12;     % frequency axis −π ≤ ω ≤ π

hf  = trendma(12);                          % 2×12 trend filter, F
hfc = compl(hf);                            % complement of trend filter, 1 − F
h33 = smav(3,3,12);                         % 3×3 comb filter, H33
h35 = smav(3,5,12);                         % 3×5 comb filter, H35
N=13; he = lprs2(N,3,3);                    % 13-term Henderson filter, H
g = conv(hfc,hfc);                          % G = (1 − F)^2, G is temporary variable
g = compl(conv(h33, g));                    % G = 1 − H33 (1 − F)^2
g = compl(conv(he,g));                      % G = 1 − H[1 − H33 (1 − F)^2]
g = conv(h35,g);                            % G = H35{1 − H[1 − H33 (1 − F)^2]}
                                            % HS = (1 − F)H35{1 − H[1 − H33 (1 − F)^2]}
hs = conv(hfc,g);          Hs = abs(freqz(hs,1,w));    % seasonal
ha = compl(hs);            Ha = abs(freqz(ha,1,w));    % adjustment
ht = conv(he,ha);          Ht = abs(freqz(ht,1,w));    % trend
hi = conv(compl(he),ha);   Hi = abs(freqz(hi,1,w));    % irregular

figure; plot(k, Hs); figure; plot(k, Ht);              % upper graphs
figure; plot(k, Ha); figure; plot(k, Hi);              % lower graphs
```

We note that the filters have the expected shapes. All cases described in [613] can be generated by variations of this code. □

The MATLAB function `x11dec` implements the above steps, amended by the use of asymmetric filters to handle the end-points of the time series,

```
[yt,ys,yi] = x11dec(y,D,M1,M2,N1,N2,R,type);      % X-11 decomposition method
```

where `D` is the seasonal period, `M1,M2` the sizes of the first and second comb filters (entered as 33, 35, or 39), `N1,N2` are the lengths of the first and second Henderson filters, `R` is the Musgrave minimum-revision parameter affecting both the Henderson filters and the trend filter, and `type` designates an additive or multiplicative decomposition. The Musgrave parameter R is usually assigned the following values, depending on the length

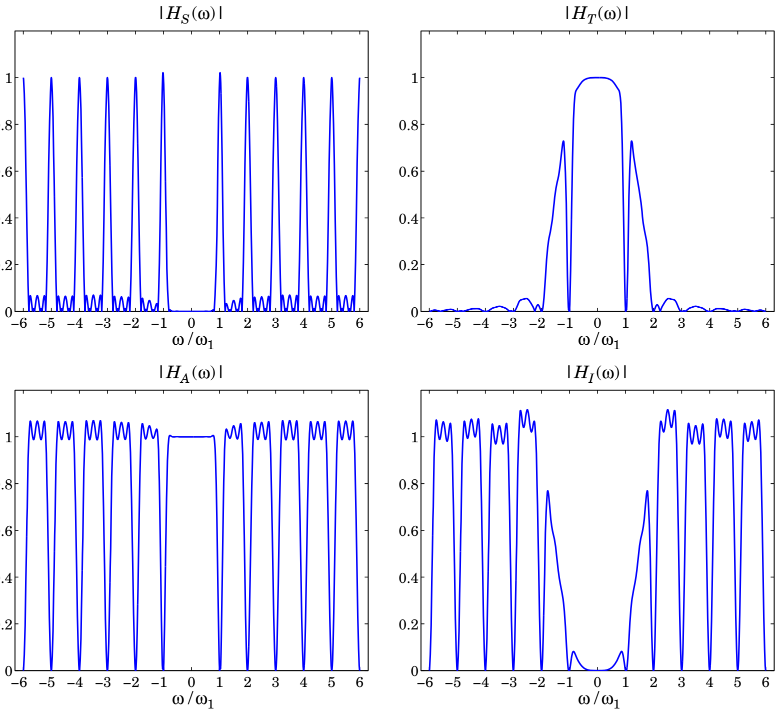


Fig. 9.7.1 X-11 decomposition filters (with 13-term Henderson).

N of the Henderson filter [618]:

N	R
5	0.001
7	4.5
9	1.0
13	3.5
23	4.5

(9.7.3)

Example 9.7.2: *Unemployment Data 1980–2008.* Fig. 9.7.2 shows the X-11 decomposition of the monthly unemployment data for 20 year and older men for the period Jan. 1980 to Dec. 2008. The data are from the US BLS web site: `http://www.bls.gov/data/`, series `LNU03000025`, under category: Unemployment > Labor Force Statistics > on-screen data search. The already seasonally adjusted data are also available as series `LNS13000025`.

The upper-left graph shows the original data and the extracted trend t_n assuming an additive model $y_n = t_n + s_n + v_n$. The lower-left graph is the extracted seasonal component s_n. The upper-right graph shows the seasonally-adjusted signal $a_n = y_n - s_n = t_n + v_n$ to be compared with that of the lower-right graph, which shows the already available adjusted signal— the two agreeing fairly well. The graphs were generated by the following code:

```
Y = loadfile('unemp-20-nsa.dat');            % not-seasonally adjusted data
```

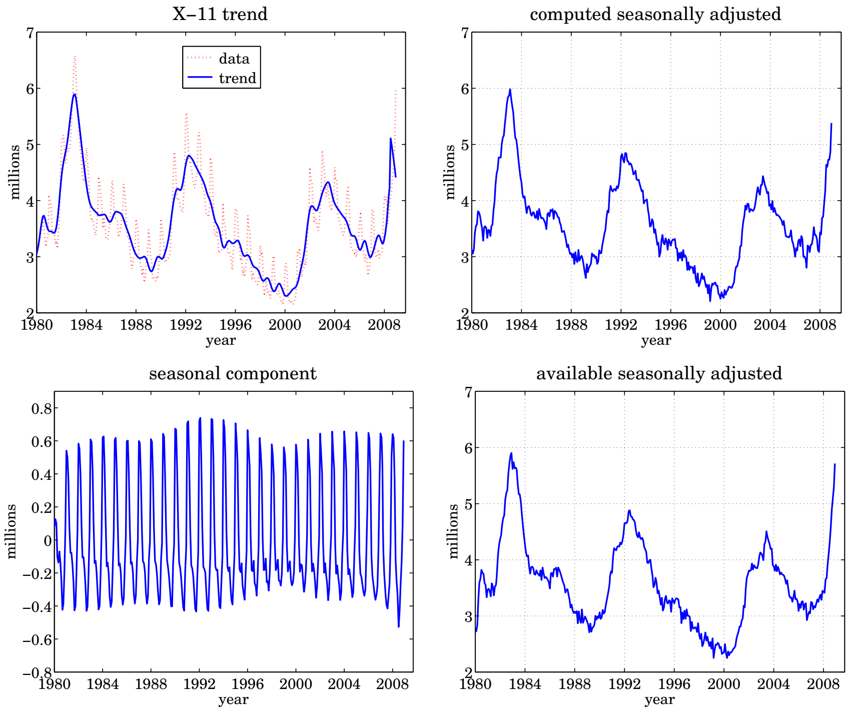


Fig. 9.7.2 X-11 decomposition of unemployment data 1980-2008.

```
i = find(Y==1980); Y = Y(i:end,2:13)';          % select years 1980-2008
y = Y(:)/1000; t = taxis(y,12,1980);            % data vector y, and time axis
                                                % data sets available in the OSP toolbox
Y = loadfile('unemp-20-sa.dat');                % seasonally adjusted data
i = find(Y==1980); Y = Y(i:end,2:13)';
yadj = Y(:)/1000;                               % yadj = already available adjusted data

D=12; M1=33; M2=35; N1=13; N2=13; R=3.5; type='a';     % X-11 parameters
[yt,ys,yi] = x11dec(y,D,M1,M2,N1,N2,R,type);           % X-11 method
ya = y-ys;                                             % seasonally adjusted

%s = 2; la = 1000; N=15; Rdb=50;                      % WH/K parameters
%[yt,ys,yi] = whkdec(y,D,s,la,N,Rdb,R,type);          % Whittaker-Henderson/Kaiser
%ya = y-ys;                                           % seasonally adjusted

figure; plot(t,y,':', t,yt,'-'); figure; plot(t,ya);   % upper graphs
figure; plot(t,ys); figure; plot(t,yadj);              % lower graphs
```

The value of R was 3.5 because a 13-term Henderson filter was used. The purpose of this example was to compare the performance of our simplified X-11 implementation with the results that are already available from the Bureau of Labor Statistics. We note that the use of the Whittaker-Henderson/Kaiser decomposition method also works comparably well, for example with parameters $s = 2$, $\lambda = 1000$, Kaiser length $N = 15$, and $R_{\text{db}} = 50$ dB. The code for that is included above but it is commented out. □

9.8 Musgrave Asymmetric Filters

The handling of the end-point problem by the use of asymmetric filters was discussed in Sec. 3.9. We saw that the output y_n of filtering a length-L signal x_n, $0 \le n \le L-1$, by a double-sided filter h_m, $-M \le m \le M$, using for example the function `filtdbl`, consists of M initial and M final transient output samples, and $L - 2M$ steady-state samples, the latter being computed by the steady-state version of the convolutional equation:

$$y_n = \sum_{m=-M}^{M} h_m x_{n-m}\,, \quad M \le n \le L-1-M \tag{9.8.1}$$

The overall operation can be cast in convolution matrix form. For example, for $L = 8$ and $M = 2$, we have:

$$\begin{bmatrix} y_0 \\ y_1 \\ \cdots \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ \cdots \\ y_6 \\ y_7 \end{bmatrix} = \begin{bmatrix} h_0 & h_{-1} & h_{-2} & 0 & 0 & 0 & 0 & 0 \\ h_1 & h_0 & h_{-1} & h_{-2} & 0 & 0 & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ h_2 & h_1 & h_0 & h_{-1} & h_{-2} & 0 & 0 & 0 \\ 0 & h_2 & h_1 & h_0 & h_{-1} & h_{-2} & 0 & 0 \\ 0 & 0 & h_2 & h_1 & h_0 & h_{-1} & h_{-2} & 0 \\ 0 & 0 & 0 & h_2 & h_1 & h_0 & h_{-1} & h_{-2} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 0 & h_2 & h_1 & h_0 & h_{-1} \\ 0 & 0 & 0 & 0 & 0 & h_2 & h_1 & h_0 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \cdots \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ \cdots \\ x_6 \\ x_7 \end{bmatrix} \tag{9.8.2}$$

The middle $L - 2M = 4$ output samples are steady, while the first and last $M = 2$ are transient and are computed by using fewer filter weights that the steady ones. The transient and steady filters can be arranged into a matrix B, which is for the above example,

$$B = \begin{bmatrix} h_0 & h_1 & h_2 & 0 & 0 \\ h_{-1} & h_0 & h_1 & h_2 & 0 \\ h_{-2} & h_{-1} & h_0 & h_1 & h_2 \\ 0 & h_{-2} & h_{-1} & h_0 & h_1 \\ 0 & 0 & h_{-2} & h_{-1} & h_0 \end{bmatrix} \tag{9.8.3}$$

The convolution matrix H can be built from the knowledge of B as described in Sec. 3.9. The matrix B conveniently summarizes the relevant filters and can be used as an input to the filtering function `lpfilt`.

As discussed in Sec. 3.9, local polynomial smoothing filters, including Henderson filters, generate their own matrix B to handle the series end-points, with the non-central columns of B consisting of the corresponding prediction filters.

However, when one does not have available such prediction filters, but only the central filter h_m, $-M \le m \le M$, one must use appropriately designed end-point filters. For example, Eq. (9.8.2) would change to:

$$\begin{bmatrix} y_0 \\ y_1 \\ \cdots \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ \cdots \\ y_6 \\ y_7 \end{bmatrix} = \begin{bmatrix} f^0_0 & f^0_{-1} & f^0_{-2} & 0 & 0 & 0 & 0 & 0 \\ f^1_1 & f^1_0 & f^1_{-1} & f^1_{-2} & 0 & 0 & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ h_2 & h_1 & h_0 & h_{-1} & h_{-2} & 0 & 0 & 0 \\ 0 & h_2 & h_1 & h_0 & h_{-1} & h_{-2} & 0 & 0 \\ 0 & 0 & h_2 & h_1 & h_0 & h_{-1} & h_{-2} & 0 \\ 0 & 0 & 0 & h_2 & h_1 & h_0 & h_{-1} & h_{-2} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 0 & g^1_2 & g^1_1 & g^1_0 & g^1_{-1} \\ 0 & 0 & 0 & 0 & 0 & g^0_2 & g^0_1 & g^0_0 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \cdots \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ \cdots \\ x_6 \\ x_7 \end{bmatrix} \tag{9.8.4}$$

where the filters f^0_m and f^1_m are used for computing the first two transient outputs y_0, y_1, and the filters g^0_m and g^1_m are for the last two outputs y_7, y_6. The corresponding B matrix would be in this case:

$$B = \begin{bmatrix} f^0_0 & f^1_1 & h_2 & 0 & 0 \\ f^0_{-1} & f^1_0 & h_1 & g^1_2 & 0 \\ f^0_{-2} & f^1_{-1} & h_0 & g^1_1 & g^0_2 \\ 0 & f^1_{-2} & h_{-1} & g^1_0 & g^0_1 \\ 0 & 0 & h_{-2} & g^1_{-1} & g^0_0 \end{bmatrix} \tag{9.8.5}$$

More generally, the filters f^i_m, g^i_m, $i = 0, 1, \dots, M-1$, compute the first M and last M output samples y_i, y_{L-1-i}, respectively, through the convolutional equations:

$$\begin{aligned} y_i &= \sum_{m=-M}^{i} f^i_m x_{i-m} = \sum_{m=-i}^{M} f^i_{-m} x_{i+m} = f^i_i x_0 + \cdots + f^i_0 x_i + \cdots + f^i_{-M} x_{i+M} \\ y_{L-1-i} &= \sum_{m=-i}^{M} g^i_m x_{L-1-i-m} = g^i_M x_{L-1-i-M} + \cdots + g^i_0 x_{L-1-i} + \cdots + g^i_{-i} x_{L-1} \end{aligned} \tag{9.8.6}$$

for $i = 0, 1, \dots, M-1$, where the limits of summations follow by the requirement that only available x_n samples appear in the sums.

Musgrave's method [615,616] constructs such asymmetric filters from the knowledge only of the central filter h_m. The construction applies to filters h_m that are symmetric, $h_m = h_{-m}$, and are normalized to unity gain at DC, such as lowpass trend filters,

$$\sum_{m=-M}^{M} h_m = 1 \tag{9.8.7}$$

The asymmetric filters f^i_m, g^i_m are required to satisfy similar moment constraints:

$$\sum_{m=-M}^{i} f^i_m = 1\,, \quad \sum_{m=-i}^{M} g^i_m = 1 \tag{9.8.8}$$

The design is based on a minimum-revision criterion. When the data record has length L, the ith output from the end, y_{L-1-i}, is computed with the filter g^i_m. If or when the series is extended to length $L - 1 + M$, then the same output can actually be computed with the symmetric filter h_m resulting in a revised output y^{rev}_{L-1-i}, that is,

$$y_{L-1-i} = \sum_{m=-i}^{M} g^i_m x_{L-1-i-m}\,, \quad y^{\text{rev}}_{L-1-i} = \sum_{m=-M}^{M} h_m x_{L-1-i-m}$$

Musgrave's criterion selects g^i_m to minimize the mean-square revision error $E[e^2_{L-1-i}]$, where $e_{L-1-i} = y_{L-1-i} - y^{\text{rev}}_{L-1-i}$, under the assumption that locally the input series is linear, that is, $x_{L-1-i-m} = a + bm + v_m$, with a, b constant parameters, and v_m zero-mean white noise with variance σ^2. The mean-square error becomes then,

$$\begin{aligned}
E[e^2_{L-1-i}] &= E\Big[\big[\sum_{m=-i}^{M} g^i_m(a+bm+v_m) - \sum_{m=-M}^{M} h_m(a+bm+v_m)\big]^2\Big] \\
&= E\Big[\big[b\sum_{m=-i}^{M} mg^i_m + \sum_{m=-i}^{M}(g^i_m - h_m)v_m - \sum_{m=-M}^{-i-1} h_m v_m\big]^2\Big] \qquad (9.8.9) \\
&= \sigma^2 \sum_{m=-i}^{M}(g^i_m - h_m)^2 + b^2\Big(\sum_{m=-i}^{M} mg^i_m\Big)^2 + \text{const.}
\end{aligned}$$

where "const." is a positive term independent of g^i_m. In deriving this, we used the moment constraints (9.8.7) and (9.8.8), and the property $\sum_{m=-M}^{M} mh_m = 0$, which follows from the assumed symmetry of h_m. Defining the constant $\beta^2 = b^2/\sigma^2$, it follows that the optimum filter g^i_m will be the solution of the following optimization criterion, which incorporates the constraint (9.8.8) by means of a Lagrange multiplier λ:

$$\mathcal{J} = \sum_{m=-i}^{M}(g_m - h_m)^2 + \beta^2\Big(\sum_{m=-i}^{M} mg_m\Big)^2 + \lambda\Big(1 - \sum_{m=-i}^{M} g_m\Big) = \min \qquad (9.8.10)$$

In a similar fashion, we can show that the filters f^i_m are the solutions of

$$\mathcal{J} = \sum_{m=-M}^{i}(f_m - h_m)^2 + \beta^2\Big(\sum_{m=-M}^{i} mf_m\Big)^2 + \lambda\Big(1 - \sum_{m=-M}^{i} f_m\Big) = \min \qquad (9.8.11)$$

Because h_m is even in m it follows (by changing variables $m \to -m$ in the sums) that $f^i_m = g^i_{-m}$, that is, the beginning filters are the reverse of the end filters. Thus, only g^i_m need be determined and is found to be [616]:

$$g^i_m = h_m + \frac{A_i}{M+i+1} + \frac{\beta^2 B_i}{D_i}(m - \mu_i)\,, \quad -i \le m \le M \qquad (9.8.12)$$

for $i = 0, 1, \dots, M-1$, with the constants A_i, B_i, D_i, μ_i defined by,

$$\begin{aligned}
&A_i = \sum_{m=-M}^{-i-1} h_m\,, \quad B_i = \sum_{m=-M}^{-i-1}(m - \mu_i)h_m\,, \quad i = 0, 1, \dots, M-1 \\
&\mu_i = \frac{M-i}{2}\,, \quad D_i = 1 + \frac{\beta^2}{12}(M+i)(M+i+1)(M+i+2)
\end{aligned} \qquad (9.8.13)$$

To show Eq. (9.8.12), we set the gradient of $\mathcal{J}$ in (9.8.10) to zero, $\partial\mathcal{J}/\partial g_m = 0$, to get,

$$g_m = h_m + \lambda - \beta^2 Gm\,, \quad G = \sum_{m=-i}^{M} mg_m \qquad (9.8.14)$$

Summing up over m and using the constraint (9.8.8), and then, multiplying by m and summing up over m, results in two equations for the two unknowns λ, G:

$$\begin{aligned}
1 &= \sum_{m=-i}^{M} h_m + \Big(\sum_{m=-i}^{M} 1\Big)\lambda - \Big(\sum_{m=-i}^{M} m\Big)\beta^2 G \\
G &= \sum_{m=-i}^{M} mh_m + \Big(\sum_{m=-i}^{M} m\Big)\lambda - \Big(\sum_{m=-i}^{M} m^2\Big)\beta^2 G
\end{aligned} \qquad (9.8.15)$$

Using the properties,

$$1 - \sum_{m=-i}^{M} h_m = \sum_{m=-M}^{-i-1} h_m\,, \quad \sum_{m=-i}^{M} mh_m = -\sum_{m=-M}^{-i-1} mh_m$$

and the identities,

$$\begin{aligned}
&\sum_{m=-i}^{M} 1 = M + i + 1 \\
&\sum_{m=-i}^{M} m = \frac{1}{2}(M+i+1)(M-i) = (M+i+1)\mu_i \\
&\sum_{m=-i}^{M} m^2 = (M+i+1)\mu_i^2 + \frac{1}{12}(M+i)(M+i+1)(M+i+2)
\end{aligned} \qquad (9.8.16)$$

and solving Eqs. (9.8.15) for the constants λ, G and substituting them in (9.8.14), gives the solution (9.8.12). The parameter β is usually computed in terms of the Musgrave parameter R, the two being related by

$$R^2 = \frac{4}{\pi\beta^2} \quad \Rightarrow \quad \beta^2 = \frac{4}{\pi R^2} \qquad (9.8.17)$$

The MATLAB function `minrev` implements Eq. (9.8.12) and arranges the asymmetric filters into a filtering matrix B, which can be passed into the filtering function `lpfilt`,

```
B = minrev(h,R);        % minimum-revision asymmetric filters
```

The input is any odd-length symmetric filter h_m and the parameter R. Typical values of R are given in Eq. (9.7.3). The value $R = \infty$ corresponds to slope $\beta = 0$. For $R = 0$ or $\beta = \infty$, the limit of the solution (9.8.12) is ignored and, instead, the function `minrev` generates the usual convolutional transients for the filter h_m, resulting in a matrix B such that in Eqs. (9.8.3).

We have made extensive use of this function since Chap. 3. As a further example, we compare the filtering matrix B for a 7-term Henderson filter resulting from `minrev` with the standard value $R = 4.5$ to that resulting from the function `lprs` using the corresponding prediction filters for the same Henderson filter:

$$h = \mathrm{lprs2}(7,3,3) = [-0.0587,\ 0.0587,\ 0.2937,\ 0.4126,\ 0.2937,\ 0.0587,\ -0.0587]$$

$$B = \mathrm{minrev}(h, 4.5) = \begin{bmatrix} 0.5345 & 0.2892 & 0.0336 & -0.0587 & 0 & 0 & 0 \\ 0.3833 & 0.4103 & 0.2747 & 0.0587 & -0.0531 & 0 & 0 \\ 0.1160 & 0.2937 & 0.3997 & 0.2937 & 0.0582 & -0.0542 & 0 \\ -0.0338 & 0.0610 & 0.2870 & 0.4126 & 0.2870 & 0.0610 & -0.0338 \\ 0 & -0.0542 & 0.0582 & 0.2937 & 0.3997 & 0.2937 & 0.1160 \\ 0 & 0 & -0.0531 & 0.0587 & 0.2747 & 0.4103 & 0.3833 \\ 0 & 0 & 0 & -0.0587 & 0.0336 & 0.2892 & 0.5345 \end{bmatrix}$$

$$B = \mathrm{lprs}(7,3,3) = \begin{bmatrix} 0.8182 & 0.1836 & -0.0587 & -0.0587 & 0.0336 & 0.0682 & -0.1049 \\ 0.4895 & 0.4510 & 0.2741 & 0.0587 & -0.0951 & -0.0874 & 0.1818 \\ -0.2448 & 0.4283 & 0.5245 & 0.2937 & -0.0140 & -0.1486 & 0.1399 \\ -0.2797 & 0.1049 & 0.3357 & 0.4126 & 0.3357 & 0.1049 & -0.2797 \\ 0.1399 & -0.1486 & -0.0140 & 0.2937 & 0.5245 & 0.4283 & -0.2448 \\ 0.1818 & -0.0874 & -0.0951 & 0.0587 & 0.2741 & 0.4510 & 0.4895 \\ -0.1049 & 0.0682 & 0.0336 & -0.0587 & -0.0587 & 0.1836 & 0.8182 \end{bmatrix}$$

The mean-square revision error (9.8.9) was calculated assuming a local linear function of time for the input. The criterion can be generalized to higher-order polynomials. For example, for a second-order polynomial,

$$x_{L-1-i-m} = a + bm + cm^2 + v_m$$

the mean-square revision error will be:

$$\begin{aligned} E[e_{L-1-i}^2] &= E\Big[\big[\sum_{m=-i}^{M} g_m(a+bm+cm^2+v_m) - \sum_{m=-M}^{M} h_m(a+bm+cm^2+v_m)\big]^2\Big] \\ &= \sigma^2 \sum_{m=-i}^{M}(g_m-h_m)^2 + c^2\Big(\sum_{m=-i}^{M} m^2 g_m\Big)^2 + \text{const.} \end{aligned} \tag{9.8.18}$$

where we assumed that h_m is symmetric, has unity gain at DC, and zero second moment (i.e., it reproduces second-order polynomials). Similarly, we assumed that g_m has unity gain at DC and zero first moment (so that it reproduces first-order polynomials). Thus, the expression (9.8.18) was obtained under the constraints:

$$h_m = h_{-m}\,, \quad \sum_{m=-M}^{M}\begin{bmatrix} 1 \\ m \\ m^2 \end{bmatrix} h_m = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad \sum_{m=-i}^{M}\begin{bmatrix} 1 \\ m \end{bmatrix} g_m = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \tag{9.8.19}$$

Defining $\gamma^2 = c^2/\sigma^2$, we obtain the following optimization criterion, which incorporates the above constraints on g_m with two Lagrange multipliers λ_1, λ_2:

$$\mathcal{J} = \sum_{m=-i}^{M}(g_m-h_m)^2 + \gamma^2\Big(\sum_{m=-i}^{M} m^2 g_m\Big)^2 + \lambda_1\Big(1 - \sum_{m=-i}^{M} g_m\Big) - \lambda_2\Big(\sum_{m=-i}^{M} m g_m\Big) \tag{9.8.20}$$

The vanishing of the gradient gives:

$$g_m = h_m + \lambda_1 + \lambda_2 m - \gamma^2 G m^2\,, \quad G = \sum_{m=-i}^{M} m^2 g_m \tag{9.8.21}$$

By multiplying by m^0, m^1, m^2 and summing up over m, we obtain three equations for the three unknowns λ_1, λ_2, G,

$$\begin{aligned} 1 &= \sum_{m=-i}^{M} h_m + \Big(\sum_{m=-i}^{M} 1\Big)\lambda_1 + \Big(\sum_{m=-i}^{M} m\Big)\lambda_2 - \Big(\sum_{m=-i}^{M} m^2\Big)\gamma^2 G \\ 0 &= \sum_{m=-i}^{M} m h_m + \Big(\sum_{m=-i}^{M} m\Big)\lambda_1 + \Big(\sum_{m=-i}^{M} m^2\Big)\lambda_2 - \Big(\sum_{m=-i}^{M} m^3\Big)\gamma^2 G \\ G &= \sum_{m=-i}^{M} m^2 h_m + \Big(\sum_{m=-i}^{M} m^2\Big)\lambda_1 + \Big(\sum_{m=-i}^{M} m^3\Big)\lambda_2 - \Big(\sum_{m=-i}^{M} m^4\Big)\gamma^2 G \end{aligned} \tag{9.8.22}$$

Substituting the solutions for λ_1, λ_2, G into (9.8.21) gives the solution:

$$g_m^i = h_m + \frac{A_i}{M+i+1} + \frac{B_i}{\Sigma_i}(m-\mu_i) + \frac{\gamma^2 C_i}{\Delta_i}\big[(m-\mu_i)^2 - \nu_i^2\big]\,, \quad -i \le m \le M \tag{9.8.23}$$

for $i = 0, 1, \dots, M-1$, with the same constants A_i, B_i, μ_i as in Eq. (9.8.13), and with $\Sigma_i, \Delta_i, \nu_i, C_i$ are defined by

$$\begin{aligned} \Sigma_i &= \frac{1}{12}(M+i)(M+i+1)(M+i+2)\,, \quad \nu_i^2 = \frac{1}{12}(M+i)(M+i+2) \\ \Delta_i &= 1 + \frac{\gamma^2}{180}(M+i-1)(M+i)(M+i+1)(M+i+2) \\ C_i &= \sum_{m=-M}^{-i-1}\big[(m-\mu_i)^2 - \nu_i^2\big]h_m \end{aligned} \tag{9.8.24}$$

9.9 Seasonal Whittaker-Henderson Decomposition

There are several other seasonal decomposition methods. The Holt-Winters exponential smoothing method [239–241], which was briefly discussed in Eq. (6.13.7), is a simple, effective, method of simultaneously tracking trend and seasonal components.

Another method is based on a seasonal generalization of the Whittaker-Henderson method [622–625] and we discuss it a more detail in this section.

Model-based methods of seasonal adjustment [626–642] are widely used and are often preferred over the X-11/X-12 methods. They are based on making ARIMA-type models for the trend and seasonal components and then estimating the components using optimum Wiener filters, or their more practical implementation as Kalman filters [643–664]. We encountered some examples in making signal models of exponential-smoothing, spline, and Whittaker-Henderson filters. We will be discussing the state-space approach in a later chapter.

The seasonal generalization of the Whittaker-Henderson method, which was originally introduced by Leser, Akaike, and Schlicht [622–624], differs from the Whittaker-Henderson/Kaiser method that we discussed earlier in that the latter determines the trend t_n using ordinary Whittaker-Henderson smoothing, and then applies a Kaiser-window comb filter to the residual $r_n = y_n - t_n$ to extract the seasonal part s_n. By contrast, in the seasonalized version, t_n and s_n are determined simultaneously from a single optimization criterion. We recall that the ordinary Whittaker-Henderson performance index for estimating the trend t_n is,

$$\mathcal{J} = \sum_{n=0}^{N-1} (y_n - t_n)^2 + \lambda \sum_{n=s}^{N-1} (\nabla^s t_n)^2 = \min \tag{9.9.1}$$

where s is the smoothing order and N, the length of y_n. The seasonalized version with period D replaces this by,

$$\mathcal{J} = \sum_{n=0}^{N-1} (y_n - t_n - s_n)^2 + \lambda \sum_{n=s}^{N-1} (\nabla^s t_n)^2 + \alpha \sum_{n=D-1}^{N-1} (s_n + s_{n-1} + \cdots + s_{n-D+1})^2 = \min \tag{9.9.2}$$

A fourth term, $\beta \sum_{n=D}^{N-1} (s_n - s_{n-D})^2$, may be added [623,624], but it is generally not necessary for the following reason. The minimization of $\mathcal{J}$ forces the sum

$$S_n = s_n + s_{n-1} + \cdots + s_{n-D+1} \tag{9.9.3}$$

to become small, ideally zero, and as a consequence the quantity $s_n - s_{n-D} = S_n - S_{n-1}$ will also be made small. Nevertheless, such a term has been implemented as an option in the function `swhdec` below. Eq. (9.9.2) can be written in a compact vectorial form as,

$$\mathcal{J} = (\mathbf{y} - \mathbf{t} - \mathbf{s})^T (\mathbf{y} - \mathbf{t} - \mathbf{s}) + \lambda\, \mathbf{t}^T (D_s^T D_s)\, \mathbf{t} + \alpha\, \mathbf{s}^T (A^T A) \mathbf{s} = \min \tag{9.9.4}$$

where, as discussed in general terms in Sec. 8.1, the matrices D_s, A have dimensions $(N-s)\times N$ and $(N-D+1)\times N$, respectively, and are the steady-state versions of the convolution matrices of the corresponding filters, that is,

$$\begin{aligned} D_s(z) &= (1 - z^{-1})^s\,, \quad \mathbf{d}_s = \text{binom}(s)\,, \quad D_s = \text{convmat}\big(\text{flip}(\mathbf{d}_s),\, N - s\big)^T \\ A(z) &= \sum_{k=0}^{D-1} z^{-k}\,, \quad \mathbf{a} = [\underbrace{1,\, 1,\, \dots,\, 1}_{D \text{ ones}}]\,, \quad A = \text{convmat}\big(\text{flip}(\mathbf{a}),\, N - D + 1\big)^T \end{aligned} \tag{9.9.5}$$

For example, we have for $N = 7$, $s = 2$, and $D = 4$:

$$D_s = \begin{bmatrix} 1 & -2 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & -2 & 1 \end{bmatrix}, \quad A = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{9.9.6}$$

The solution of the minimization problem (9.9.4) is obtained from the vanishing of the gradient of $\mathcal{J}$ with respect to $\mathbf{t}$ and $\mathbf{s}$, which results in the system and its solution:

$$\begin{aligned} (I + P)\mathbf{t} + \mathbf{s} &= \mathbf{y} \\ \mathbf{t} + (I + Q)\mathbf{s} &= \mathbf{y} \end{aligned} \quad \Rightarrow \quad \boxed{\begin{aligned} \mathbf{t} &= (Q + P + QP)^{-1} Q\mathbf{y} \\ \mathbf{s} &= \mathbf{y} - (I + P)\mathbf{t} \end{aligned}} \tag{9.9.7}$$

where we defined $P = \lambda(D_s^T D_s)$ and $Q = \alpha(A^T A)$. The matrices P, Q and $(Q + P + QP)$ are *banded sparse* matrices and therefore the indicated inverse[†] in (9.9.7) can be computed very efficiently with $O(N)$ operations (provided it is implemented by the backslash operator in MATLAB.)

From the above system we also have, $\mathbf{t} = (I + P)^{-1}(\mathbf{y} - \mathbf{s})$, which has the appealing interpretation that the trend is obtained by an ordinary Whittaker-Henderson smoother, i.e., the operator $(I + P)^{-1}$, applied to the seasonally-adjusted signal $(\mathbf{y} - \mathbf{s})$, which is similar to how the X-11 method obtains the final trend by applying a Henderson filter.

The function `swhdec` implements this method. It has an optional argument for the fourth β-term mentioned above:

```
[yt,ys,yi] = swhdec(y,D,s,lambda,alpha,beta);      % seasonal Whittaker-Henderson
```

The larger the parameters α, β, the closer to zero the quantity (9.9.3), and the “more periodic” the seasonal component. Thus, if one wants to extract a slowly evolving periodic component, one should choose smaller values for these parameters, relative to λ. The latter, can be estimated using the GCV criterion. The simultaneous estimation of λ, α, β can be accomplished by maximizing an appropriate likelihood function in a Bayesian formulation of this method [623,636,638].

The ℓ_1-regularized version can be obtained by replacing the ℓ_2 norms of the regularizing parts by their ℓ_1 norms, that is,

$$\mathcal{J} = \sum_{n=0}^{N-1} (y_n - t_n - s_n)^2 + \lambda \sum_{n=s}^{N-1} |\nabla^s t_n| + \alpha \sum_{n=D-1}^{N-1} |s_n + s_{n-1} + \cdots + s_{n-D+1}| = \min$$

$$\boxed{\mathcal{J} = \|\mathbf{y} - \mathbf{t} - \mathbf{s})\|_2^2 + \lambda\, \|D_s \mathbf{t}\|_1 + \alpha\, \|A\, \mathbf{s}\|_1 = \min} \tag{9.9.8}$$

and can be solved easily with the CVX package.[‡]

Example 9.9.1: We revisit the unemployment data for 16–19 year old men for the 1965–79 period, which we encountered in Example 9.6.2. Fig. 9.9.1 compares the trend/seasonal decomposition obtained by the X-11 method (top graphs) and by the seasonal Whittaker-Henderson (middle graphs), as well as the corresponding L_1 version (bottom graphs). The input parameters were as follows, where λ was determined in Example 9.6.2 by the GCV criterion,

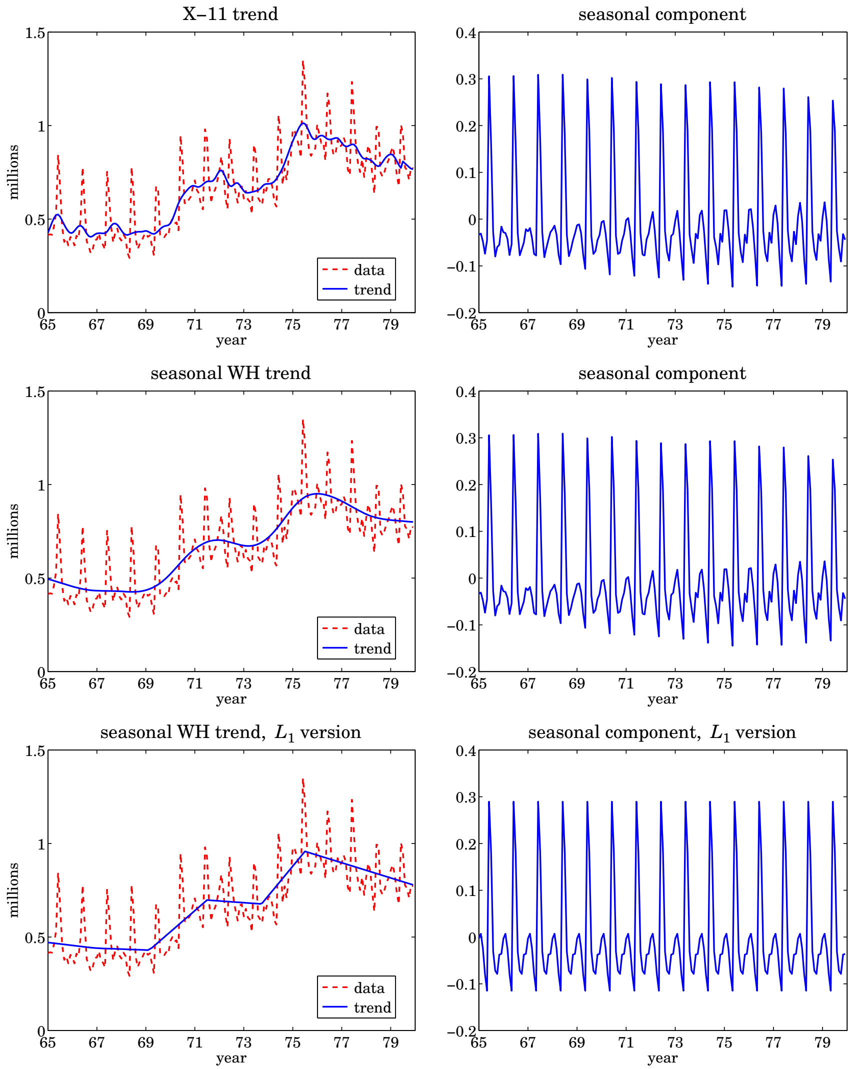
$$D = 12, \quad s = 2, \quad \lambda = 2039, \quad \alpha = 10, \quad \beta = 0$$

The MATLAB code used to generate the six graphs was as follows:

```
Y = loadfile('unemp-1619-nsa.dat'); i=find(Y==1965);      % read data
Y = Y(i:i+14,2:13)'; y = Y(:)/1000; t = taxis(y,12,65);   % extract 1965-79 range

D=12; M1=33; M2=35; N1=13; N2=13; R=3.5; type='a';        % X-11 input parameters
[yt,ys,yi] = x11dec(y,D,M1,M2,N1,N2,R,type);              % X-11 decomposition

figure; plot(t,y, t,yt); figure; plot(t,ys);              % upper graphs
```

[†] It can be shown [624] that the inverse exists for all positive values of λ, α.

[‡] `http://cvxr.com/cvx`


X−11 trend

seasonal component

seasonal WH trend

seasonal component

seasonal WH trend, L_1 version

seasonal component, L_1 version

Fig. 9.9.1 X-11 and seasonal Whittaker-Henderson decomposition methods.

```
D=12; s=2; la=2039; alpha=10;                   % input parameters
[yt,ys,yi] = swhdec(y,D,s,la,alpha);            % seasonal WH decomposition

figure; plot(t,y, t,yt); figure; plot(t,ys);    % middle graphs


la=5; alpha=10;                                 % L1 version
```

```
N = length(y); s=2; Ds = diff(eye(N),s);        % construct matrices Ds and A
A = convmat(ones(1,D), N-D+1)';

cvx_quiet(true);                                % CVX package
cvx_begin
    variable X(2*N)                             % pack trend and seasonal into X
    T = X(1:N); S = X(N+1:2*N);
    minimize( sum_square(y-T-S) + la * norm(Ds*T,1) + alpha * norm(A*S,1) );
cvx_end

yt = X(1:N); ys = X(N+1:2*N);                   % extract trend and seasonal parts

figure; plot(t,y, t,yt); figure; plot(t,ys);    % lower graphs
```

The seasonal components extracted by the methods are comparable, as are the outputs of this method and the Whittaker-Henderson/Kaiser method plotted in Fig. 9.6.2. □

In Sec. 8.2 we obtained the equivalent Whittaker-Henderson trend-extraction filter and showed that it could be thought of as the optimum unrealizable Wiener filter of a particular state-space model. The optimum filter had frequency response:

$$H(\omega)=\frac{1}{1+\lambda\,|D_s(\omega)|^2}\,,\quad \text{where}\quad D_s(\omega)=\big(1-e^{-j\omega}\big)^s \tag{9.9.9}$$

and the state-space model was defined by

$$y_n=t_n+v_n\,,\quad \nabla^s t_n=w_n \tag{9.9.10}$$

where v_n, w_n were zero-mean, mutually-uncorrelated, white-noise signals of variances σ_v^2, σ_w^2, and the smoothing parameter was identified as $\lambda=\sigma_v^2/\sigma_w^2$.

All of these results carry over to the seasonal case. First, we obtain the effective trend and seasonal filters $H_T(\omega), H_S(\omega)$ for extracting t_n, s_n. Then, we show that they are optimal in the Wiener sense. As we did in Sec. 8.2, we consider a double-sided infinitely-long signal y_n and using Parseval's identity, we may write the performance index (9.9.2) in the frequency domain, as follows:

$$\mathcal{J}=\int_{-\pi}^{\pi}\Big[\,|Y(\omega)-T(\omega)-S(\omega)|^2+\lambda\,|D_s(\omega)T(\omega)|^2+\alpha\,|A(\omega)S(\omega)|^2\Big]\frac{d\omega}{2\pi} \tag{9.9.11}$$

where $D_s(\omega)$ and $A(\omega)$ are the frequency responses of the filters in Eq. (9.9.5). From the vanishing of the gradients $\partial\mathcal{J}/\partial T^*$ and $\partial\mathcal{J}/\partial S^*$, we obtain the equations:

$$\begin{aligned} T(\omega)+\lambda\,|D_s(\omega)|^2 T(\omega)+S(\omega) &= Y(\omega)\\ S(\omega)+\alpha\,|A(\omega)|^2 S(\omega)+T(\omega) &= Y(\omega) \end{aligned} \tag{9.9.12}$$

which may be solved for the transfer functions $H_T(\omega)=T(\omega)/Y(\omega)$ and $H_S(\omega)=S(\omega)/Y(\omega)$, resulting in,

$$\begin{aligned} H_T(\omega) &= \frac{\alpha\,|A(\omega)|^2}{\lambda\,|D_s(\omega)|^2+\alpha\,|A(\omega)|^2+\lambda\alpha\,|D_s(\omega)|^2|A(\omega)|^2}\\ H_S(\omega) &= \frac{\lambda\,|D_s(\omega)|^2}{\lambda\,|D_s(\omega)|^2+\alpha\,|A(\omega)|^2+\lambda\alpha\,|D_s(\omega)|^2|A(\omega)|^2} \end{aligned} \tag{9.9.13}$$

with $|D_s(\omega)|^2$ and $|A(\omega)|^2$ given by,

$$
\begin{aligned}
|D_s(\omega)|^2 &= |1 - e^{-j\omega}|^{2s} = |2\sin(\omega/2)|^{2s} \\
|A(\omega)|^2 &= |1 + e^{-j\omega} + \cdots + e^{-j(D-1)\omega}|^2 = \left|\frac{\sin(\omega D/2)}{\sin(\omega/2)}\right|^2
\end{aligned}
\tag{9.9.14}
$$

The filters (9.9.13) generalize the Whittaker-Henderson, or Hodrick-Prescott filter (9.9.9) to the seasonal case. The filters may be identified as the optimum Wiener filters for the following signal model:

$$
y_n = t_n + s_n + v_n\,, \quad \nabla^s t_n = w_n\,, \quad s_n + s_{n-1} + \cdots + s_{n-D+1} = u_n \tag{9.9.15}
$$

where v_n, w_n, u_n are mutually-uncorrelated, zero-mean, white noises. The model can be written symbolically in operator form:

$$
y_n = t_n + s_n + v_n\,, \quad D_s(z)t_n = w_n\,, \quad A(z)s_n = u_n \tag{9.9.16}
$$

The signals t_n, s_n are not stationary, but nevertheless the optimum Wiener filters can be derived as though the signals were stationary [643–649]. Alternatively, multiplication by $D_s(z)A(z)$ acts as a stationarity-inducing transformation, resulting in the stationary signal model,

$$
\begin{aligned}
\bar{y}_n &= D_s(z)A(z)y_n = \bar{t}_n + \bar{s}_n + \bar{v}_n = A(z)w_n + D_s(z)u_n + D_s(z)A(z)v_n \\
\bar{t}_n &= D_s(z)A(z)t_n = A(z)w_n \\
\bar{s}_n &= D_s(z)A(z)s_n = D_s(z)u_n \\
\bar{v}_n &= D_s(z)A(z)v_n
\end{aligned}
\tag{9.9.17}
$$

with spectral densities:

$$
\begin{aligned}
S_{\bar{t}\bar{y}}(\omega) &= S_{\bar{t}\bar{t}}(\omega) = \sigma_w^2|A(\omega)|^2 \\
S_{\bar{s}\bar{y}}(\omega) &= S_{\bar{s}\bar{s}}(\omega) = \sigma_u^2|D_s(\omega)|^2 \\
S_{\bar{y}\bar{y}}(\omega) &= \sigma_u^2|D_s(\omega)|^2 + \sigma_w^2|A(\omega)|^2 + \sigma_v^2|D_s(\omega)A(\omega)|^2
\end{aligned}
$$

It follows from [643–649] that the optimum Wiener filters for estimating t_n, s_n will be:

$$
\begin{aligned}
H_T(\omega) &= \frac{S_{\bar{t}\bar{y}}(\omega)}{S_{\bar{y}\bar{y}}(\omega)} = \frac{\sigma_w^2|A(\omega)|^2}{\sigma_u^2|D_s(\omega)|^2 + \sigma_w^2|A(\omega)|^2 + \sigma_v^2|D_s(\omega)A(\omega)|^2} \\
H_S(\omega) &= \frac{S_{\bar{s}\bar{y}}(\omega)}{S_{\bar{y}\bar{y}}(\omega)} = \frac{\sigma_u^2|D_s(\omega)|^2}{\sigma_u^2|D_s(\omega)|^2 + \sigma_w^2|A(\omega)|^2 + \sigma_v^2|D_s(\omega)A(\omega)|^2}
\end{aligned}
\tag{9.9.18}
$$

It is evident that these are identical to (9.9.13) with the identifications $\lambda = \sigma_v^2/\sigma_w^2$ and $\alpha = \sigma_v^2/\sigma_u^2$. For a finite, length-$N$, signal y_n, the model (9.9.15) has been used to derive Kalman smoothing algorithms for estimating t_n, s_n with $O(N)$ operations, and for efficiently evaluating the model's likelihood function [636,638]. We note, however, that the matrix solutions (9.9.7) are equally efficient.

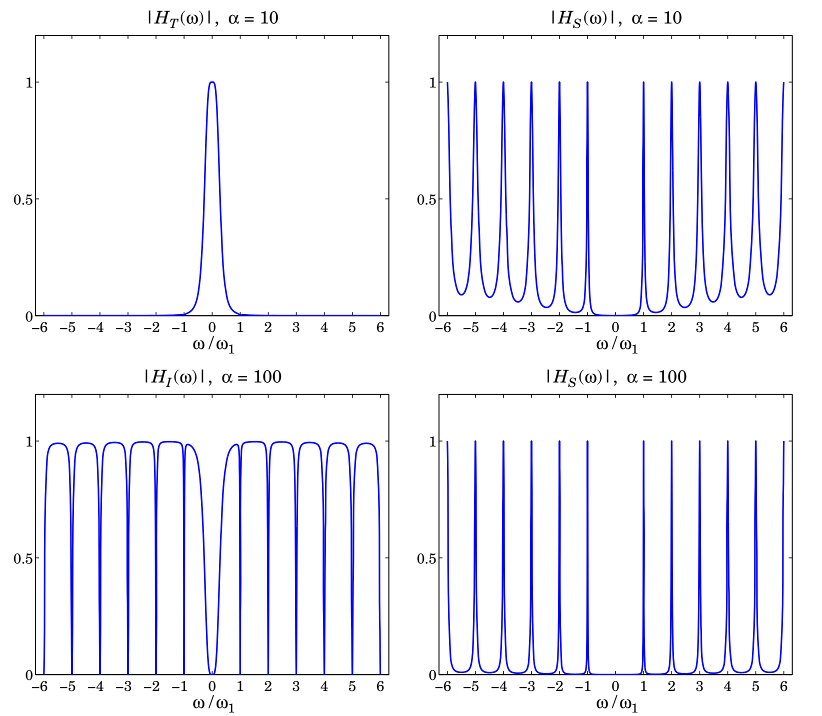


Fig. 9.9.2 Frequency responses of seasonal Whittaker-Henderson filters.

Example 9.9.2: Fig. 9.9.2 plots the frequency responses $H_T(\omega)$ and $H_S(\omega)$ of Eq. (9.9.13). For the upper graphs, the parameter values were the same as those of Example 9.9.1, that is, $D = 12$, $s = 2$, $\lambda = 2039$, $\alpha = 10$. We note that the responses have the expected shapes.

In the lower graphs, we increased the parameter α to 100 in order to sharpen the comb peaks. The lower-left graph depicts the filter $H_I(\omega) = 1 - H_T(\omega) - H_S(\omega)$ for extracting the irregular component, and the right graph depicts $H_S(\omega)$. The trend filter is not shown since it is virtually identical to that of the upper-left graph. The MATLAB code used to generate the upper graphs was as follows:

```
k = linspace(-6,6,2401); w = 2*pi*k/12;          % frequencies −π ≤ ω ≤ π

D = 12; s = 2; la = 2039; alpha = 10;

a = ones(D,1); A = freqz(a,1,w);                 % calculate A(ω)

P = la * abs(1 - exp(-j*w)).^(2*s);              % evaluate P(ω) = λ |D_s(ω)|^2
Q = alpha * abs(A).^2;                           % evaluate Q(ω) = α |A(ω)|^2
R = Q + P + Q.*P;

HT = Q./R; HS = P./R; HI = 1-HS-HT;
```

```
figure; plot(k,HT);  figure; plot(k,HS);          % upper graphs
```

9.10 Problems

9.1 First prove Eq. (9.1.2) for all n. Then, using the DFT/IDFT pair in Eq. (9.1.1), show that a more general form of (9.1.2) is,

$$\sum_{m=0}^{D-1} s_{n-m}e^{j\omega_k m} = e^{j\omega_k n}S_k\,, \quad k = 0,1,\dots,D-1, \quad -\infty < n < \infty$$

9.2 Consider the analog signal $s(t) = \cos(2\pi f_1 t)$ and its sampled version $s_n = \cos(2\pi f_1 nT)$, where T is the sampling interval related to the sampling rate by $f_s = 1/T$. It is required that s_n be periodic in n with period of D samples, that is, $\cos(2\pi f_1(n+D)T) = \cos(2\pi f_1 nT)$, for all n. How does this requirement constrain f_s and f_1?

9.3 Show that the IIR comb and notch filters defined in Eq. (9.1.9) are complementary and power complementary in the sense that they satisfy Eqs. (9.1.7).

Working with the magnitude response $|H_{\text{comb}}(\omega)|^2$ show that the 3-dB width of the comb peaks is given by Eq. (9.1.11).

9.4 Show that the solution of the system (9.9.7) can be written in the more symmetric, but computationally less efficient, form:

$$\mathbf{t} = (Q + P + QP)^{-1}Q\mathbf{y}$$
$$\mathbf{s} = (P + Q + PQ)^{-1}P\mathbf{y}$$

10 Wavelets

Over the past two decades, wavelets have become useful signal processing tools for signal representation, compression, and denoising [665–833]. There exist several books on the subject [665–686], and several tutorial reviews [687–708]. The theory of wavelets and multiresolution analysis is by now very mature [709–761] and has been applied to a remarkably diverse range of applications, such as image compression and coding, JPEG2000 standard, FBI fingerprint compression, audio signals, numerical analysis and solution of integral equations, electromagnetics, biomedical engineering, astrophysics, turbulence, chemistry, infrared spectroscopy, power engineering, economics and finance, bioinformatics, characterization of long-memory and fractional processes, and statistics with regression and denoising applications [762–833].

In this chapter, we present a short review of wavelet concepts, such as multiresolution analysis, dilation equations, scaling and wavelet filters, filter banks, discrete wavelet transforms in matrix and convolutional forms, wavelet denoising, and undecimated wavelet transforms. Our discussion emphasizes computational aspects.

10.1 Multiresolution Analysis

Wavelet multiresolution analysis expands a time signal into components representing different scales—from a coarser to a finer resolution. Each term in the expansion captures the signal details at a particular scale level. The expansion is defined in terms of a sequence of nested closed subspaces V_j of the space $L^2(\mathbb{R})$ of square integrable functions on the real line $\mathbb{R}$:

$$\cdots \subset V_{-2} \subset V_{-1} \subset V_0 \subset V_1 \subset V_2 \subset \cdots \subset L^2(\mathbb{R}) \tag{10.1.1}$$

The space V_j approximates a signal at a scale j with a resolution of 2^{-j} time units. Roughly speaking, if T_0 is the sampling time interval in subspace V_0, then the sampling interval in V_j will be $T_j = 2^{-j}T_0$, which is coarser if $j < 0$, and finer if $j > 0$. The union of the V_j subspaces is the entire $L^2(\mathbb{R})$ space, and their intersection, the zero function:

$$\lim_{j\to\infty} V_j = \bigcup_{j=-\infty}^{\infty} V_j = L^2(\mathbb{R})\,, \quad \lim_{j\to-\infty} V_j = \bigcap_{j=-\infty}^{\infty} V_j = \{0\} \tag{10.1.2}$$

The spaces V_j have a special structure, being defined as the *linear spans* of the scaled and translated replicas of a single function $\phi(t)$, called the *scaling function*, or the *father wavelet*, which can be of compact support. The scaled/translated replicas of $\phi(t)$ are defined for any integers j, n by:

$$\phi_{jn}(t)=2^{j/2}\phi(2^jt-n) \tag{10.1.3}$$

The functions $\phi_{jn}(t)$ are orthonormal for each fixed j, and form a basis of V_j. The orthonormality condition is defined with respect to the $L^2(\mathbb{R})$ inner product:[†]

$$(\phi_{jn},\phi_{jm})=\int_{-\infty}^{\infty}\phi_{jn}(t)\phi_{jm}(t)\,dt=\delta_{nm} \tag{10.1.4}$$

The factor $2^{j/2}$ in Eq. (10.1.3) serves to preserve the unit norm of ϕ_{jn} for each j. Conditions (10.1.1)–(10.1.4) are strong constraints and it is remarkable that such functions $\phi(t)$ exist other than the simple Haar function defined to be unity over $0\le t\le 1$ and zero otherwise. Fig. 10.1.1 shows three examples, the Haar, and the Daubechies D_2 and D_3 cases, all of which have compact support (the support of D_2 is 3 time units and that of D_3, 5 units). The figure also shows a related function $\psi(t)$ derived from $\phi(t)$, called the *wavelet function*, or the *mother wavelet*.

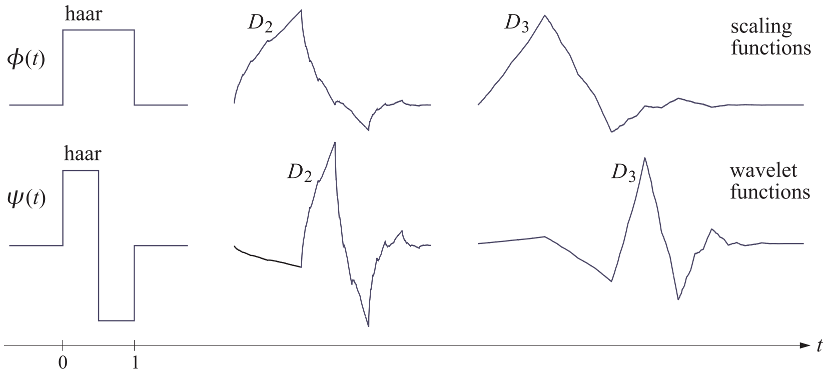


Fig. 10.1.1 Haar, Daubechies D_2 and D_3 scaling and wavelet functions $\phi(t),\psi(t)$.

Fig. 10.1.2 shows the scaled versions of the scaling and wavelet functions (for the D_2 case). Each successive copy $\phi(t),\phi(2t),\phi(2^2t),\phi(2^3t)$, etc., is compressed by a factor of two relative to the previous one, so that for higher and higher values of j, the basis function $\phi(2^jt)$ is capable of capturing smaller and smaller signal details.

The *projection* of an arbitrary signal $f(t)\in L^2(\mathbb{R})$ onto the subspace V_j is defined by the following expansion in the ϕ_{jn} basis:

$$f_j(t)=\sum_n c_{jn}\phi_{jn}(t)=\sum_n c_{jn}2^{j/2}\phi(2^jt-n) \tag{10.1.5}$$

[†]In this chapter, all time signals are assumed to be real-valued.

with coefficients following from the orthonormality of $\phi_{jn}(t)$:

$$c_{jn}=(f_j,\phi_{jn})=(f,\phi_{jn})=\int_{-\infty}^{\infty}f(t)\phi_{jn}(t)\,dt=\int_{-\infty}^{\infty}f(t)2^{j/2}\phi(2^jt-n)\,dt \tag{10.1.6}$$

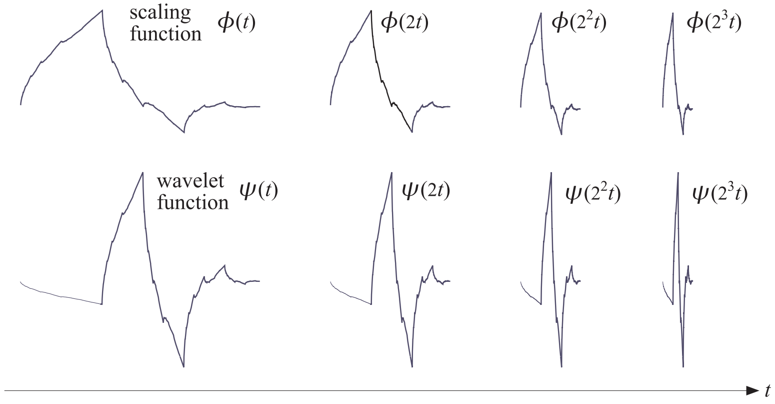


Fig. 10.1.2 Daubechies D_2 functions $\phi(t),\psi(t)$ and their compressed versions.

The projection $f_j(t)$ can be thought of as an approximation of $f(t)$ at scale j with time resolution of 2^{-j}. Because $V_i\subset V_j$ for $i\le j$, the signal $f_j(t)$ incorporates information about $f(t)$ from all coarser resolutions (cf. Eq. (10.2.8)).

The significance of the wavelet function $\psi(t)$ is that the orthogonal complement $V_j^{\perp}$ of V_j with respect to $L^2(\mathbb{R})$ is actually spanned by the scaled and translated versions of ψ, that is, $\psi_{in}(t)=2^{i/2}\psi(2^it-n)$ for $i\ge j$, which are orthogonal to $\phi_{jn}(t)$, and are also mutually orthonormal,

$$(\phi_{jn},\psi_{im})=0\,,\quad i\ge j\,,\qquad (\psi_{in},\psi_{i'n'})=\delta_{ii'}\delta_{nn'} \tag{10.1.7}$$

Thus, we have the direct sum $L^2(\mathbb{R})=V_j\oplus V_j^{\perp}$, resulting in the decomposition of $f(t)$ into two orthogonal parts:

$$f(t)=f_j(t)+w_j(t)\,,\quad f_j(t)\in V_j\,,\quad w_j(t)\in V_j^{\perp}\,,\quad f_j(t)\perp w_j(t) \tag{10.1.8}$$

The component $w_j(t)$ is referred to as the "detail," and incorporates the details of $f(t)$ at all the higher resolution levels $i\ge j$, or finer time scales $2^{-i}\le 2^{-j}$. It admits the ψ-basis expansion:

$$w_j(t)=\sum_{i\ge j}\sum_n d_{in}\psi_{in}(t)=\sum_{i\ge j}\sum_n d_{in}2^{i/2}\psi(2^it-n) \tag{10.1.9}$$

with detail coefficients $d_{in}=(w_j,\psi_{in})=(f,\psi_{in})$. In summary, one form of the multiresolution decomposition is,

$$\boxed{f(t)=f_j(t)+w_j(t)=\sum_n c_{jn}\phi_{jn}(t)+\sum_{i=j}^{\infty}\sum_n d_{in}\psi_{in}(t)} \tag{10.1.10}$$

Another form is obtained in the limit $j\to-\infty$. Since $V_{-\infty}=\{0\}$, we have $f_{-\infty}(t)=0$, and we obtain the representation of $f(t)$ purely in terms of the wavelet basis $\psi_{in}(t)$:

$$\boxed{f(t)=\sum_{i=-\infty}^{\infty}\sum_n d_{in}\psi_{in}(t)}\,,\qquad d_{in}=\int_{-\infty}^{\infty}f(t)\psi_{in}(t)\,dt \tag{10.1.11}$$

Yet another, and most practical, version of the multiresolution decomposition is obtained by noting that $V_\infty = L^2(\mathbb{R})$. We may assume then that our working signal $f(t)$ belongs to some V_J for a sufficiently large value of J, representing the highest desired resolution, or finest scale. Since $f(t)\in V_J$, it follows from the decomposition $f(t)=f_J(t)+w_J(t)$ that $w_J(t)=0$, which implies that $d_{in}=0$ for $i\ge J$, and therefore,

$$f(t)=f_J(t)=\sum_n c_{Jn}\phi_{Jn}(t) \tag{10.1.12}$$

Combining this with Eq. (10.1.10) applied at some lower resolution $j<J$, we obtain the two alternative forms (cf. Eq. (10.2.10)):

$$\boxed{f(t)=\sum_n c_{Jn}\phi_{Jn}(t)=\sum_n c_{jn}\phi_{jn}(t)+\sum_{i=j}^{J-1}\sum_n d_{in}\psi_{in}(t)=f_j(t)+w_j(t)} \tag{10.1.13}$$

The mapping of the expansion coefficients from level J to levels j through $J-1$,

$$c_{Jn}\to\{c_{jn};\ d_{in},\, j\le i\le J-1\} \tag{10.1.14}$$

is essentially the discrete wavelet transform (DWT). For sufficiently large J, the coefficients c_{Jn} can be taken to be the time samples of $f(t)$, sampled at the rate $f_s=2^J$, or sampling time interval $T_J=2^{-J}$. To see this, we note that the function $2^J\phi(2^Jt-n)$ tends to a Dirac delta function for large J (see [665] for a proof), so that,

$$2^J\phi(2^Jt-n)\approx 2^J\delta(2^Jt-n)=\delta(t-n2^{-J}) \tag{10.1.15}$$

Therefore, $2^{J/2}\phi(2^Jt-n)\approx 2^{-J/2}\delta(t-n2^{-J})$, and Eq. (10.1.6) gives,

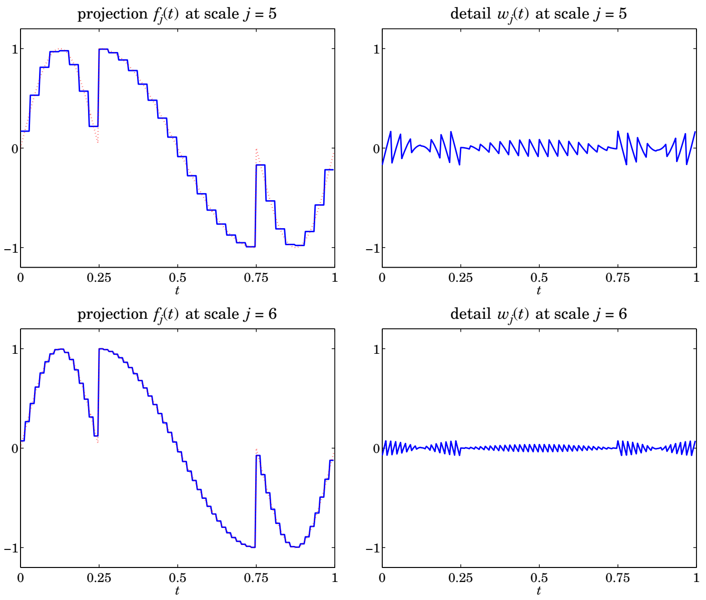
$$c_{Jn}\approx\int_{-\infty}^{\infty}f(t)2^{-J/2}\delta(t-n2^{-J})\Phi_0dt=2^{-J/2}f(n2^{-J}) \tag{10.1.16}$$

In practice, we may ignore the factor $2^{-J/2}$ and set simply $c_{Jn}=f(n2^{-J})=f(nT_J)$. The coefficients c_{Jn} serve as the input to the discrete wavelet transform. The approximation of c_{Jn} by the time samples is usually adequate, although there exist more precise ways to initialize the transform.

Example 10.1.1: An example of the decomposition (10.1.13) is shown in Fig. 10.1.3 using the Haar basis. The original signal (dotted line) is defined by sampling the following discontinuous function at $N=2^8=256$ equally spaced points over the interval $0\le t\le 1$,

$$f(t)=\begin{cases}\sin(4\pi t), & 0\le t<0.25\\ \sin(2\pi t), & 0.25\le t<0.75\\ \sin(4\pi t), & 0.75\le t<1\end{cases} \tag{10.1.17}$$

Thus, the highest resolution level is $J=\log_2 N=8$. The upper graphs show the components $f_j(t),w_j(t)$ for the lower resolution level of $j=5$. The bottom graphs correspond to $j=6$. As j increases, the step-function approximation becomes more accurate and captures better the two sharp breaks of the original signal. For each j, the sums of the left and right graphs make up the original signal.


projection $f_j(t)$ at scale $j = 5$

detail $w_j(t)$ at scale $j = 5$

projection $f_j(t)$ at scale $j = 6$

detail $w_j(t)$ at scale $j = 6$

Fig. 10.1.3 Haar-basis projections $f_j(t),w_j(t)$ from scale $J=8$ to scales $j=5,6$.

Fig. 10.1.4 shows the case of using the Daubechies D_3 wavelet basis for the same signal (10.1.17). The following MATLAB code generates the top graphs in the two figures:

```
J = 8; N = 2^J;
t1 = (0:N/4-1)'/N; t2 = (N/4:3*N/4-1)'/N; t3 = (3*N/4:N-1)'/N;
t = [t1; t2; t3];
y = [sin(4*pi*t1); sin(2*pi*t2); sin(4*pi*t3)];     % define signal

h = daub(1);                                    % use h=daub(3) for Fig. 10.1.4
j = 5; Y = dwtdec(y,h,j);                       % DWT decomposition to level j
fj = Y(:,1); wj = sum(Y(:,2:end),2);            % approximation fj(t) and detail wj(t)

figure; plot(t,fj, t,y,':'); figure; plot(t,wj);       % left, right graphs
```

The function `dwtdec` is explained in Sec. 10.5, but we mention here that its output Y is an $N\times(J-j+1)$ matrix whose first column holds the projection $f_j(t)$, and the sum of its other columns are the detail $w_j(t)$. □

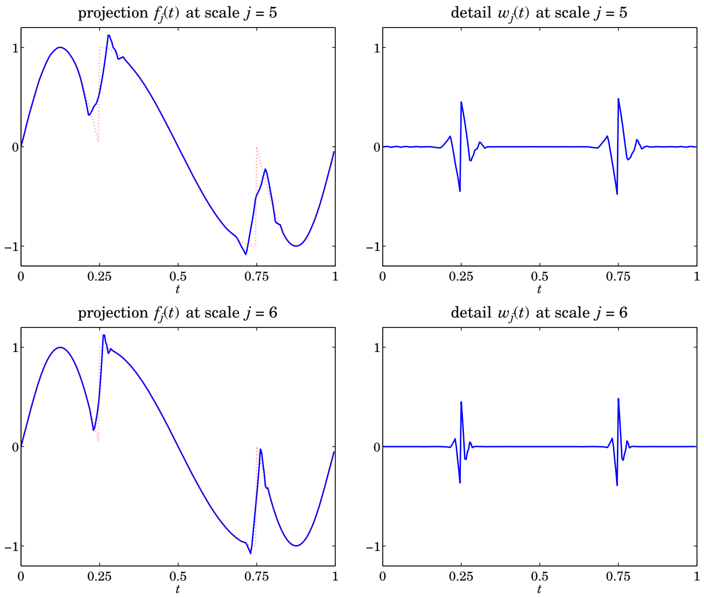


Fig. 10.1.4 Daubechies-D_3 projections $f_j(t), w_j(t)$ from scale $J = 8$ to scales $j = 5, 6$.

10.2 Dilation Equations

The subspaces V_j have even more interesting structure than described so far. Since $V_0 \subset V_1$, it follows that the scaling function $\phi(t)\in V_0$ can be expanded in the basis $\phi_{1n}(t) = 2^{1/2}\phi(2t-n)$ that spans V_1, that is, there must exist coefficients h_n such that

$$\boxed{\phi(t) = \sum_n h_n 2^{1/2}\phi(2t-n)} \qquad \text{(dilation equation)} \tag{10.2.1}$$

which is known as the dilation or refinement equation. The coefficients h_n are given by:

$$h_n = (\phi, \phi_{1n}) = 2^{1/2}\int_{-\infty}^{\infty}\phi(t)\phi(2t-n)\,dt \tag{10.2.2}$$

Moreover, the wavelet function $\psi(t)$ and its translates $\psi_{0n} = \psi(t-n)$ form an orthonormal basis for the orthogonal complement of V_0 relative to V_1, that is, the space $W_0 = V_1 \setminus V_0$, so that we have the direct-sum decomposition:

$$V_0 \oplus W_0 = V_1 \tag{10.2.3}$$

The space W_0 is referred to as the “detail” subspace. Because $\psi(t)\in W_0 \subset V_1$, it also can be expanded in the $\phi_{1n}(t)$ basis, as in Eq. (10.2.1),

$$\boxed{\psi(t) = \sum_n g_n 2^{1/2}\phi(2t-n)} \tag{10.2.4}$$

In a similar fashion, we have the decomposition $V_j \oplus W_j = V_{j+1}$, for all j, with W_j being spanned by the scaled/translated ψ-basis, $\psi_{jn}(t) = 2^{j/2}\psi(2^j t - n)$. The dilation equations can also be written with respect to the ϕ_{jn}, ψ_{jn} bases. For example,

$$\phi_{jk}(t) = 2^{j/2}\phi(2^j t - k) = \sum_m h_m 2^{(j+1)/2}\phi(2^{j+1}t - 2k - m) = \sum_m h_m \phi_{j+1,m+2k}(t)$$

and similarly for $\psi_{jk}(t)$. Thus, we obtain the alternative forms,

$$\begin{aligned}\phi_{jk}(t) &= \sum_m h_m\phi_{j+1,m+2k}(t) = \sum_n h_{n-2k}\phi_{j+1,n}(t)\\ \psi_{jk}(t) &= \sum_m g_m\phi_{j+1,m+2k}(t) = \sum_n g_{n-2k}\phi_{j+1,n}(t)\end{aligned} \tag{10.2.5}$$

Using the orthogonality property $(\phi_{j+1,n}, \phi_{j+1,m}) = \delta_{nm}$, we have the inner products,

$$\begin{aligned}h_{n-2k} &= (\phi_{jk}, \phi_{j+1,n})\\ g_{n-2k} &= (\psi_{jk}, \phi_{j+1,n})\end{aligned} \tag{10.2.6}$$

Also, because $\phi_{j+1,n}(t)$ is a basis for $V_{j+1} = V_j \oplus W_j$, it may be expanded into its two orthogonal parts belonging to the subspaces V_j and W_j, which are in turn spanned by ϕ_{jk} and ψ_{jk}, that is,

$$\phi_{j+1,n} = \sum_k (\phi_{j+1,n}, \phi_{jk})\phi_{jk} + \sum_k(\phi_{j+1,n}, \psi_{jk})\psi_{jk}$$

Using (10.2.6), we may rewrite this as,

$$\phi_{j+1,n}(t) = \sum_k h_{n-2k}\phi_{jk}(t) + \sum_k g_{n-2k}\psi_{jk}(t) \tag{10.2.7}$$

Eqs. (10.2.5)–(10.2.7) are the essential tools for deriving Mallat’s pyramidal multiresolution algorithm for the discrete wavelet transform.

The various decompositions discussed in Sec. 10.1 can be understood in the geometric language of subspaces. For example, starting at level j and repeating the direct-sum decomposition, and using $V_{-\infty} = \{0\}$, we obtain the representation of the subspace V_j,

$$V_j = V_{j-1}\oplus W_{j-1} = V_{j-2}\oplus W_{j-2}\oplus W_{j-1} = \cdots = \bigoplus_{i=-\infty}^{j-1} W_i \tag{10.2.8}$$

which states that V_j incorporates the details of all coarser resolutions. Similarly, increasing j and using $V_\infty = L^2(\mathbb{R})$, we obtain the subspace interpretation of Eq. (10.1.10),

$$\begin{aligned}V_{j+1} &= V_j \oplus W_j\\ V_{j+2} &= V_{j+1}\oplus W_{j+1} = V_j \oplus W_j \oplus W_{j+1}\\ V_{j+3} &= V_{j+2}\oplus W_{j+2} = V_j\oplus W_j\oplus W_{j+1}\oplus W_{j+2}\\ &\cdots\\ L^2(\mathbb{R}) &= V_j \oplus \big(W_j \oplus W_{j+1}\oplus W_{j+2}\oplus\cdots\big) = V_j \oplus V_j^{\perp}\end{aligned} \tag{10.2.9}$$

which explains the remark that the term $w_j(t)$ in (10.1.10) incorporates all the higher-level details. Finally, going from level $j < J$ to level $J-1$, we obtain the geometric interpretation of Eq. (10.1.13),

$$V_J = V_j \oplus \big(W_j \oplus W_{j+1} \oplus \cdots \oplus W_{J-1}\big)\,, \qquad j < J \tag{10.2.10}$$

The coefficients h_n define a lowpass filter $H(z) = \sum_n h_n z^{-n}$ called the *scaling filter*. Similarly, g_n define a highpass filter $G(z)$, the *wavelet filter*. The coefficients h_n, g_n must satisfy certain orthogonality relations, discussed below, that follow from the dilation equations (10.2.5).

The filters h_n, g_n can be IIR or FIR, but the FIR ones are of more practical interest, and lead to functions $\phi(t), \psi(t)$ of compact support. Daubechies [665] has constructed several families of such FIR filters: the minimum-phase family or daublets, the least-asymmetric family or symmlets, and coiflets. The MATLAB function daub incorporates these three families:

```
h = daub(K,type);        % Daubechies scaling filters - daublets, symmlets, coiflets
```

```
h = daub(K,1) = Daublets K = 1,2,3,4,5,6,7,8,9,10, denoted as D_K (D_1 = Haar)
h = daub(K,2) = Symmlets K = 4,5,6,7,8,9,10, denoted as S_K
h = daub(K,3) = Coiflets K = 1,2,3,4,5
h = daub(K)   = equivalent to daub(K,1)

Daublets (minimum phase) have length = 2K and K vanishing moments for ψ(t).
Symmlets (least asymmetric) have length = 2K and K vanishing moments for ψ(t).
Coiflets have length = 6K and 2K vanishing moments for ψ(t), and 2K-1 for φ(t).
         for coiflets, h(n) is indexed over -2K <= n <= 4K-1

all filters have norm(h) = 1 and sum(h) = √2
```

For example, the scaling filters for the Haar, and daublet D_2 and D_3 cases, whose $\phi(t), \psi(t)$ functions were shown in Fig. 10.1.1, are obtained from the MATLAB calls:

```
h = daub(1) ⇒ h = [0.7071    0.7071]
h = daub(2) ⇒ h = [0.4830    0.8365    0.2241   -0.1294]
h = daub(3) ⇒ h = [0.3327    0.8069    0.4599   -0.1350   -0.0854    0.0352]
```

The filters h_n can be taken to be causal, i.e., h_n, $0 \le n \le M$, where M is the filter order, which is odd for the above three families, with $M = 2K-1$ for daublets and symmlets, and $M = 6K-1$ for coiflets (these are defined to be slightly anticausal, over $-2K \le n \le 4K-1$). The parameter K is related to certain flatness constraints or moment constraints for h_n at the Nyquist frequency $\omega = \pi$.

The filters g_n are defined to be the *conjugate* or *quadrature* mirror filters to h_n, that is, $g_n = (-1)^n h_n^R$, where $h_n^R = h_{M-n}$, $n = 0, 1, \dots, M$ is the reversed version of h_n.

In the z-domain, we have $H^R(z) = z^{-M}H(z^{-1})$, while multiplication by $(-1)^n$ is equivalent to the substitution $z \to -z$, therefore, $G(z) = H^R(-z) = (-z)^{-M}H(-z^{-1})$. In the frequency domain, this reads:

$$\boxed{G(\omega) = e^{-jM(\omega+\pi)}H^*(\omega+\pi) \quad \Leftrightarrow \quad g_n = (-1)^n h_{M-n}\,, \quad 0 \le n \le M} \tag{10.2.11}$$

with the frequency-responses defined by:

$$G(\omega) = \sum_{n=0}^{M} g_n e^{-j\omega n}\,, \quad H(\omega) = \sum_{n=0}^{M} h_n e^{-j\omega n} \tag{10.2.12}$$

The function cmf implements this definition:

```
g = cmf(h);        % conjugate mirror filter
```

For example, if $\mathbf{h} = [h_0, h_1, h_2, h_3]$, then, $\mathbf{g} = [h_3, -h_2, h_1, -h_0]$, e.g., we have for the daublet D_2:

```
h = daub(2) = [ 0.4830    0.8365    0.2241   -0.1294]
g = cmf(h)  = [-0.1294   -0.2241    0.8365   -0.4830]
```

Fig. 10.2.1 shows the magnitude responses of the Haar, Daubechies D_2, and Symmlet S_6 scaling and wavelet filters.

For all scaling filters, the DC gain of $H(\omega)$, and the Nyquist gain of $G(\omega)$, are equal to $\sqrt{2}$ because of the conditions (which are a consequence of the dilation equation):

$$\begin{aligned} H(0) &= \sum_{n=0}^{M} h_n = \sqrt{2} \\ G(\pi) &= \sum_{n=0}^{M} (-1)^n g_n = \sum_{n=0}^{M} h_{M-n} = H(0) = \sqrt{2} \end{aligned} \tag{10.2.13}$$

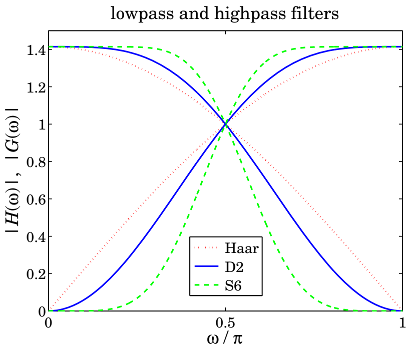


Fig. 10.2.1 Haar, Daubechies D_2, and Symmlet S_6 scaling and wavelet filters.

The dilation equations can be expressed in the frequency domain as follows:

$$\begin{aligned} \Phi(\omega) &= 2^{-1/2}H(2^{-1}\omega)\Phi(2^{-1}\omega) \\ \Psi(\omega) &= 2^{-1/2}G(2^{-1}\omega)\Phi(2^{-1}\omega) \end{aligned} \tag{10.2.14}$$

where $\Phi(\omega), \Psi(\omega)$ are the Fourier transforms:

$$\Phi(\omega) = \int_{-\infty}^{\infty} \phi(t)e^{-j\omega t}\,dt\,, \quad \Psi(\omega) = \int_{-\infty}^{\infty} \psi(t)e^{-j\omega t}\,dt \tag{10.2.15}$$

In fact, setting $\omega = 0$ in the first of (10.2.14) and assuming that $\Phi(0)\neq 0$, we immediately obtain the gain conditions (10.2.13). The iteration of Eqs. (10.2.14) leads to the infinite product expressions:

$$\begin{aligned}\Phi(\omega) &= \Phi(0)\prod_{j=1}^{\infty}\Big[2^{-1/2}H(2^{-j}\omega)\Big]\\ \Psi(\omega) &= \Phi(0)\Big[2^{-1/2}G(2^{-1}\omega)\Big]\prod_{j=2}^{\infty}\Big[2^{-1/2}H(2^{-j}\omega)\Big]\end{aligned}\tag{10.2.16}$$

We show later that $\Phi(0)$ can be chosen to be unity, $\Phi(0)= 1$. As an example, Fig. 10.2.2 shows the normalized magnitude spectra $|\Phi(\omega)|$ and $|\Psi(\omega)|$, where the infinite products were replaced by a finite number of factors up to a maximum $j\leq J$ chosen such that the next factor $J+1$ would add a negligible difference to the answer. For Fig. 10.2.2, an accuracy of 0.001 percent was achieved with the values of $J = 7$ and $J = 5$ for the left and right graphs, respectively. The following MATLAB code illustrates the generation of the left graph:

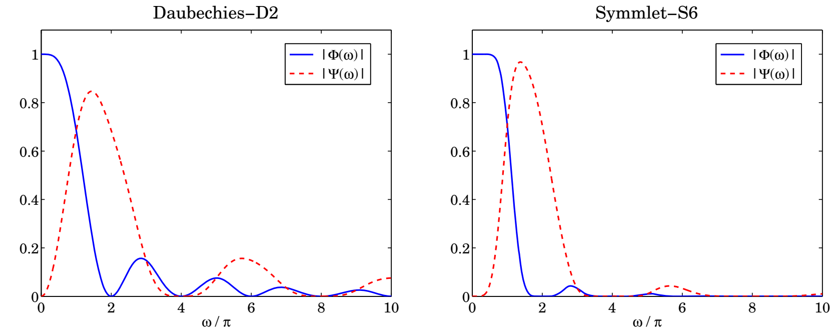


Fig. 10.2.2 Fourier transforms $\Phi(\omega),\Psi(\omega)$ of scaling and wavelet functions $\phi(t),\psi(t)$.

```
epsilon = 1e-5; Jmax = 30;                       % percent error = 100ε
f = linspace(0,10,513); w = pi*f;                % frequency range in units of π
h = daub(2)/sqrt(2); g = cmf(h);                 % normalize h_n such that H(0) = 1.
                                                 % use h=daub(6,2)/sqrt(2) for right graph
Phi0 = abs(freqz(h,1,w/2));                      % initialize recursions at |H(ω/2)|, |G(ω/2)|
Psi0 = abs(freqz(g,1,w/2));

for J = 2:Jmax,
   Phi = Phi0 .* abs(freqz(h,1,w/2^J));          % update by the factor |H(2^{-J}ω)|
   Psi = Psi0 .* abs(freqz(h,1,w/2^J));
   if norm(Phi-Phi0) < norm(Phi0) * epsilon, J, break; end        % stopping J
   Phi0 = Phi;
   Psi0 = Psi;
end

figure; plot(f,Phi, f,Psi,'--');
```

We observe from the graphs that $\Phi(\omega)$ has zeros at $\omega = 2\pi m$ for non-zero integers m. This is justified in the next section. Similarly, $\Psi(\omega)$ vanishes at $\omega = 2\pi m$ for even m, including zero.

Given the filters h_n, g_n, the dilation equations (10.2.1) and (10.2.4) can be solved iteratively for the functions $\phi(t),\psi(t)$ by the so-called *cascade algorithm*, which amounts to the iterations,

$$\begin{aligned}\phi^{(r+1)}(t) &= \sum_n h_n\, 2^{1/2}\phi^{(r)}(2t-n)\\ \psi^{(r+1)}(t) &= \sum_n g_n\, 2^{1/2}\phi^{(r)}(2t-n)\end{aligned}\tag{10.2.17}$$

for $r = 0,1,2,\dots$, starting with some simple initial choice, such as $\phi^{(0)}(t)= 1$. The iteration converges quickly for all the scaling filters incorporated into the function daub. The algorithm can be cast as a convolutional operation with the so-called à trous[†] filters generated from the scaling filter. First, we note that if h_n, g_n have order M, and are defined over $0\leq n\leq M$, then the dilation equations imply that $\phi(t)$ and $\psi(t)$ will have compact support over $0\leq t\leq M$. Thus, we may evaluate the rth iterate $\phi^{(r)}(t)$ at the equally-spaced points, $t = 2^{-r}n$, for $0\leq n\leq M2^r$, spanning the support interval. To this end, we define the discrete-time signals of the sampled $\phi^{(r)}(t)$:

$$f^{(r)}(n)= 2^{-r/2}\phi^{(r)}(2^{-r}n)\tag{10.2.18}$$

where $2^{-r/2}$ is a convenient normalization factor. It follows then from Eq. (10.2.17) that

$$2^{-(r+1)/2}\phi^{(r+1)}(2^{-(r+1)}n)= \sum_m h_m\, 2^{-r/2}\phi^{(r)}(2^{-r}n-m)\,,\quad\text{or,}$$

$$f^{(r+1)}(n)= \sum_m h_m f^{(r)}(n-2^r m)= \sum_k h^{[r]}(k)f^{(r)}(n-k)\tag{10.2.19}$$

where we defined the à trous filter corresponding to the interpolation factor 2^r by

$$h^{[r]}(k)= \sum_m h_m\delta(k-2^r m)\tag{10.2.20}$$

which is the original filter h_n with (2^r-1) zeros inserted between the h_n samples, so that its z-transform and frequency response are $H^{[r]}(z)= H(z^{2^r})$ and $H^{[r]}(\omega)= H(2^r\omega)$.

Thus, Eq. (10.2.19) can be interpreted as the convolution of the rth iterate with the rth à trous filter. The recursion can be iterated for $r = 0,1,2,\dots,J$, for sufficiently large J (typically, $J = 10$ works well.) The MATLAB function casc implements this algorithm:

```
[phi,psi,t] = casc(h,J,phi0);        % cascade algorithm
```

where t is the vector of final evaluation points $t = 2^{-J}n$, $0\leq n\leq M2^J$. For example, the Daubechies D_2 functions $\phi(t),\psi(t)$ shown in Fig. 10.1.1 can be computed and plotted by the following code:

```
h = daub(2); J = 10; phi0 = 1;

[phi,psi,t] = casc(h,J,phi0);

figure;  plot(t,phi, t,psi,'--');
```

[†] "a trous" means "with holes" in French. The filters are similar to the comb "seasonal" filters of Chap. 9.

The scaling function output `phi` is normalized to unit L_2-norm, and the wavelet output `psi` is commensurately normalized. The following MATLAB code fragment from `casc` illustrates the construction method:

```
phi0=1;
for r=0:J-1,
   phi = conv(phi, upr(h,r));
end
```

where the function `upr` constructs the à trous filter $h^{[r]}(k)$ by upsampling h_n by a factor of 2^r. This function can also be implemented using the MATLAB's built-in function `upsample`. For example, if $\mathbf{h} = [h_0, h_1, h_2, h_3]$, then for $r = 2$, the à trous filter will be,

$$\mathbf{h}^{[r]} = \text{upr}(\mathbf{h}, r) = \text{upsample}(\mathbf{h}, 2^r) = [h_0, 0, 0, 0, h_1, 0, 0, 0, h_2, 0, 0, 0, h_3, 0, 0, 0]$$

10.3 Wavelet Filter Properties

The scaling and wavelet filters h_n, g_n must satisfy certain necessary constraints which are a consequence of the orthogonality of the scaling and wavelet basis functions. Using the property $(\phi_{j+1,n},\ \phi_{j+1,m}) = \delta_{nm}$, it follows from Eq. (10.2.5) that,

$$(\phi_{j0}, \phi_{jk}) = \Big(\sum_n h_n \phi_{j+1,n},\ \sum_m h_{m-2k}\phi_{j+1,m}\Big) = \sum_{n,m} h_n h_{m-2k}(\phi_{j+1,n},\ \phi_{j+1,m}) = \sum_n h_n h_{n-2k}$$

Similarly, we find,

$$\begin{aligned}
(\psi_{j0}, \psi_{jk}) &= \Big(\sum_n g_n \phi_{j+1,n},\ \sum_m g_{m-2k}\phi_{j+1,m}\Big) = \sum_n g_n g_{n-2k} \\
(\phi_{j0}, \psi_{jk}) &= \Big(\sum_n h_n \phi_{j+1,n},\ \sum_m g_{m-2k}\phi_{j+1,m}\Big) = \sum_n h_n g_{n-2k}
\end{aligned}$$

But $(\phi_{j0}, \phi_{jk}) = (\psi_{j0}, \psi_{jk}) = \delta_k$ and $(\phi_{j0}, \psi_{jk}) = 0$, therefore h_n, g_n must satisfy the orthogonality properties:

$$\boxed{\begin{aligned}
\sum_n h_n h_{n-2k} &= \delta_k \\
\sum_n g_n g_{n-2k} &= \delta_k \\
\sum_n h_n g_{n-2k} &= 0
\end{aligned}} \tag{10.3.1}$$

These may also be expressed in the frequency domain. We will make use of the following cross-correlation identities that are valid for any two filters h_n, g_n and their frequency responses $H(\omega), G(\omega)$:

$$\begin{aligned}
\sum_n h_n g_{n-k} \quad &\Leftrightarrow \quad H(\omega)G^*(\omega) \\
(-1)^k \sum_n h_n g_{n-k} \quad &\Leftrightarrow \quad H(\omega+\pi)G^*(\omega+\pi) \\
\Big[1 + (-1)^k\Big] \sum_n h_n g_{n-k} \quad &\Leftrightarrow \quad H(\omega)G^*(\omega) + H(\omega+\pi)G^*(\omega+\pi)
\end{aligned} \tag{10.3.2}$$

where the second follows from the "modulation" property of Fourier transforms, and the third, by adding the first two. We note next that Eqs. (10.3.1) can be written in the following equivalent manner obtained by replacing $2k$ by any k, even or odd:

$$\begin{aligned}
\big[1 + (-1)^k\big] \sum_n h_n h_{n-k} &= 2\delta_k \\
\big[1 + (-1)^k\big] \sum_n g_n g_{n-k} &= 2\delta_k \\
\big[1 + (-1)^k\big] \sum_n h_n g_{n-k} &= 0
\end{aligned} \tag{10.3.3}$$

Taking the Fourier transforms of both sides of (10.3.3) and using the transform properties (10.3.2), we obtain the frequency-domain equivalent conditions to Eqs. (10.3.1):

$$\boxed{\begin{aligned}
&|H(\omega)|^2 + |H(\omega+\pi)|^2 = 2 \\
&|G(\omega)|^2 + |G(\omega+\pi)|^2 = 2 \\
&H(\omega)G^*(\omega) + H(\omega+\pi)G^*(\omega+\pi) = 0
\end{aligned}} \tag{10.3.4}$$

The conjugate mirror filter choice (10.2.11) for $G(\omega)$ automatically satisfies the third of Eqs. (10.3.4). Indeed, using the 2π-periodicity of $H(\omega)$, we have,

$$\begin{aligned}
G^*(\omega) &= e^{jM(\omega+\pi)} H(\omega+\pi) \\
G^*(\omega+\pi) &= e^{jM(\omega+2\pi)} H(\omega+2\pi) = e^{jM\omega} H(\omega)
\end{aligned}$$

so that,

$$H(\omega)G^*(\omega) + H(\omega+\pi)G^*(\omega+\pi) = e^{jM\omega} H(\omega)H(\omega+\pi)\Big[e^{jM\pi} + 1\Big] = 0$$

where $e^{jM\pi} = -1$, because M was assumed to be odd. With this choice of $G(\omega)$, the first of (10.3.4) can be written in the following form, which will be used later on to derive the undecimated wavelet transform:

$$\boxed{\frac{1}{2}\big[H^*(\omega)H(\omega) + G^*(\omega)G(\omega)\big] = 1} \tag{10.3.5}$$

Setting $\omega = 0$ in the first of Eqs. (10.3.4), and using the DC gain constraint $H(0) = \sqrt{2}$, we find immediately that $H(\pi) = 0$, that is, the scaling filter must have a zero at the Nyquist frequency $\omega = \pi$. Since

$$H(\pi) = \sum_n (-1)^n h_n = \sum_{n=\text{even}} h_n - \sum_{n=\text{odd}} h_n\,,$$

it follows in conjunction with the DC condition that:

$$\sum_{n=\text{even}} h_n = \sum_{n=\text{odd}} h_n = \frac{1}{\sqrt{2}} \tag{10.3.6}$$

The correlation constraints and the DC gain condition,

$$\sum_n h_n h_{n-2k} = \delta_k\,, \quad \sum_n h_n = \sqrt{2}, \tag{10.3.7}$$

provide only $N/2+1$ equations, where N is the (even) length of the filter h_n. Therefore, one has $N/2-1$ additional degrees of freedom to specify the scaling filters uniquely. For example, Daubechies' minimum-phase D_K filters have length $N=2K$ and K zeros at Nyquist. These zeros translate into K equivalent moment constraints, or derivative flatness constraints at Nyquist:

$$\sum_{n=0}^{N-1}(-1)^n n^i h_n = 0 \quad\Leftrightarrow\quad \frac{d^i H(\omega)}{d\omega^i}\bigg|_{\omega=\pi} = 0\,, \quad i = 0,1,\dots,K-1 \tag{10.3.8}$$

The $i=0$ case is already a consequence of the correlation constraint, therefore, this leaves $K-1$ additional conditions, which together with the $K+1$ equations (10.3.7), determines the $N=2K$ coefficients h_n uniquely. The construction method may be found in [665]. As an example, we work out the three cases D_1, D_2, D_3 explicitly. The Haar D_1 case corresponds to $K=1$ or $N=2K=2$, so that $\mathbf{h}=[h_0,h_1]$ must satisfy:

$$h_0^2 + h_1^2 = 1\,, \quad h_0 + h_1 = \sqrt{2} \tag{10.3.9}$$

with (lowpass) solution $h_0 = h_1 = 1/\sqrt{2}$. For the Daubechies D_2 case, we have $K=2$ and $N=2K=4$, so that $\mathbf{h}=[h_0,h_1,h_2,h_3]$ must satisfy,

$$\begin{aligned} &h_0^2 + h_1^2 + h_2^2 + h_3^2 = 1 \\ &h_0 + h_2 = \frac{1}{\sqrt{2}}\,, \quad h_1 + h_3 = \frac{1}{\sqrt{2}} \\ &-h_1 + 2h_2 - 3h_3 = 0 \end{aligned} \tag{10.3.10}$$

where the third is the Nyquist moment constraint with $i=1$, and the middle two are equivalent to the DC gain and the $h_0h_2 + h_1h_3 = 0$ correlation constraint; indeed, this follows from the identity:

$$\begin{aligned} &\Big(h_0 + h_2 - \frac{1}{\sqrt{2}}\Big)^2 + \Big(h_1 + h_3 - \frac{1}{\sqrt{2}}\Big)^2 = \\ &\quad = 1 + (h_0^2 + h_1^2 + h_2^2 + h_3^2) - \sqrt{2}(h_0 + h_1 + h_2 + h_3) + 2(h_0h_2 + h_1h_3) \end{aligned}$$

Solving the three linear ones for h_1, h_2, h_3 in terms of h_0 and inserting them in the first one, we obtain the quadratic equation for h_0, with solutions:

$$4h_0^2 - \sqrt{2}\,h_0 - \frac{1}{4} = 0 \quad\Rightarrow\quad h_0 = \frac{1 \pm \sqrt{3}}{4\sqrt{2}}$$

The "+" choice leads to the following minimum-phase filter (the "−" choice leads to the reverse of that, which has maximum phase):

$$\begin{aligned} \mathbf{h} = [h_0,h_1,h_2,h_3] &= \frac{1}{4\sqrt{2}}\big[1+\sqrt{3}\,,\ 3+\sqrt{3}\,,\ 3-\sqrt{3}\,,\ 1-\sqrt{3}\big] \\ &= [0.4830\,,\ 0.8365\,,\ 0.2241\,,\ -0.1294] \end{aligned} \tag{10.3.11}$$

The corresponding transfer function $H(z)$ has a double zero at Nyquist $z=-1$ and one inside the unit circle at $z = 2-\sqrt{3}$. In fact, $H(z)$ factors as follows:

$$H(z) = h_0(1+z^{-1})^2\big(1-(2-\sqrt{3})z^{-1}\big)$$

For the D_3 case corresponding to $K=3$, we have the following two quadratic equations and four linear ones that must be satisfied by the filter $\mathbf{h} = [h_0,h_1,h_2,h_3,h_4,h_5]$:

$$\begin{aligned} &h_0^2 + h_1^2 + h_2^2 + h_3^2 + h_4^2 + h_5^2 = 1\,, \quad h_0h_4 + h_1h_5 = 0 \\ &h_0 + h_2 + h_4 = 1/\sqrt{2}\,, \quad h_1 + h_3 + h_5 = 1/\sqrt{2} \\ &-h_1 + 2h_2 - 3h_3 + 4h_4 - 5h_5 = 0 \\ &-h_1 + 2^2h_2 - 3^2h_3 + 4^2h_4 - 5^2h_5 = 0 \end{aligned} \tag{10.3.12}$$

where the last two correspond to the values $i=1,2$ in (10.3.8), and we have omitted the correlation constraint $h_0h_2 + h_1h_3 + h_2h_4 + h_3h_5 = 0$ as it is obtainable from Eqs. (10.3.12). Solving the linear ones for h_2,h_3,h_4,h_5 in terms of h_0,h_1, we find,

$$\begin{aligned} h_2 &= -4h_0 + 2h_1 + \sqrt{2}/8 \\ h_3 &= -2h_0 + 3\sqrt{2}/8 \\ h_4 &= 3h_0 - 2h_1 + 3\sqrt{2}/8 \\ h_5 &= 2h_0 - h_1 + \sqrt{2}/8 \end{aligned} \tag{10.3.13}$$

Inserting these into the first two of Eqs. (10.3.12), we obtain the quadratic system:

$$\begin{aligned} &34h_0^2 - (32h_1 - \sqrt{2}/4)h_0 + 10h_1^2 - 5\sqrt{2}h_1/4 - 3/8 = 0 \\ &h_1^2 - \sqrt{2}h_1/8 - 3h_0^2 - 3\sqrt{2}h_0/8 = 0 \end{aligned} \tag{10.3.14}$$

Solving the second for h_1 in terms of h_0, we find:

$$h_1 = \frac{1}{16}\Big[\sqrt{2} + \sqrt{768h_0^2 + 96\sqrt{2}h_0 + 2}\Big] \tag{10.3.15}$$

and inserting this into the first of (10.3.14), we obtain:

$$64h_0^2 + 2\sqrt{2}h_0 - 2h_0\sqrt{768h_0^2 + 96\sqrt{2}h_0 + 2} - \frac{3}{8} = 0$$

or, the equivalent quartic equation:

$$1024h_0^4 - 128\sqrt{2}h_0^3 - 48h_0^2 - \frac{3\sqrt{2}}{2}h_0 + \frac{9}{64} = 0 \tag{10.3.16}$$

which has two real solutions and two complex-conjugate ones. Of the real solutions, the one that leads to a minimum-phase filter $\mathbf{h}$ is

$$h_0 = \frac{\sqrt{2}}{32}\Big[1 + \sqrt{10} + \sqrt{5 + 2\sqrt{10}}\Big] \tag{10.3.17}$$

With this solution for h_0, Eqs. (10.3.15) and (10.3.13) lead to the desired minimum-phase filter. Its transfer function $H(z)$ factors as:

$$H(z)=h_0+h_1z^{-1}+h_2z^{-2}+h_3z^{-3}+h_4z^{-4}+h_5z^{-5}=h_0(1+z^{-1})^3(1-z_1z^{-1})(1-z_1^*z^{-1})$$

where z_1 is the following zero lying inside the unit circle:

$$z_1=\frac{\sqrt{10}-1+j\sqrt{2\sqrt{10}-5}}{1+\sqrt{10}+\sqrt{5+2\sqrt{10}}}\quad\Rightarrow\quad |z_1|=\frac{\sqrt{6}}{1+\sqrt{10}+\sqrt{5+2\sqrt{10}}}=0.3254 \tag{10.3.18}$$

Finally, we mention that the K flatness constraints (10.3.8) at $\omega=\pi$ for $H(\omega)$ are equivalent to K flatness constraints for the wavelet filter $G(\omega)$ at DC, that is,

$$\left.\frac{d^iG(\omega)}{d\omega^i}\right|_{\omega=0}=0\,,\quad i=0,1,\dots,K-1 \tag{10.3.19}$$

In turn, these are equivalent to the K vanishing moment constraints for the wavelet function $\psi(t)$, that is,

$$\int_{-\infty}^{\infty}t^i\psi(t)dt=0\,,\quad i=0,1,\dots,K-1 \tag{10.3.20}$$

The equivalence between (10.3.19) and (10.3.20) is easily established by differentiating the dilation equation (10.2.14) for $\Psi(\omega)$ with respect to ω and setting $\omega=0$.

Because the D_K filters have minimum phase by construction, their energy is concentrated at earlier times and their shape is very asymmetric. Daubechies' other two families of scaling and wavelet filters, the "least asymmetric" symmlets, and the coiflets, have a more symmetric shape. They are discussed in detail in [665].

Another consequence of the orthonormality of the ϕ and ψ bases can be stated in terms of the Fourier transforms $\Phi(\omega)$ and $\Psi(\omega)$ as identities in ω:

$$\begin{aligned}&\sum_{m=-\infty}^{\infty}|\Phi(\omega+2\pi m)|^2=\sum_{m=-\infty}^{\infty}|\Psi(\omega+2\pi m)|^2=1\\&\sum_{m=-\infty}^{\infty}\Phi(\omega+2\pi m)\Psi^*(\omega+2\pi m)=0\end{aligned} \tag{10.3.21}$$

These follow by applying Parseval's identity to the cross-correlation inner products of the ϕ and ψ bases. For example, we have,

$$\begin{aligned}\delta_k&=(\phi_{j0},\phi_{jk})=(\phi_{00},\phi_{0k})=\int_{-\infty}^{\infty}\phi(t)\phi(t-k)dt=\frac{1}{2\pi}\int_{-\infty}^{\infty}|\Phi(\omega)|^2e^{j\omega k}d\omega\\&=\frac{1}{2\pi}\int_{-\pi}^{\pi}\left[\sum_{m=-\infty}^{\infty}|\Phi(\omega+2\pi m)|^2\right]e^{j\omega k}d\omega\end{aligned}$$

where the last expression was obtained by noting that because k is an integer, the exponential $e^{j\omega k}$ is periodic in ω with period 2π, which allowed us to fold the infinite integration range into the $[-\pi,\pi]$ range. But this result is simply the inverse DTFT of the first of Eqs. (10.3.21). The other results are shown in a similar fashion using the inner products $(\psi_{j0},\psi_{jk})=\delta_k$ and $(\phi_{j0},\psi_{jk})=0$.

It can be easily argued from Eqs. (10.2.16) that $\Phi(2\pi m)=0$ for all non-zero integers m. Indeed, setting $m=2^p(2q+1)$ for some integers $p\geq 0, q\geq 0$, it follows that after p iterations, an H-factor will appear such that $H\big((2q+1)\pi\big)=H(\pi)=0$. Setting $\omega=0$ in the first of Eqs. (10.3.21) and using this property, it follows that $|\Phi(0)|^2=1$. Thus, up to a sign, we may set:

$$\Phi(0)=\int_{-\infty}^{\infty}\phi(t)dt=1 \tag{10.3.22}$$

10.4 Multiresolution and Filter Banks

We saw in Eq. (10.1.12) that a signal belonging to a higher-resolution subspace can be expanded in terms of its lower-resolution components. If J and J_0 are the highest and lowest resolutions of interest, then for a signal $f(t)\in V_J$, the multiresolution expansion will have the form:

$$f(t)=\sum_n c_{Jn}\phi_{Jn}(t)=\sum_k c_{J_0k}\phi_{J_0k}(t)+\sum_{j=J_0}^{J-1}\sum_k d_{jk}\psi_{jk}(t) \tag{10.4.1}$$

with the various terms corresponding to the direct-sum decomposition:

$$V_J=V_{J_0}\oplus\big(W_{J_0}\oplus W_{J_0+1}\oplus\cdots\oplus W_{J-1}\big) \tag{10.4.2}$$

The choice of J,J_0 is dictated by the application at hand. Typically, we start with a signal $f(t)$ sampled at $N=2^J$ samples that are equally-spaced over the signal's duration. The duration interval can always be normalized to be $0\leq t\leq 1$ so that the sample spacing is 2^{-J}. The lowest level is $J_0=0$ corresponding to sample spacing $2^{-J_0}=1$, that is, one sample in the interval $0\leq t\leq 1$. One does not need to choose $J_0=0$; any value $0\leq J_0\leq J-1$ could be used.

The lower-level expansion coefficients $\{c_{J_0k};d_{jk},J_0\leq j\leq J-1\}$ can be computed from those of the highest level c_{Jn} by *Mallat's multiresolution algorithm* [721], which establishes a connection between multiresolution analysis and filter banks.

The algorithm successively computes the coefficients at each level from those of the level just above. It is based on establishing the relationship between the expansion coefficients for the decomposition $V_{j+1}=V_j\oplus W_j$ and iterating it over $J_0\leq j\leq J-1$. An arbitrary element $f(t)$ of V_{j+1} can be expanded in two ways:

$$f(t)=\underbrace{\sum_n c_{j+1,n}\phi_{j+1,n}(t)}_{V_{j+1}}=\underbrace{\sum_k c_{jk}\phi_{jk}(t)}_{V_j}+\underbrace{\sum_k d_{jk}\psi_{jk}(t)}_{W_j} \tag{10.4.3}$$

The right-hand side coefficients are:

$$\begin{aligned}c_{jk}&=(f,\phi_{jk})=\left(\sum_n c_{j+1,n}\phi_{j+1,n},\,\phi_{jk}\right)=\sum_n c_{j+1,n}(\phi_{j+1,n},\phi_{jk})\\d_{jk}&=(f,\psi_{jk})=\left(\sum_n c_{j+1,n}\phi_{j+1,n},\,\psi_{jk}\right)=\sum_n c_{j+1,n}(\phi_{j+1,n},\psi_{jk})\end{aligned}$$

which become, using Eq. (10.2.6),

$$\boxed{\begin{aligned} c_{jk} &= \sum_n h_{n-2k}c_{j+1,n} \\ d_{jk} &= \sum_n g_{n-2k}c_{j+1,n} \end{aligned}} \qquad \text{(analysis)} \qquad (10.4.4)$$

for $j = J-1, J-2, \dots, J_0$, initialized at $c_{Jn} = f(t_n)$, $n = 0, 1, \dots, 2^J-1$, with $t_n = n2^{-J}$, that is, the 2^J samples of $f(t)$ in the interval $0 \le t \le 1$. Conversely, the coefficients $c_{j+1,n}$ can be reconstructed from c_{jk}, d_{jk}:

$$\begin{aligned} c_{j+1,n} = (f, \phi_{j+1,n}) &= \left(\sum_k c_{jk}\phi_{jk} + \sum_k d_{jk}\psi_{jk}\,,\ \phi_{j+1,n}\right) \\ &= \sum_k c_{jk}(\phi_{jk}, \phi_{j+1,n}) + \sum_k d_{jk}(\psi_{jk}, \phi_{j+1,n}) \end{aligned}$$

or,

$$\boxed{c_{j+1,n} = \sum_k h_{n-2k}c_{jk} + \sum_k g_{n-2k}d_{jk}} \qquad \text{(synthesis)} \qquad (10.4.5)$$

for $j = J_0, J_0+1, \dots, J-1$. To see the filter bank interpretation of these results, let us define the *time-reversed* filters $\bar{h}_n = h_{-n}$ and $\bar{g}_n = g_{-n}$, and the downsampling and upsampling operations by a factor of two [670]:

$$\begin{aligned} y_{\text{down}}(n) &= x(2n) \\ y_{\text{up}}(n) &= \sum_k x(k)\delta(n-2k) = \begin{cases} x(k), & \text{if } n = 2k \\ 0, & \text{otherwise} \end{cases} \end{aligned} \qquad (10.4.6)$$

and pictorially,

$$x(n) \xrightarrow{f_s} \boxed{2\downarrow} \xrightarrow{f_s/2} y_{\text{down}}(n) \qquad\qquad x(n) \xrightarrow{f_s} \boxed{2\uparrow} \xrightarrow{2f_s} y_{\text{up}}(n)$$

The downsampling operation decreases the sampling rate by a factor of two by keeping only the even-index samples of the input. The upsampling operation increases the sampling rate by a factor of two by inserting a zero between successive input samples. It is the same as the "à trous" operation for filters that we encountered earlier.

With these definitions, the analysis algorithm (10.4.4) is seen to be equivalent to convolving with the time-reversed filters, followed by downsampling. Symbolically,

$$\begin{aligned} c_{jk} &= \sum_n \bar{h}_{2k-n}c_{j+1,n} = (\bar{h} * c_{j+1})(2k) \\ d_{jk} &= \sum_n \bar{g}_{2k-n}c_{j+1,n} = (\bar{g} * c_{j+1})(2k) \end{aligned} \quad \Rightarrow \quad \boxed{\begin{aligned} \mathbf{c}_j &= (\bar{\mathbf{h}} * \mathbf{c}_{j+1})_{\text{down}} \\ \mathbf{d}_j &= (\bar{\mathbf{g}} * \mathbf{c}_{j+1})_{\text{down}} \end{aligned}} \qquad (10.4.7)$$

Similarly, the synthesis algorithm (10.4.5) is equivalent to upsampling the signals c_{jk} and d_{jk} by two and then filtering them through h_n, g_n,

$$c_{j+1,n} = \sum_k h_{n-2k}c_{jk} + \sum_k g_{n-2k}d_{jk} = \sum_m h_{n-m}c_{jm}^{\text{up}} + \sum_m g_{n-m}d_{jm}^{\text{up}} \qquad (10.4.8)$$

where $c_{jm}^{\text{up}} = \sum_k c_{jk}\delta(m-2k)$. Symbolically,

$$\boxed{\mathbf{c}_{j+1} = \mathbf{h} * \mathbf{c}_j^{\text{up}} + \mathbf{g} * \mathbf{d}_j^{\text{up}}} \qquad (10.4.9)$$

Fig. 10.4.1 shows a block diagram realization of the analysis and synthesis equations (10.4.7) and (10.4.9) in terms of a so-called *tree-structured iterated filter bank*.

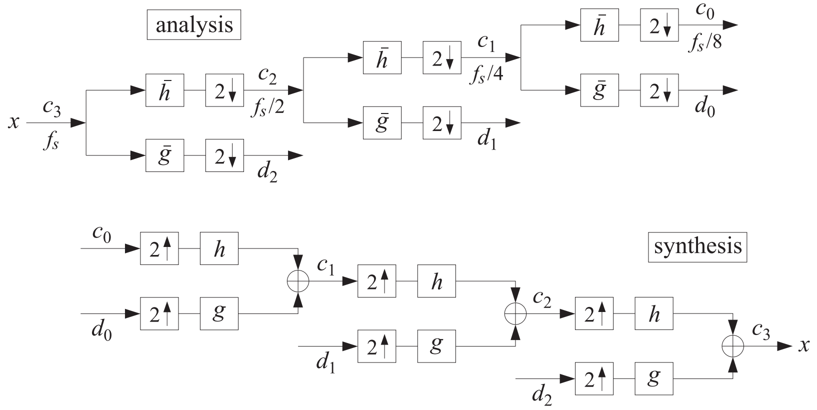


Fig. 10.4.1 Analysis and synthesis filter bank.

In the figure, we used $J = 3$ and $J_0 = 0$. Each stage of the analysis bank produces the coefficients at the next coarser level. Similarly, the synthesis bank starts with the coarsest level and successively reconstructs the higher levels.

The time-reversed filters $\bar{h}_n, \bar{g}_n$ are still lowpass and highpass, indeed, their frequency responses are $\bar{H}(\omega) = H^*(\omega)$ and $\bar{G}(\omega) = G^*(\omega)$. Therefore, at the first analysis stage, the input signal c_3 is split into the low- and high-frequency parts c_2, d_2 representing, respectively, a smoother trend and a more irregular detail. At the second stage, the smooth trend c_2 is split again into a low and high frequency part, c_1, d_1, and so on. The subband frequency operation of the filter bank can be understood by looking at the spectra of the signals at the successive output stages.

Because successive stages operate at different sampling rates, it is best to characterize the spectra using a common frequency axis, for example, the physical frequency f. The spectrum of a discrete-time signal $x(n)$ sampled at a rate f_s is defined by,

$$X(f) = \sum_n x(n)e^{-j\omega n} = \sum_n x(n)e^{-2\pi jfn/f_s} \qquad (10.4.10)$$

where $\omega = 2\pi f/f_s$ is the digital frequency in radians/sample. We will use the notation $X(f, f_s)$ whenever it is necessary to indicate the dependence on f_s explicitly.

Just like the sampling of a continuous-time signal causes the periodic replication of its spectrum at multiples of the sampling rate, the operation of downsampling causes the periodic replication of the input spectrum at multiples of the downsampled rate. It is a general result [30] that for a downsampling ratio by a factor L, and input and output rates of f_s and $f_s^{\text{down}} = f_s/L$, the downsampled signal $y_{\text{down}}(n) = x(nL)$ will have the following replicated spectrum at multiples of f_s^{down}:

$$Y_{\text{down}}(f) = \frac{1}{L}\sum_{m=0}^{L-1} X(f - mf_s^{\text{down}}) \tag{10.4.11}$$

where according to (10.4.10),

$$Y_{\text{down}}(f) = \sum_n y_{\text{down}}(n)\,e^{-2\pi jfn/f_s^{\text{down}}} = \sum_n x(nL)\,e^{-2\pi jfnL/f_s} \tag{10.4.12}$$

In particular, for downsampling by $L = 2$, we have $f_s^{\text{down}} = f_s/2$ and

$$Y_{\text{down}}(f) = \frac{1}{2}\Big[X(f) + X(f - f_s^{\text{down}})\Big] = \sum_n x(2n)\,e^{-2\pi jf2n/f_s} \tag{10.4.13}$$

If f_s is the sampling rate at the input stage for the signal c_3 of the analysis bank, then the rates for the signals c_2, c_1, c_0 will be $f_s/2, f_s/4, f_s/8$, respectively. Fig. 10.4.2 shows the corresponding spectra, including the effect of filtering and downsampling.

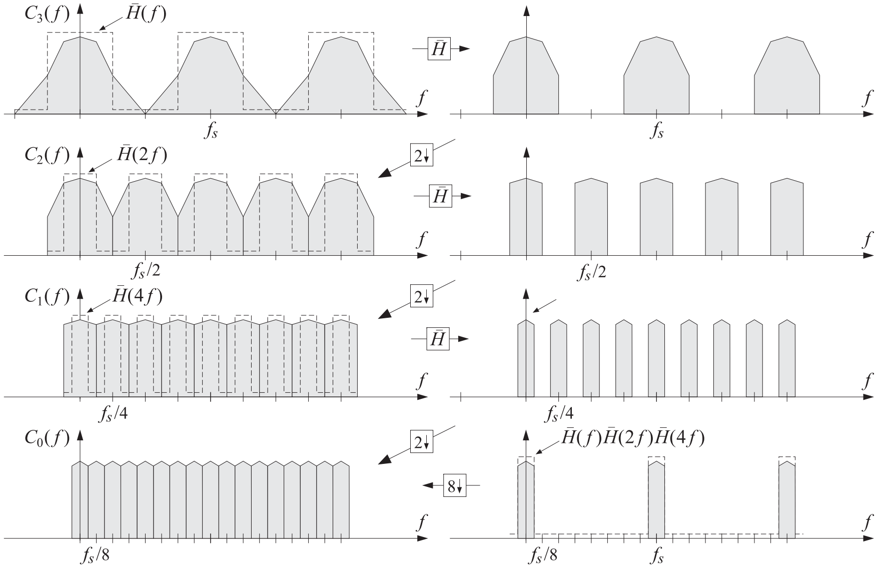


Fig. 10.4.2 Spectra of the signals c_3, c_2, c_1, c_0 at successive stages of the analysis bank of Fig. 10.4.1.

For clarity, we took $\bar{H}(f)$ to be an ideal lowpass filter with cutoff frequency equal to half the Nyquist frequency, that is, $f_s/4$. Starting at the top left with the input spectrum $C_3(f)$, which is replicated at multiples of f_s, the first lowpass filtering operation produces the spectrum at the upper right. According to Eq. (10.4.13), downsampling will replicate this spectrum at multiples of $f_s^{\text{down}} = f_s/2$, thereby filling the gaps created by the ideal filter, and resulting in the spectrum $C_2(f)$ shown on the left graph of the second row. The sampling rate at that stage is now $f_s/2$.

The second lowpass filtering operation of the signal c_2 indicated on Fig. 10.4.1 will be by the filter $\bar{H}(f, f_s/2)$ which is equal to $\bar{H}(2f, f_s)$ if referred to the original sampling rate f_s; indeed, we have,

$$\bar{H}(f, f_s/2) = \sum_n \bar{h}_n e^{-2\pi jfn/(f_s/2)} = \sum_n \bar{h}_n e^{-2\pi jf2n/f_s} = \bar{H}(2f, f_s) \tag{10.4.14}$$

The filter $\bar{H}(2f, f_s)$ is the twice-compressed version of $\bar{H}(f, f_s)$, and still has an ideal shape but with cutoff frequency $f_s/8$. The result of the second filtering operation is shown on the right graph of the second row. The lowpass-filtered replicas are at multiples of $f_s/2$, and after the next downsampling operation, they will be replicated at multiples of $f_s/4$ resulting in the spectrum $C_1(f)$ of the signal c_1 shown on the left of the third row. At the new sampling rate $f_s/4$, the third-stage lowpass filter will be:

$$\bar{H}(f, f_s/4) = \bar{H}(2f, f_s/2) = \bar{H}(4f, f_s) \tag{10.4.15}$$

which is the four-times compressed version of that at rate f_s, or twice-compressed of that of the previous stage. Its cutoff is now at $f_s/16$. The result of filtering by $\bar{H}(4f, f_s)$ is shown on the right of the third row, and its downsampled version replicated at multiples of $f_s/8$ is shown on the bottom left as the spectrum $C_0(f)$.

Thus, the output spectra $C_2(f), C_1(f), C_0(f)$ capture the frequency content of the original signal in the corresponding successive subbands, each subband having half the passband of the previous one (often referred to as an octave filter bank.)

The bottom-right graph shows an equivalent way of obtaining the same final output $C_0(f)$, namely, by first filtering by the combined filter,

$$\bar{H}(f, f_s)\,\bar{H}(2f, f_s)\,\bar{H}(4f, f_s) = \bar{H}(f, f_s)\,\bar{H}(f, f_s/2)\,\bar{H}(f, f_s/4)$$

running at the original rate f_s, and then dropping the rate all at once by a factor of $2^3 = 8$, which will cause a replication at multiples of $f_s/8$. This point of view is justified by applying the standard multirate identity depicted below [670]:

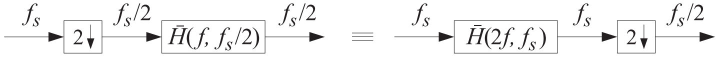


Fig. 10.4.3 shows the successive application of this identity to the three stages of Fig. 10.4.1 until all the downsamplers are pushed to the right-most end and all the filters to the left-most end. The corresponding sampling rates are indicated at the outputs of the downsamplers.

For non-ideal filters $\bar{H}(f), \bar{G}(f)$, such as the scaling and wavelet filters, the downsampling replication property (10.4.13) will cause aliasing. However, because of Eq. (10.3.4), the filter bank satisfies the so-called *pefect reconstruction* property, which allows the aliasing to be canceled at the reconstruction, synthesis, stage.

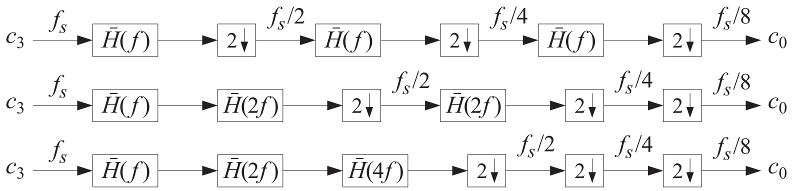


Fig. 10.4.3 Equivalent realizations of the lowpass portion of the analysis bank of Fig. 10.4.1.

10.5 Discrete Wavelet Transform

We summarize the analysis and synthesis algorithms:

$$\boxed{\begin{aligned}\mathbf{c}_{j-1} &= \big(\bar{\mathbf{h}} * \mathbf{c}_j\big)_{\text{down}}\\ \mathbf{d}_{j-1} &= \big(\bar{\mathbf{g}} * \mathbf{c}_j\big)_{\text{down}}\end{aligned}} \qquad j = J, J-1, \dots, J_0+1 \tag{10.5.1}$$

$$\boxed{\mathbf{c}_j = \mathbf{h} * \mathbf{c}_{j-1}^{\text{up}} + \mathbf{g} * \mathbf{d}_{j-1}^{\text{up}}} \qquad j = J_0+1, J_0+2, \dots, J \tag{10.5.2}$$

The discrete wavelet transform (DWT) consists of the coefficients generated by the analysis algorithm. The DWT can be defined for each resolution level. Starting with an input signal vector $\mathbf{x} = [x_0, x_1, \dots, x_{N-1}]^T$, where $N = 2^J$, the DWTs at successive stages are defined as the following sets of coefficients:

$$\begin{array}{rll} \mathbf{x} = \mathbf{c}_J & \to [\mathbf{c}_{J-1}, \mathbf{d}_{J-1}], & \text{(level } J-1) \\ & \to [\mathbf{c}_{J-2}, \mathbf{d}_{J-2}, \mathbf{d}_{J-1}], & \text{(level } J-2) \\ & \to [\mathbf{c}_{J-3}, \mathbf{d}_{J-3}, \mathbf{d}_{J-2}, \mathbf{d}_{J-1}], & \text{(level } J-3) \\ & \quad\vdots & \\ & \to [\mathbf{c}_{J_0}, \mathbf{d}_{J_0}, \mathbf{d}_{J_0+1}, \dots, \mathbf{d}_{J-1}], & \text{(level } J_0) \end{array} \tag{10.5.3}$$

Starting with the coefficients at any level, the inverse discrete wavelet transform (IDWT) applies the synthesis algorithm to reconstruct the original signal **x**.

In practice, there are as many variants of the DWT as there are ways to implement the filtering operations in (10.5.1)–(10.5.2), such as deciding on how to deal with the filter transients (the edge effects), realizing convolution in a matrix form, periodizing or symmetrizing the signals or not, and so on.

There exist several commercial implementations in MATLAB, Mathematica, Maple, and S+, incorporating the many variants, as well as several freely available packages in MATLAB, C++, and R [834–848].

In this section, we consider only the periodized version implemented both in matrix form and in filtering form using circular convolutions. Given a (possibly infinite) signal $x(n)$, we define its "modulo-N reduction" [29] as its periodic extension with period N:

$$\tilde{x}(n) = \sum_{p=-\infty}^{\infty} x(n+pN) \tag{10.5.4}$$

The signal $\tilde{x}(n)$ is periodic with period N, and therefore, we only need to know it over one period, $0 \le n \le N-1$. It is characterized by the property that it has the same N-point DFT as the signal $x(n)$, that is,

$$X(\omega_k) = \sum_{n=-\infty}^{\infty} x(n)e^{-j\omega_k n} = \sum_{n=0}^{N-1} \tilde{x}(n)e^{-j\omega_k n} \tag{10.5.5}$$

where ω_k are the DFT frequencies $\omega_k = 2\pi k/N$, $k = 0, 1, \dots, N-1$. The signal $\tilde{x}(n)$ can be visualized as dividing the original signal $x(n)$ into contiguous blocks of length N, then aligning them in time, and adding them up. This operation is referred to as "mod-N wrapping" and is depicted in Fig. 10.5.1 for a signal $x(n)$ of length $4N$.

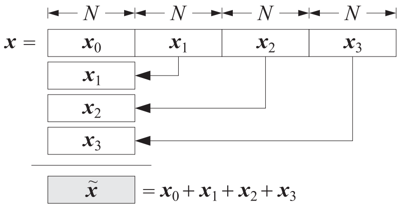
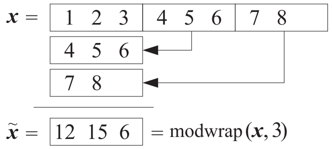


Fig. 10.5.1 Modulo-N reduction or wrapping.

The MATLAB function `modwrap` implements this operation. Its argument can be a row or column vector, or a matrix. For the matrix case, it wraps each column modulo N:

```
Y = modwrap(X,N);          % mod-N reduction of a matrix
```

For the example, we have for the signal $\mathbf{x} = [1, 2, 3, 4, 5, 6, 7, 8]$ and $N = 3$,



Circular convolution is defined as the modulo-N reduction of ordinary linear convolution, that is,

$$\mathbf{y} = \mathbf{h} * \mathbf{x} \quad \Rightarrow \quad \mathbf{y}_{\text{circ}} = \tilde{\mathbf{y}} = \widetilde{\mathbf{h} * \mathbf{x}} \tag{10.5.6}$$

or more explicitly,

$$y(n) = \sum_m h(m)x(n-m) \quad \Rightarrow \quad \tilde{y}(n) = \sum_p y(n+pN)$$

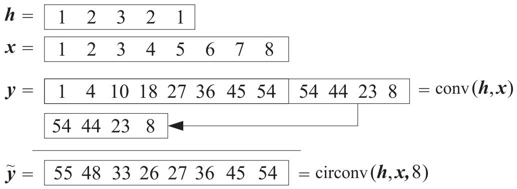
Its MATLAB implementation is straightforward with the help of the function `modwrap`, for example,

```
y = modwrap(conv(h,x), N);
```

This code has been incorporated into the function `circonv`, with usage:

```
y = circonv(h,x,N);          % mod-N circular convolution
```

For example, we have the outputs for $N = 8$:



$\boldsymbol{h} =$ 1 2 3 2 1

$\boldsymbol{x} =$ 1 2 3 4 5 6 7 8

$\boldsymbol{y} =$ 1 4 10 18 27 36 45 54 | 54 44 23 8 $= \text{conv}(\boldsymbol{h},\boldsymbol{x})$

54 44 23 8

$\tilde{\boldsymbol{y}} =$ 55 48 33 26 27 36 45 54 $= \text{circonv}(\boldsymbol{h},\boldsymbol{x},8)$

Circular convolution can also be implemented in the frequency domain by computing the N-point DFTs of the signals $\mathbf{h},\mathbf{x}$, multiplying them pointwise together, and performing an inverse N-point DFT. Symbolically,

$$\mathbf{y}_{\text{circ}} = \tilde{\mathbf{y}} = \widetilde{\mathbf{h}*\mathbf{x}} = \text{IDFT}\big[\text{DFT}(\mathbf{h})\cdot\text{DFT}(\mathbf{x})\big] \tag{10.5.7}$$

or, explicitly,

$$\tilde{y}(n) = \frac{1}{N}\sum_{k=0}^{N-1} H(\omega_k)X(\omega_k)e^{j\omega_k n} \tag{10.5.8}$$

where $H(\omega_k), X(\omega_k)$ are N-point DFTs as in Eq. (10.5.5). The following MATLAB code illustrates the implementation of the above example in the frequency and time domains:

```
h = [1 2 3 2 1];
x = [1 2 3 4 5 6 7 8];
H = fft(h,8); X = fft(x,8);          % calculate 8-point DFTs
Y = H.*X;                            % point-wise multiplication of the DFTs
ytilde = ifft(Y,8);                  % inverse DFT generates ỹ = [55,48,33,26,27,36,45,54]
ytilde = circonv(h,x,8);             % time-domain calculation
```

The frequency method (10.5.7) becomes efficient if FFTs are used in the right-hand side. However, for our DWT functions, we have used the time-domain implementations, which are equally efficient because the typical wavelet filter lengths are fairly short. The convolutional operations in Eqs. (10.5.1) and (10.5.2) can now be replaced by their circular versions, denoted symbolically,

$$\boxed{\begin{aligned}\mathbf{c}_{j-1} &= \big(\text{circonv}(\bar{\mathbf{h}},\mathbf{c}_j)\big)_{\text{down}}\\ \mathbf{d}_{j-1} &= \big(\text{circonv}(\bar{\mathbf{g}},\mathbf{c}_j)\big)_{\text{down}}\end{aligned}} \quad j = J, J-1,\dots,J_0+1$$

$$\boxed{\mathbf{c}_j = \text{circonv}(\mathbf{h},\mathbf{c}_{j-1}^{\text{up}}) + \text{circonv}(\mathbf{g},\mathbf{c}_{j-1}^{\text{up}})} \quad j = J_0+1, J_0+2,\dots,J \tag{10.5.9}$$

DWT in Matrix Form

The convolutional operation $\mathbf{y} = \mathbf{h}*\mathbf{x}$ can be represented in matrix form:

$$\mathbf{y} = H\mathbf{x}$$

where H is the convolution matrix of the filter h_n, defined by its matrix elements:

$$H_{nm} = h_{n-m}$$

The convolution matrix corresponding to the time-reversed filter $\bar{h}_n = h_{-n}$ is given by the transposed matrix

$$\bar{H} = H^T$$

because $\bar{H}_{nm} = \bar{h}_{n-m} = h_{m-n} = H_{mn}$. Thus, in matrix notation, the typical convolutional and down- and up-sampling operations being performed at the analysis and synthesis stages have the forms:

$$\mathbf{y} = (H^T\mathbf{x})_{\text{down}}\,, \qquad \mathbf{y} = H\mathbf{x}^{\text{up}} \tag{10.5.10}$$

Moreover, replacing the linear convolutions by circular ones amounts to replacing the convolutional matrices by their mod-N wrapped versions obtained by reducing their columns modulo-N, where N is the length of the input vector $\mathbf{x}$. Denoting $\tilde{H} = \text{modwrap}(H,N)$, then the circular version of (10.5.10) would read:

$$\tilde{\mathbf{y}} = (\tilde{H}^T\mathbf{x})_{\text{down}}\,, \qquad \tilde{\mathbf{y}} = \tilde{H}\mathbf{x}^{\text{up}} \tag{10.5.11}$$

The reduced matrix $\tilde{H}$ will have size $N\times N$, and after downsampling, the output $\tilde{\mathbf{y}} = (\tilde{H}^T\mathbf{x})_{\text{down}}$ will have size $N/2$. Similarly, in the operation $\tilde{\mathbf{y}} = \tilde{H}\mathbf{x}^{\text{up}}$, the upsampled vector $\mathbf{x}^{\text{up}}$ will have length N, as will the output $\tilde{\mathbf{y}}$. Before upsampling, the input $\mathbf{x}$ had length $N/2$. Because every other entry of $\mathbf{x}^{\text{up}}$ is zero, the matrix operation $\tilde{H}\mathbf{x}^{\text{up}}$ can be simplified by replacing $\tilde{H}$ by its "downsampled" version $\tilde{H}_{\text{down}}$ obtained by keeping every other column, and acting on the original vector $\mathbf{x}$, that is, $\tilde{H}\mathbf{x}^{\text{up}} = \tilde{H}_{\text{down}}\mathbf{x}$. The matrix elements of H_{down}, before they are wrapped modulo-N, are $(H_{\text{down}})_{nk} = h_{n-2k}$.

To clarify these remarks, we look at some examples. Consider a length-6 filter, such as D_3 or a Coiflet-1 filter, $\mathbf{h} = [h_0,h_1,h_2,h_3,h_4,h_5]^T$ and take $N = 8$. If the length-4 signal vector $\mathbf{x} = [x_0,x_2,x_4,x_6]^T$ is upsampled by a factor of two, it will become the length-8 vector $\mathbf{x}^{\text{up}} = [x_0,0,x_2,0,x_4,0,x_6,0]^T$. Before wrapping them modulo-8, the convolution matrices H, H_{down} generate the following equivalent outputs:

$$\mathbf{y} = \begin{bmatrix} h_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ h_1 & h_0 & 0 & 0 & 0 & 0 & 0 & 0\\ h_2 & h_1 & h_0 & 0 & 0 & 0 & 0 & 0\\ h_3 & h_2 & h_1 & h_0 & 0 & 0 & 0 & 0\\ h_4 & h_3 & h_2 & h_1 & h_0 & 0 & 0 & 0\\ h_5 & h_4 & h_3 & h_2 & h_1 & h_0 & 0 & 0\\ 0 & h_5 & h_4 & h_3 & h_2 & h_1 & h_0 & 0\\ 0 & 0 & h_5 & h_4 & h_3 & h_2 & h_1 & h_0\\ 0 & 0 & 0 & h_5 & h_4 & h_3 & h_2 & h_1\\ 0 & 0 & 0 & 0 & h_5 & h_4 & h_3 & h_2\\ 0 & 0 & 0 & 0 & 0 & h_5 & h_4 & h_3\\ 0 & 0 & 0 & 0 & 0 & 0 & h_5 & h_4\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & h_5 \end{bmatrix} \begin{bmatrix} x_0\\ 0\\ x_2\\ 0\\ x_4\\ 0\\ x_6\\ 0 \end{bmatrix} = \begin{bmatrix} h_0 & 0 & 0 & 0\\ h_1 & 0 & 0 & 0\\ h_2 & h_0 & 0 & 0\\ h_3 & h_1 & 0 & 0\\ h_4 & h_2 & h_0 & 0\\ h_5 & h_3 & h_1 & 0\\ 0 & h_4 & h_2 & h_0\\ 0 & h_5 & h_3 & h_1\\ 0 & 0 & h_4 & h_2\\ 0 & 0 & h_5 & h_3\\ 0 & 0 & 0 & h_4\\ 0 & 0 & 0 & h_5\\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_0\\ x_2\\ x_4\\ x_6 \end{bmatrix}$$

or, $\mathbf{y} = H\mathbf{x}^{\text{up}} = H_{\text{down}}\mathbf{x}$. The circular convolution output can be obtained by either wrapping $\mathbf{y}$ modulo-8 or by wrapping H, H_{down} columnwise:

$$\tilde{\mathbf{y}} = \tilde{H}\mathbf{x}^{\text{up}} = \tilde{H}_{\text{down}}\mathbf{x} = \begin{bmatrix} h_0 & 0 & h_4 & h_2 \\ h_1 & 0 & h_5 & h_3 \\ h_2 & h_0 & 0 & h_4 \\ h_3 & h_1 & 0 & h_5 \\ h_4 & h_2 & h_0 & 0 \\ h_5 & h_3 & h_1 & 0 \\ 0 & h_4 & h_2 & h_0 \\ 0 & h_5 & h_3 & h_1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_2 \\ x_4 \\ x_6 \end{bmatrix} \tag{10.5.12}$$

Similarly, in the analysis operation $\tilde{\mathbf{y}} = (\tilde{H}^T\mathbf{x})_{\text{down}}$, downsampling amounts to keeping every other row of the matrix $\tilde{H}^T$, which is $\tilde{H}^T_{\text{down}}$. For example, for the length-8 signal $\mathbf{x} = [x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7]^T$, the corresponding operation will be:

$$\tilde{\mathbf{y}} = (\tilde{H}^T\mathbf{x})_{\text{down}} = \tilde{H}^T_{\text{down}}\mathbf{x} = \begin{bmatrix} h_0 & h_1 & h_2 & h_3 & h_4 & h_5 & 0 & 0 \\ 0 & 0 & h_0 & h_1 & h_2 & h_3 & h_4 & h_5 \\ h_4 & h_5 & 0 & 0 & h_0 & h_1 & h_2 & h_3 \\ h_2 & h_3 & h_4 & h_5 & 0 & 0 & h_0 & h_1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} \tag{10.5.13}$$

The wrapped/downsampled convolution matrix $\tilde{H}_{\text{down}}$ can be calculated very simply in MATLAB, using, for example, the built-in convolution matrix function `convmtx` and the function `modwrap`:

```
H = convmtx(h(:), N);      % ordinary convolution matrix with N columns, h entered as column
H = H(:, 1:2:N);           % downsampled convolution matrix
H = modwrap(H, N);         % wrapped column-wise modulo-N
```

Because $\mathbf{h}$ is fairly short and N typically large, the convolution matrix H can be defined as sparse. This can be accomplished by replacing `convmtx` by the function `convmat`, which we encountered before in Sec. 3.9. Similar convolution matrices $\tilde{G}_{\text{down}}$ can be constructed for the conjugate mirror filter g_n. The function `dwtmat` constructs both matrices for any scaling filter $\mathbf{h}$ and signal length N using `convmat`:

```
[H,G] = dwtmat(h,N);      % sparse DWT matrices
```

The output matrices H, G are defined as sparse and have dimension $N\times(N/2)$. They represent the matrices $\tilde{H}_{\text{down}}, \tilde{G}_{\text{down}}$.

We can now state the precise form of the matrix version of the periodized DWT algorithm. Given a signal (column) vector $\mathbf{x}$ of length $N = 2^J$, we define the DWT matrices H_j, G_j at level j with dimension $N_j\times(N_j/2)$, where $N_j = 2^j$, by

$$[H_j, G_j] = \text{dwtmat}(\mathbf{h}, N_j)\,, \qquad J_0 + 1 \le j \le J \tag{10.5.14}$$

Then, the analysis and synthesis algorithms are as follows, initialized with $\mathbf{c}_J = \mathbf{x}$,

$$\text{(DWT)}\quad \boxed{\begin{aligned} \mathbf{c}_{j-1} &= H_j^T\mathbf{c}_j \\ \mathbf{d}_{j-1} &= G_j^T\mathbf{c}_j \end{aligned}} \qquad j = J, J-1, \dots, J_0 + 1$$

$$\text{(IDWT)}\quad \boxed{\mathbf{c}_j = H_j\mathbf{c}_{j-1} + G_j\mathbf{d}_{j-1}} \qquad j = J_0+1, J_0+2, \dots, J \tag{10.5.15}$$

The column vector $\mathbf{c}_j$ has dimension $N_j = 2^j$, while the vectors $\mathbf{c}_{j-1}, \mathbf{d}_{j-1}$ have dimension half of that, $N_{j-1} = N_j/2 = 2^{j-1}$. The computations for the forward and inverse transforms are illustrated in Fig. 10.5.2. The *discrete wavelet transform* of $\mathbf{x}$ to level J_0 is the concatenation of the coefficient vectors:

$$\mathbf{w} = \begin{bmatrix} \mathbf{c}_{J_0} \\ \mathbf{d}_{J_0} \\ \mathbf{d}_{J_0+1} \\ \vdots \\ \mathbf{d}_{J-1} \end{bmatrix} \qquad \text{(DWT)} \tag{10.5.16}$$

Its total dimension is $N = 2^J$, as can be verified easily,

$$2^{J_0} + 2^{J_0} + \left(2^{J_0+1} + \cdots + 2^{J-1}\right) = 2^J$$

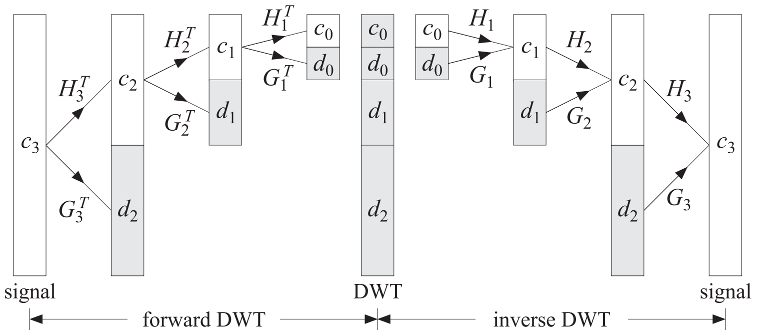


Fig. 10.5.2 Forward and inverse DWT in matrix form.

At each level j, the $N_j\times N_j$ matrix $U_j = [H_j, G_j]$ is an orthogonal matrix, as required by the consistency of the analysis and synthesis steps:

$$\begin{bmatrix} \mathbf{c}_{j-1} \\ \mathbf{d}_{j-1} \end{bmatrix} = \begin{bmatrix} H_j^T \\ G_j^T \end{bmatrix} \mathbf{c}_j \quad \Leftrightarrow \quad \mathbf{c}_j = H_j\mathbf{c}_{j-1} + G_j\mathbf{d}_{j-1} = [H_j, G_j] \begin{bmatrix} \mathbf{c}_{j-1} \\ \mathbf{d}_{j-1} \end{bmatrix}$$

implying the conditions $U_j^T U_j = U_j U_j^T = I_{N_j}$, or,

$$\begin{bmatrix} H_j^T \\ G_j^T \end{bmatrix} [H_j, G_j] = [H_j, G_j] \begin{bmatrix} H_j^T \\ G_j^T \end{bmatrix} = I_{N_j}$$

which are equivalent to the orthogonality conditions:

$$H_j^T H_j = G_j^T G_j = I_{N_j/2}\,, \quad H_j^T G_j = 0\,, \quad H_j H_j^T + G_j G_j^T = I_{N_j} \tag{10.5.17}$$

These follow from the scaling filter orthogonality properties (10.3.1). To see the mechanics by which this happens, consider again our length-6 filter h_n and the corresponding CMF filter g_n defined by $[g_0, g_1, g_2, g_3, g_4, g_5] = [h_5, -h_4, h_3, -h_2, h_1, -h_0]$. Let us also define the cross-correlation quantities:

$$R_k = \sum_n h_n h_{n-2k} \quad \Rightarrow \quad \begin{cases} R_0 = h_0^2 + h_1^2 + h_2^2 + h_3^2 + h_4^2 + h_5^2 \\ R_1 = h_5 h_3 + h_4 h_2 + h_3 h_1 + h_2 h_0 \\ R_2 = h_5 h_1 + h_4 h_0 \end{cases} \tag{10.5.18}$$

From Eq. (10.3.1), we have $R_k = \delta_k$, but let us not assume this just yet, but rather treat h_n as an arbitrary filter and g_n as the corresponding CMF filter. Then, starting with level $J = 3$, the wavelet matrices H_j at $j = 3, 2, 1$, will be:

$$H_3 = \begin{bmatrix} h_0 & 0 & h_4 & h_2 \\ h_1 & 0 & h_5 & h_3 \\ h_2 & h_0 & 0 & h_4 \\ h_3 & h_1 & 0 & h_5 \\ h_4 & h_2 & h_0 & 0 \\ h_5 & h_3 & h_1 & 0 \\ 0 & h_4 & h_2 & h_0 \\ 0 & h_5 & h_3 & h_1 \end{bmatrix}, \qquad \begin{aligned} H_2 &= \begin{bmatrix} h_0 + h_4 & h_2 \\ h_1 + h_5 & h_3 \\ h_2 & h_0 + h_4 \\ h_3 & h_1 + h_5 \end{bmatrix} \\ H_1 &= \begin{bmatrix} h_0 + h_2 + h_4 \\ h_1 + h_3 + h_5 \end{bmatrix} \end{aligned} \tag{10.5.19}$$

with similar definitions for G_j, $j = 3, 2, 1$. By explicit multiplication, we can verify:

$$H_3^T H_3 = G_3^T G_3 = \begin{bmatrix} R_0 & R_1 & 2R_2 & R_1 \\ R_1 & R_0 & R_1 & 2R_2 \\ 2R_2 & R_1 & R_0 & R_1 \\ R_1 & 2R_2 & R_1 & R_0 \end{bmatrix}, \quad H_3^T G_3 = 0$$

$$H_3 H_3^T + G_3 G_3^T = \begin{bmatrix} R_0 & 0 & R_1 & 0 & 2R_2 & 0 & R_1 & 0 \\ 0 & R_0 & 0 & R_1 & 0 & 2R_2 & 0 & R_1 \\ R_1 & 0 & R_0 & 0 & R_1 & 0 & 2R_2 & 0 \\ 0 & R_1 & 0 & R_0 & 0 & R_1 & 0 & 2R_2 \\ 2R_2 & 0 & R_1 & 0 & R_0 & 0 & R_1 & 0 \\ 0 & 2R_2 & 0 & R_1 & 0 & R_0 & 0 & R_1 \\ R_1 & 0 & 2R_2 & 0 & R_1 & 0 & R_0 & 0 \\ 0 & R_1 & 0 & 2R_2 & 0 & R_1 & 0 & R_0 \end{bmatrix}$$

Similarly, we have,

$$H_2^T H_2 = G_2^T G_2 = \begin{bmatrix} R_0 + 2R_2 & 2R_1 \\ 2R_1 & R_0 + 2R_2 \end{bmatrix}, \quad H_2^T G_2 = 0$$

$$H_2 H_2^T + G_2 G_2^T = \begin{bmatrix} R_0 + 2R_2 & 0 & 2R_1 & 0 \\ 0 & R_0 + 2R_2 & 0 & 2R_1 \\ 2R_1 & 0 & R_0 + 2R_2 & 0 \\ 0 & 2R_1 & 0 & R_0 + 2R_2 \end{bmatrix}$$

$$H_1^T H_1 = G_1^T G_1 = R_0 + 2R_1 + 2R_2\,, \quad H_1^T G_1 = 0$$

$$H_1 H_1^T + G_1 G_1^T = \begin{bmatrix} R_0 + 2R_1 + 2R_2 & 0 \\ 0 & R_0 + 2R_1 + 2R_2 \end{bmatrix}$$

Setting $R_0 = 1$ and $R_1 = R_2 = 0$ in all of the above, we verify the orthogonality properties (10.5.17) at all levels $j = 3, 2, 1$. We note that the matrix H_{j-1} can be derived very simply from H_j by keeping only the first $N_{j-1}/2 = N_j/4$ columns and wrapping them modulo-N_{j-1}, that is, in MATLAB notation:

$$H_{j-1} = \text{modwrap}\big(H_j(:, 1:N_{j-1}/2), N_{j-1}\big)\,, \quad j = J, J{-}1, \dots, J_0 + 1 \tag{10.5.20}$$

and similarly for G_{j-1}. This simple operation has been incorporated into the function `dwtwrap`, with usage:

```
H_lower = dwtwrap(H);        % wrap a DWT matrix into a lower one
```

This is evident in Eq. (10.5.19), where H_2 is derivable from H_3, and H_1 from H_2. Because the successive DWT matrices H_j, G_j have different dimensions, $2^j{\times}2^{j-1}$, it is convenient to use a cell array to store them in MATLAB. The function `dwtcell` constructs and stores them in sparse form:

```
F = dwtcell(h,N);        % cell array of sparse DWT matrices
```

with the conventions $H_j = F\{1,j\}$ and $G_j = F\{2,j\}$, for $J_0 + 1 \le j \le J$, where N is the highest dimension. The function `fwtm` implements the analysis algorithm in (10.5.15). Its inputs are the signal vector $\mathbf{x}$, the cell array F, and the lowest desired level J_0,

```
w = fwtm(x,F,J0);        % fast wavelet transform in matrix form
```

The vector $\mathbf{w}$ is as in Eq. (10.5.16). If J_0 is omitted, it defaults to $J_0 = 0$. Once the cell array F is created, the function `fwtm` is extremely fast, even faster than the convolution-based function `fwt` discussed below. The function `ifwtm` implements the inverse DWT synthesis algorithm in (10.5.15),

```
x = ifwtm(w,F,J0);        % inverse fast wavelet transform in matrix form
```

An example is the following MATLAB code using the D_3 scaling filter:

```
x = [1 2 3 4 5 6 7 8];
h = daub(3);                % h = [0.3327, 0.8069, 0.4599, −0.1350, −0.0854, 0.0352]
F = dwtcell(h,8);           % construct cell array of DWT matrices

w = fwtm(x,F,0);            % w = [12.7279, −1.4794, −4.4090, 2.2467, 0, 0, −3.7938, 0.9653]
x = ifwtm(w,F,0);           % returns x = [1, 2, 3, 4, 5, 6, 7, 8]
                            % similarly, for J0 = 1,2,3, we find,
w = fwtm(x,F,1);            % w = [7.9539, 10.0461, −4.409, 2.2467, 0, 0, −3.7938, 0.9653]
w = fwtm(x,F,2);            % w = [2.5702, 5.3986, 8.6288, 8.8583, 0, 0, −3.7938, 0.9653]
w = fwtm(x,F,3);            % w = [1, 2, 3, 4, 5, 6, 7, 8] = x, as expected since J = 3
```

These outputs can be understood by looking at the individual matrix operations. Defining, $\mathbf{c}_3 = \mathbf{x} = [1,2,3,4,5,6,7,8]^T$, and the level-3 matrices H_3, G_3, obtained from the call, $[H_3, G_3] = \text{dwtmat}(\mathbf{h}, 8)$,

$$H_3 = \begin{bmatrix} 0.3327 & 0 & -0.0854 & 0.4599 \\ 0.8069 & 0 & 0.0352 & -0.1350 \\ 0.4599 & 0.3327 & 0 & -0.0854 \\ -0.1350 & 0.8069 & 0 & 0.0352 \\ -0.0854 & 0.4599 & 0.3327 & 0 \\ 0.0352 & -0.1350 & 0.8069 & 0 \\ 0 & -0.0854 & 0.4599 & 0.3327 \\ 0 & 0.0352 & -0.1350 & 0.8069 \end{bmatrix}, \quad G_3 = \begin{bmatrix} 0.0352 & 0 & 0.8069 & -0.1350 \\ 0.0854 & 0 & -0.3327 & -0.4599 \\ -0.1350 & 0.0352 & 0 & 0.8069 \\ -0.4599 & 0.0854 & 0 & -0.3327 \\ 0.8069 & -0.1350 & 0.0352 & 0 \\ -0.3327 & -0.4599 & 0.0854 & 0 \\ 0 & 0.8069 & -0.1350 & 0.0352 \\ 0 & -0.3327 & -0.4599 & 0.0854 \end{bmatrix}$$

we calculate the level-2 coefficient vectors $\mathbf{c}_2, \mathbf{d}_2$, and the level-2 DWT,

$$\mathbf{c}_2 = H_3^T \mathbf{c}_3 = \begin{bmatrix} 2.5702 \\ 5.3986 \\ 8.6288 \\ 8.8583 \end{bmatrix}, \quad \mathbf{d}_2 = G_3^T \mathbf{c}_3 = \begin{bmatrix} 0 \\ 0 \\ -3.7938 \\ 0.9653 \end{bmatrix}, \quad \mathbf{w} = \begin{bmatrix} \mathbf{c}_2 \\ \mathbf{d}_2 \end{bmatrix}$$

which agrees with the above MATLAB output of `fwtm(x,F,2)`. Then, from the matrices H_2, G_2, obtained from $[H_2, G_2] = \text{dwtmat}(\mathbf{h}, 4)$, or, from $H_2 = \text{dwtwrap}(H_3)$,

$$H_2 = \begin{bmatrix} 0.2472 & 0.4599 \\ 0.8421 & -0.1350 \\ 0.4599 & 0.2472 \\ -0.1350 & 0.8421 \end{bmatrix}, \quad G_2 = \begin{bmatrix} 0.8421 & -0.1350 \\ -0.2472 & -0.4599 \\ -0.1350 & 0.8421 \\ -0.4599 & -0.2472 \end{bmatrix}$$

we calculate the level-1 coefficient vectors $\mathbf{c}_1, \mathbf{d}_1$, and the level-1 DWT,

$$\mathbf{c}_1 = H_2^T \mathbf{c}_2 = \begin{bmatrix} 7.9539 \\ 10.0461 \end{bmatrix}, \quad \mathbf{d}_1 = G_2^T \mathbf{c}_2 = \begin{bmatrix} -4.4090 \\ 2.2467 \end{bmatrix}, \quad \mathbf{w} = \begin{bmatrix} \mathbf{c}_1 \\ \mathbf{d}_1 \\ \mathbf{d}_2 \end{bmatrix}$$

which agrees with the above MATLAB output of `fwtm(x,F,1)`. Finally, from the matrices H_1, G_1, obtained from $[H_1, G_1] = \text{dwtmat}(\mathbf{h}, 2)$, or, from $H_1 = \text{dwtwrap}(H_2)$,

$$H_1 = \begin{bmatrix} 0.7071 \\ 0.7071 \end{bmatrix}, \quad G_1 = \begin{bmatrix} 0.7071 \\ -0.7071 \end{bmatrix}$$

we find the level-0 coefficient vectors $\mathbf{c}_0, \mathbf{d}_0$, and the level-0 DWT,

$$\mathbf{c}_0 = H_1^T \mathbf{c}_1 = 12.7279, \quad \mathbf{d}_0 = G_1^T \mathbf{c}_1 = -1.4794, \quad \mathbf{w} = \begin{bmatrix} \mathbf{c}_0 \\ \mathbf{d}_0 \\ \mathbf{d}_1 \\ \mathbf{d}_2 \end{bmatrix}$$

Orthogonal DWT Transformation

The mapping of a length-N signal vector $\mathbf{x}$ to the length-N vector $\mathbf{w}$ of wavelet coefficients given in Eq. (10.5.16) is equivalent to an orthogonal matrix transformation, say, $\mathbf{w} = W^T\mathbf{x}$, with inverse $\mathbf{x} = W\mathbf{w}$, such that $W^T W = W W^T = I_N$. The overall $N\times N$ matrix W depends on the stopping level J_0 and can be constructed in terms of the matrices H_j, G_j of the successive stages of the analysis or synthesis algorithms. For example, we have for $N = 2^3$, and $J_0 = 2, 1, 0$,

$$W = [H_3,\ G_3]$$

$$W = \big[H_3[H_2,\ G_2],\ G_3\big] = [H_3H_2,\ H_3G_2,\ G_3]$$

$$W = \big[H_3H_2[H_1,\ G_1],\ H_3G_2,\ G_3\big] = [H_3H_2H_1,\ H_3H_2G_1,\ H_3G_2,\ G_3]$$

We verify the reconstruction of $\mathbf{x}$ from $\mathbf{w}$ starting at $J_0 = 0$,

$$[H_3H_2H_1,\ H_3H_2G_1,\ H_3G_2,\ G_3]\begin{bmatrix} \mathbf{c}_0 \\ \mathbf{d}_0 \\ \mathbf{d}_1 \\ \mathbf{d}_2 \end{bmatrix} = \begin{cases} H_3H_2\,(H_1\mathbf{c}_0 + G_1\mathbf{d}_0) + H_3G_2\mathbf{d}_1 + G_3\mathbf{d}_2 = \\ H_3\,(H_2\mathbf{c}_1 + G_2\mathbf{d}_1) + G_3\mathbf{d}_2 = \\ H_3\mathbf{c}_2 + G_3\mathbf{d}_2 = \mathbf{c}_3 = \mathbf{x} \end{cases}$$

The construction of W can be carried out with the following very simple recursive algorithm, stated in MATLAB notation,

$$\begin{aligned} & W = I_N\,, \quad (N = 2^J) \\ & \text{for } j = J, J-1, \dots, J_0 + 1, \\ & \qquad W(:,\ 1:2^j) = W(:,\ 1:2^j)\,[H_j, G_j] \end{aligned} \tag{10.5.21}$$

The algorithm updates the first 2^j columns of W at each level j. The MATLAB function `fwtmat` implements (10.5.21) and constructs W as a sparse matrix:

```
W = fwtmat(h,N,J0);        % overall DWT orthogonal matrix
```

As an example, for the D_3 scaling filter and $N = 8$ and lowest level $J_0 = 0$, we find:

$$W = \begin{bmatrix} 0.3536 & -0.3806 & 0.0802 & -0.2306 & 0.0352 & 0 & 0.8069 & -0.1350 \\ 0.3536 & -0.0227 & 0.7368 & -0.0459 & 0.0854 & 0 & -0.3327 & -0.4599 \\ 0.3536 & 0.2197 & 0.3443 & -0.1940 & -0.1350 & 0.0352 & 0 & 0.8069 \\ 0.3536 & 0.5535 & -0.3294 & -0.3616 & -0.4599 & 0.0854 & 0 & -0.3327 \\ 0.3536 & 0.3806 & -0.2306 & 0.0802 & 0.8069 & -0.1350 & 0.0352 & 0 \\ 0.3536 & 0.0227 & -0.0459 & 0.7368 & -0.3327 & -0.4599 & 0.0854 & 0 \\ 0.3536 & -0.2197 & -0.1940 & 0.3443 & 0 & 0.8069 & -0.1350 & 0.0352 \\ 0.3536 & -0.5535 & -0.3616 & -0.3294 & 0 & -0.3327 & -0.4599 & 0.0854 \end{bmatrix}$$

which generates the same DWT as the example above:

```
x = [1 2 3 4 5 6 7 8]'; h = daub(3); W = fwtmat(h,8,0);
w = W'*x;    % gives w = [12.7279, −1.4794, −4.4090, 2.2467, 0, 0, −3.7938, 0.9653]^T
```

The matrix W becomes more and more sparse as N increases. Its sparsity pattern is illustrated in Fig. 10.5.3, for the case of the D_3 scaling filter and dimensions $N = 64$ and $N = 512$. The graphs were generated by the MATLAB code:

```
h = daub(3); N = 64; W = fwtmat(h,N,0); spy(W); percent_nonzero = 100*nnz(W)/N^2
```

The percentages of nonzero entries were 30.5 % for $N = 64$, and 6.7 % for $N = 512$.

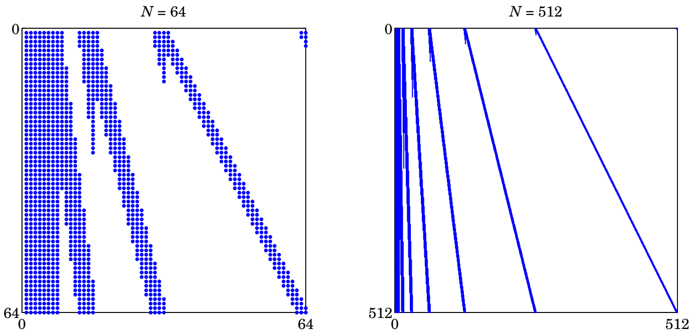


Fig. 10.5.3 Sparsity patterns of DWT matrices.

DWT in Convolutional Form

Next, we look at the detailed implementation of Eq. (10.5.9) using filtering by circular convolution. For practical implementation, we must replace the time-reversed filters $\bar{h}_n, \bar{g}_n$ of the analysis algorithm by their reversed versions, which are delayed by the filter order M to make them causal, that is, $h_n^R = \bar{h}_{n-M} = h_{M-n}$, or in the z-domain $H^R(z) = z^{-M}\bar{H}(z) = z^{-M}H(z^{-1})$.

In order to get the same output as the matrix implementation, we must compensate for such a delay by advancing the input by the same amount. In other words, filtering by $\bar{H}(z)$ is equivalent to advancing the input and then filtering by $H^R(z)$. In the z-domain,

$$Y(z) = \bar{H}(z)X(z) = z^M H^R(z)X(z) = H^R(z)\big[z^M X(z)\big]$$

With these changes, Eq. (10.5.9) now reads,

$$\boxed{\begin{aligned}&\text{advance}(\mathbf{c}_j, M)\\ &\mathbf{c}_{j-1} = \big(\text{circonv}(\mathbf{h}^R, \mathbf{c}_j)\big)_{\text{down}}\\ &\mathbf{d}_{j-1} = \big(\text{circonv}(\mathbf{g}^R, \mathbf{c}_j)\big)_{\text{down}}\end{aligned}} \qquad j = J, J-1, \dots, J_0 + 1$$

$$\boxed{\mathbf{c}_j = \text{circonv}(\mathbf{h}, \mathbf{c}_{j-1}^{\text{up}}) + \text{circonv}(\mathbf{g}, \mathbf{c}_{j-1}^{\text{up}})} \qquad j = J_0+1, J_0+2, \dots, J \tag{10.5.22}$$

The concrete MATLAB implementation for computing the forward DWT is:

```
g = cmf(h);                          % conjugate mirror of h
hR = flip(h);                        % reversed h
gR = flip(g);
M = length(h) - 1;                   % filter order

c = x(:);                            % initial smooth, x has length 2^J
w = [];                              % DWT coefficient vector

for j=J:-1:J0+1,                     % loop from finest down to coarsest level
   c = advance(c, M);                % length(c) = 2^j
   d = dn2(circonv(gR, c, 2^j));     % convolve circularly and downsample
```

```
   c = dn2(circonv(hR, c, 2^j));
   w = [d; w];                               % prepend detail d to previous details
end

w = [c; w];                                  % prepend last smooth
```

The function `advance` actually performs a *circular* time-advance modulo the length of its argument vector. The function `dn2` performs downsampling by a factor of two. The results of each loop calculation are appended into the DWT vector **w**. Similarly, the inverse DWT can be calculated by the loop:

```
w = w(:);                                                      % work columnwise
c = wcoeff(w,J0);                                              % coarsest smooth at level J0
for j=J0+1:J,
  d = wcoeff(w,J0,j-1);                                        % get detail at level j-1
  c = circonv(h, up2(c), 2^j) + circonv(g, up2(d), 2^j);       % output c is 2^j-dimensional
end
x = c;                                                         % reconstructed x
```

Here, the function `wcoeff(w,J0,j-1)` extracts the subvector $\mathbf{d}_{j-1}$ from the wavelet transform vector **w**, and the function `up2` upsamples by a factor of two.

The MATLAB functions `fwt` and `ifwt` incorporate the above code segments to realize the convolutional forms of the DWT and IDWT:

```
w = fwt(x,h,J0);     % fast wavelet transform
x = ifwt(w,h,J0);    % inverse fast wavelet transform
```

Some examples are,

```
x = [1 2 3 4 5 6 7 8];
h = daub(3);
w = fwt(x,h,0);      % w = [12.7279, −1.4794, −4.4090, 2.2467, 0, 0, −3.7938, 0.9653]
x = ifwt(w,h,0);     % returns x = [1, 2, 3, 4, 5, 6, 7, 8]
w = fwt(x,h,1);      % w = [7.9539, 10.0461, −4.409, 2.2467, 0, 0, −3.7938, 0.9653]
w = fwt(x,h,2);      % w = [2.5702, 5.3986, 8.6288, 8.8583, 0, 0, −3.7938, 0.9653]
w = fwt(x,h,3);      % w = [1, 2, 3, 4, 5, 6, 7, 8] = x, as expected
```

A second optional output of `fwt` (and `fwtm`) is the $N\times(J-J_0+1)$ matrix V whose columns are the sub-blocks of **w** according to their resolution,

$$\mathbf{w} = \begin{bmatrix}\mathbf{c}_{J_0}\\ \mathbf{d}_{J_0}\\ \mathbf{d}_{J_0+1}\\ \vdots\\ \mathbf{d}_{J-1}\end{bmatrix} \quad\Rightarrow\quad V = \begin{bmatrix}\mathbf{c}_{J_0} & 0 & 0 & \cdots & 0\\ 0 & \mathbf{d}_{J_0} & 0 & \cdots & 0\\ 0 & 0 & \mathbf{d}_{J_0+1} & \cdots & 0\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & 0 & \cdots & \mathbf{d}_{J-1}\end{bmatrix} \tag{10.5.23}$$

It is obtained by the calls,

```
[w,V] = fwt(x,h,J0);
[w,V] = fwtm(x,F,J0);
```

The computations in fwt are very efficient, resulting in an $O(N)$ algorithm, or more precisely, $O(MN)$, where M is the filter order. By contrast, the FFT is an $O(N\log_2 N)$ algorithm. However, because of their sparsity, the matrix versions are just as efficient if the sparse wavelet matrices are precomputed.

We mentioned earlier that there are several different implementations of the DWT. Different packages may produce different answers, sometimes only differing by a sign or a cyclic permutation within each level. For example, we obtained the following answers for the above example ($\mathbf{x} = [1,2,3,4,5,6,7,8]$ with D_3 and $J_0 = 0$) from the packages:

$$\begin{array}{l}
\mathbf{w} = [12.7279, -1.4794, -4.4090, \;\; 2.2467, \;\; 0.0000, \;\; 0.0000, -3.7938, \;\; 0.9653] = \text{fwt - ours} \\
\mathbf{w} = [12.7279, \;\; 1.4794, \;\; 4.4090, -2.2467, \;\; 3.7938, -0.9653, \;\; 0.0000, \;\; 0.0000] = \text{Wavelab850, Ref. [836]} \\
\mathbf{w} = [12.7279, -1.4794, -4.4090, \;\; 2.2467, -3.7938, \;\; 0.9653, \;\; 0.0000, \;\; 0.0000] = \text{Wavethresh, Ref. [840]} \\
\mathbf{w} = [12.7279, \;\; 1.4794, -2.2467, \;\; 4.4090, -0.9653, \;\; 3.7938, \;\; 0.0000, \;\; 0.0000] = \text{WMTSA, Ref. [844]} \\
\mathbf{w} = [12.7279, -1.4794, \;\; 2.2467, -4.4090, \;\; 0.9653, \;\; 0.0000, \;\; 0.0000, -3.7938] = \text{Uvi-Wave, Ref. [845]} \\
\mathbf{w} = [12.7279, \;\; 1.4794, \;\; 4.4090, -2.2467, \;\; 0.0000, \;\; 0.0000, \;\; 3.7938, -0.9653] = \text{Getz, Ref. [846]} \\
\mathbf{w} = [12.7279, -1.4794, -4.4090, \;\; 2.2467, \;\; 0.0000, \;\; 0.0000, -3.7938, \;\; 0.9653] = \text{Wavekit, Ref. [847]}
\end{array}$$

10.6 Multiresolution Decomposition

The multiresolution decomposition defined in Eq. (10.1.13), with coarsest level J_0, which was illustrated by Example 10.1.1, and implemented by the function dwtdec,

$$f(t) = \sum_n c_{Jn}\phi_{Jn}(t) = \sum_n c_{J_0 n}\phi_{J_0 n}(t) + \sum_{j=J_0}^{J-1}\sum_n d_{jn}\psi_{jn}(t)\,, \tag{10.6.1}$$

can be given a vectorial interpretation. Let $\mathbf{x}$ be the $N = 2^J$ dimensional vector of time samples of the function $f(t)$ at the finest level J, and let W be the orthogonal DWT matrix down to level J_0, with corresponding DWT, $\mathbf{w} = W^T\mathbf{x}$, and inverse $\mathbf{x} = W\mathbf{w}$.

Writing the DWT $\mathbf{w}$ in the partitioned form of Eq. (10.5.23), we may write $\mathbf{x}$ as the sum of multiresolution components, corresponding to the terms of (10.6.1), with each term representing the part of $\mathbf{x}$ arising from a particular level j with all the other levels having zero coefficients:

$$\begin{aligned}
\mathbf{x} = W\begin{bmatrix}\mathbf{c}_{J_0}\\ \mathbf{d}_{J_0}\\ \mathbf{d}_{J_0+1}\\ \vdots \\ \mathbf{d}_{J-1}\end{bmatrix} &= W\begin{bmatrix}\mathbf{c}_{J_0}\\ 0\\ 0\\ \vdots \\ 0\end{bmatrix} + W\begin{bmatrix}0\\ \mathbf{d}_{J_0}\\ 0\\ \vdots \\ 0\end{bmatrix} + W\begin{bmatrix}0\\ 0\\ \mathbf{d}_{J_0+1}\\ \vdots \\ 0\end{bmatrix} + \cdots + W\begin{bmatrix}0\\ 0\\ 0\\ \vdots \\ \mathbf{d}_{J-1}\end{bmatrix} \\
&= \mathbf{x}_{J_0} + \underbrace{(\,\bar{\mathbf{x}}_{J_0} + \bar{\mathbf{x}}_{J_0+1} + \cdots + \bar{\mathbf{x}}_{J-1}\,)}_{\mathbf{x}_{J_0}^{\perp}} \\
&= \mathbf{x}_{J_0} + \mathbf{x}_{J_0}^{\perp}
\end{aligned} \tag{10.6.2}$$

The terms $\mathbf{x}_{J_0}, \mathbf{x}_{J_0}^{\perp}$ represent the two parts of $\mathbf{x}$ lying in the subspaces V_{J_0} and $V_{J_0}^{\perp}$. The individual terms of $\mathbf{x}_{J_0}^{\perp} = \bar{\mathbf{x}}_{J_0} + \bar{\mathbf{x}}_{J_0+1} + \cdots + \bar{\mathbf{x}}_{J-1}$ contain all the details for levels $J_0 \le j \le J-1$. The various components are mutually orthogonal, as follows from the property $WW^T = I$, and the non-overlapping of the sub-blocks of $\mathbf{w}$,

$$\begin{aligned}
\mathbf{x}_{J_0}^T\bar{\mathbf{x}}_j &= 0, \quad J_0 \le j \le J-1 \\
\bar{\mathbf{x}}_i^T\bar{\mathbf{x}}_j &= 0, \quad J_0 \le i, j \le J-1\,, \quad i \ne j
\end{aligned} \tag{10.6.3}$$

For the "diagonal" terms, we obtain the norms, again following from $WW^T = I$,

$$\|\mathbf{x}_{J_0}\|^2 = \|\mathbf{c}_{J_0}\|^2\,, \quad \|\bar{\mathbf{x}}_j\|^2 = \|\mathbf{d}_j\|^2\,, \quad J_0 \le j \le J-1 \tag{10.6.4}$$

where $\|\mathbf{x}\|^2 = \mathbf{x}^T\mathbf{x}$, which lead to the sum,

$$\|\mathbf{x}\|^2 = \|\mathbf{x}_{J_0}\|^2 + \sum_{j=J_0}^{J-1}\|\bar{\mathbf{x}}_j\|^2 = \|\mathbf{c}_{J_0}\|^2 + \sum_{j=J_0}^{J-1}\|\mathbf{d}_j\|^2 = \|\mathbf{w}\|^2 \tag{10.6.5}$$

The $N\times(J-J_0+1)$ matrix $X = [\mathbf{x}_{J_0}, \bar{\mathbf{x}}_{J_0}, \bar{\mathbf{x}}_{J_0+1} \cdots \bar{\mathbf{x}}_{J-1}]$ incorporates the individual orthogonal columns and is produced as the output of the MATLAB function dwtdec,

```
X = dwtdec(x,h,J0);      % DWT decomposition into orthogonal multiresolution components
```

In fact, X is the product $X = WV$, where V is the DWT-component matrix given in (10.5.23). As an example of dwtdec, we have for $\mathbf{x} = [1,2,3,4,5,6,7,8]^T$ and D_3,

$$\mathbf{x} = \begin{bmatrix}1\\2\\3\\4\\5\\6\\7\\8\end{bmatrix} \Rightarrow \mathbf{w} = \begin{bmatrix}12.7279\\-1.4794\\-4.4090\\2.2467\\0\\0\\-3.7938\\0.9653\end{bmatrix} \Rightarrow X = \begin{bmatrix}4.5000 & 0.5631 & -0.8716 & -3.1915\\ 4.5000 & 0.0337 & -3.3518 & 0.8181\\ 4.5000 & -0.3251 & -1.9538 & 0.7789\\ 4.5000 & -0.8188 & 0.6399 & -0.3211\\ 4.5000 & -0.5631 & 1.1967 & -0.1336\\ 4.5000 & -0.0337 & 1.8578 & -0.3241\\ 4.5000 & 0.3251 & 1.6287 & 0.5462\\ 4.5000 & 0.8188 & 0.8541 & 1.8271\end{bmatrix}$$

generated with the MATLAB code and test,

```
h = daub(3); x = [1 2 3 4 5 6 7 8]'; X = dwtdec(x,h,0);
[w,V] = fwt(x,h,0); W = fwtmat(h,8,0); norm(X-W*V)
```

10.7 Wavelet Denoising

Figure 10.7.1 shows some wavelet denoising examples consisting of the same four signals (bumps, blocks, heavisine, doppler) that we discussed in Sec. 5.4 under local polynomial modeling with adaptive variable bandwidth. These examples have served as benchmarks in the wavelet denoising literature [821–824].

Fig. 10.7.1 should be compared with Figs. 5.4.1–5.4.4. It should be evident that the results are comparable, with, perhaps, local polynomial modeling doing a bit better. The MATLAB codes generating the noisy signals were given in Sec. 5.4. The following code segment illustrates the generation of the upper row of graphs and demonstrates the use of the denoising function wdenoise:

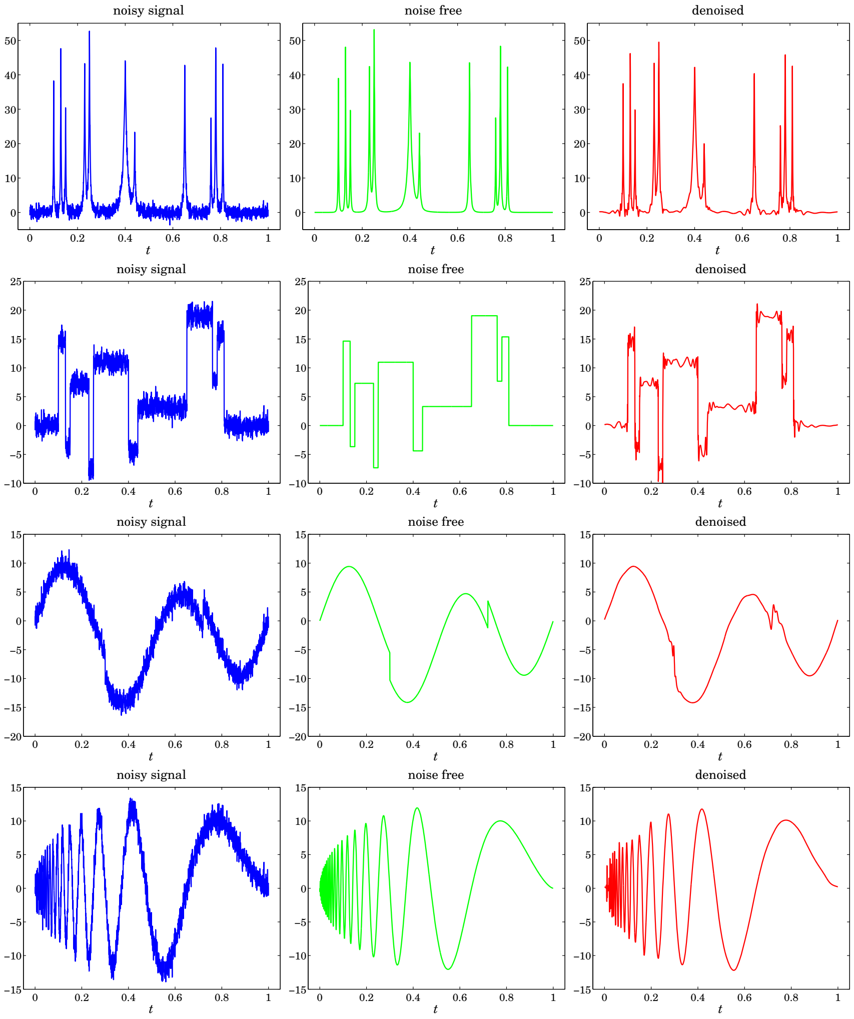


Fig. 10.7.1 Wavelet denoising.

```
F = inline('1./(1 + abs(x)).^4');                  % bumps function

N = 2048; t = (0:N-1)'/N; x = zeros(size(t));      % normalize time to 0 ≤ t ≤ 1

t0 = [10 13 15 23 25 40 44 65 76 78 81]/100;       % signal parameters
a = [40,50,30,40,50,42,21,43,31,51,42] * 1.0523;
w = [5,5,6,10,10,30,10,10,5,8,5]/1000;

for i=1:length(a),                                 % construct noise-free signal
```

```
   x = x + a(i) * F((t-t0(i))/w(i));
end

seed=2009; randn('state',seed); v = randn(size(t));   % generate noise

y = x + v;                                            % noisy signal with SNR, σx/σv = 7

h = daub(8,2); J0=5; type=1;                          % use Symmlet-8 and soft thresholding
xd = wdenoise(y,h,J0,type);                           % wavelet denoising

figure; plot(t,y,'-'); figure; plot(t,x,'-'); figure; plot(t,xd,'r-');     % top row
```

The main idea in wavelet denoising is to (a) perform a DWT on the noisy signal down to some lower resolution level , (b) modify the wavelet detail coefficients by reducing them to zero if they fall below a certain threshold, and (c) perform an inverse DWT to obtain the denoised signal. The procedure is depicted below:

$$\mathbf{x} \xrightarrow{\text{DWT}} \mathbf{w} \xrightarrow{\text{thresh}} \mathbf{w}_{\text{thr}} \xrightarrow{\text{IDWT}} \mathbf{x}_{\text{thr}}$$

Given a wavelet coefficient d, we denote the thresholding operation by $d_{\text{thr}} = f(d,\lambda)$, where λ is threshold. There are various thresholding functions, but the two simplest ones are the so-called hard and soft thresholding, defined with the help of the unit-step function $u(x)$ as follows:

$$\begin{aligned} d_{\text{thr}} &= f(d,\lambda) = d\,u(|d|-\lambda) && \text{(hard)} \\ d_{\text{thr}} &= f(d,\lambda) = \operatorname{sign}(d)\,(|d|-\lambda)\,u(|d|-\lambda) && \text{(soft)} \end{aligned} \qquad (10.7.1)$$

or, equivalently,

$$d_{\text{thr}}^{\text{hard}} = \begin{cases} d\,, & |d| \ge \lambda \\ 0\,, & |d| < \lambda \end{cases}\,, \qquad d_{\text{thr}}^{\text{soft}} = \begin{cases} d - \operatorname{sign}(d)\lambda\,, & |d| \ge \lambda \\ 0\,, & |d| < \lambda \end{cases}$$

If the wavelet transform starts at level J (input length $N = 2^J$) and proceeds down to level J_0, the wavelet transform coefficients will be $\mathbf{w} = \{c_{J_0 n}\,;\, d_{jn}\,,\; J_0 \le j \le J-1\}$. The thresholding operation is applied only to the detail coefficients d_{jn}, replacing them by their thresholded values, with a possibly level-dependent threshold λ_j, that is,

$$d_{jn}^{\text{thr}} = f(d_{jn}, \lambda_j) \qquad (10.7.2)$$

The simplest possibility is to use the same threshold for all levels. Donoho & Johnstone [821] suggest the following "universal" threshold,

$$\lambda = \sigma\sqrt{2\log_2 N} \qquad \text{(universal threshold)} \qquad (10.7.3)$$

where σ^2 is the variance of the additive noise in the data. Since σ is not known, it can be estimated from the wavelet detail coefficients $\mathbf{d}_{J-1}$ at level $J-1$, which for a smooth desired signal are presumably dominated mostly by the noise component. The vector $\mathbf{d} \equiv \mathbf{d}_{J-1}$ has length $N/2 = 2^{J-1}$ and one may estimate σ by using either the standard deviation of $\mathbf{d}$, or its mean-absolute-deviation (MAD), that is,

$$\hat{\sigma} = \operatorname{std}(\mathbf{d})\,, \qquad \hat{\sigma} = \frac{\operatorname{mad}(\mathbf{d})}{0.6745} = \frac{\operatorname{median}\big(\,|\mathbf{d} - \operatorname{median}(\mathbf{d})|\,\big)}{0.6745} \qquad (10.7.4)$$

where the factor 0.6745 arises from the implicit assumption that $\mathbf{d}$ is a vector of zero-mean independent normally-distributed components (for a zero-mean, unit-variance, gaussian random variable x, one has the relationship, $\text{median}(|x|) = 0.6745$).

Donoho & Johnstone's [821] so-called VisuShrink method uses the universal threshold with the MAD estimate of σ and soft thresholding. The MATLAB function `wdenoise` implements the VisuShrink procedure, but also allows the use of hard thresholding:

```
y = wdenoise(x,h,J0,type);          % wavelet denoising
```

It is possible to derive the soft thresholding rule, as well as some of the other rules, from a regularized optimization point of view. Let $\mathbf{y}$ and $\mathbf{w} = W^T\mathbf{y}$ be the noisy data vector and its DWT, and let $\hat{\mathbf{x}}$ and $\hat{\mathbf{w}} = W^T\hat{\mathbf{y}}$ be the sought estimate and its DWT of the desired signal component $\mathbf{x}$ in the noisy signal model $\mathbf{y} = \mathbf{x} + \mathbf{v}$. An estimation criterion similar to the smoothing spline and reproducing kernel criteria that we considered earlier is the following performance index,

$$\mathcal{J} = \|\mathbf{y} - \hat{\mathbf{x}}\|^2 + P(\hat{\mathbf{x}}) = \min$$

where the first term is the L_2-norm and the second, a positive penalty term. Since the DWT matrix W is orthogonal the first term can be written in terms of the DWTs $\|\mathbf{y} - \hat{\mathbf{x}}\|^2 = \|\mathbf{w} - \hat{\mathbf{w}}\|^2$. Therefore, with a redefinition of P, we may replace the above criterion with one that is formulated in the wavelet domain:

$$\begin{aligned}\mathcal{J} &= \|\mathbf{w} - \hat{\mathbf{w}}\|^2 + P(\hat{\mathbf{w}})\\ &= \|\mathbf{c}_{J_0} - \hat{\mathbf{c}}_{J_0}\|^2 + \sum_{j=J_0}^{J-1}\|\mathbf{d}_j - \hat{\mathbf{d}}_j\|^2 + P(\hat{\mathbf{d}}_{J_0},\dots,\hat{\mathbf{d}}_{J-1}) = \min\end{aligned} \tag{10.7.5}$$

where in the second expression, we used the component representation (10.5.23), and we assumed that P depends only on the wavelet detail coefficients. The following particular choice of P using the L_1 norm leads to the soft thresholding rule:

$$\mathcal{J} = \|\mathbf{c}_{J_0} - \hat{\mathbf{c}}_{J_0}\|^2 + \sum_{j=J_0}^{J-1}\sum_{n=0}^{N_j-1}\|d_{jn} - \hat{d}_{jn}\|^2 + 2\lambda\sum_{j=J_0}^{J-1}\sum_{n=0}^{N_j-1}|\hat{d}_{jn}| = \min \tag{10.7.6}$$

where $N_j = 2^j$ is the dimension of the vector $\mathbf{d}_j$. The minimization with respect to $\mathbf{c}_{J_0}$ gives $\hat{\mathbf{c}}_{J_0} = \mathbf{c}_{J_0}$. Since the d_{jn} terms are decoupled, their minimization can be carried on a typical such term, that is, with the simple scalar criterion:

$$\mathcal{J} = |d - \hat{d}|^2 + 2\lambda|\hat{d}| = \min \tag{10.7.7}$$

whose solution is the soft-thresholding rule,

$$\hat{d} = \begin{cases} d - \text{sign}(d)\lambda\,, & |d| \ge \lambda \\ 0\,, & |d| < \lambda \end{cases} \tag{10.7.8}$$

Other variants of wavelet thresholding and other applications and uses of wavelets in statistics can be found in Refs. [819–833].

10.8 Undecimated Wavelet Transform

In this section, we discuss the undecimated wavelet transform (UWT), also known as the stationary, redundant, maximum-overlap, translation- or shift-invariant wavelet transform [748–761]. It has certain advantages over the conventional DWT exhibiting, for example, better performance in denoising applications. Its minor disadvantage is that it generates $N\log_2 N$ wavelet coefficients instead of N, and its computational cost is $O(N\log_2 N)$ instead of $O(N)$.

The essential feature of the wavelet transform is the property that successive stages of the analysis filter bank in Fig. 10.4.1 probe the frequency content of the input signal at successively lower frequency bands.

This property was depicted in Fig. 10.4.2 in which the output spectrum after three stages, shown at the bottom two graphs, was the result of filtering by the cascaded filter $\bar{H}(\omega)\bar{H}(2\omega)\bar{H}(4\omega)$, where $\omega = 2\pi f/f_s$, with f_s being the sampling rate at the finest scale. This frequency property is preserved whether the output is undecimated, as in the bottom right graph of Fig. 10.4.2, or decimated as in the left bottom graph. The reason for downsampling the outputs after each splitting stage is to keep constant the total number of samples produced by the two filters.

Fig. 10.8.1 shows the analysis bank redrawn to emphasize this frequency property. In the middle graph, all downsamplers are pushed to the overall outputs, and in the bottom graph, the downsamplers have been removed altogether.

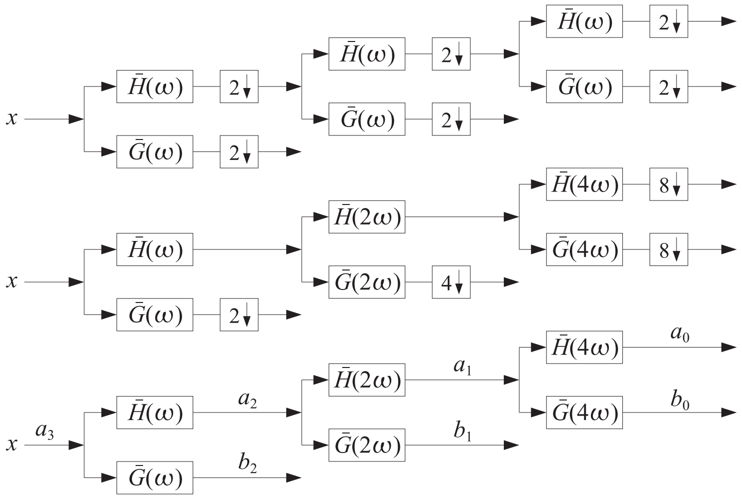


Fig. 10.8.1 Decimated and undecimated filter banks.

The bottom graph effectively implements the undecimated wavelet transform. The individual stages no longer have the orthogonality properties of the usual DWT, such as Eqs. (10.5.17). However, perfect reconstruction can still be achieved by using the property (10.3.5) for the scaling and wavelet filters:

$$\frac{1}{2}\left[\bar{H}(\omega)H(\omega)+\bar{G}(\omega)G(\omega)\right]=1 \qquad (10.8.1)$$

where $\bar{H}(\omega)=H^*(\omega)$ denotes the frequency response of the time-reversed filter $\bar{h}_n$. This relationship admits a block diagram realization as shown in Fig. 10.8.2.

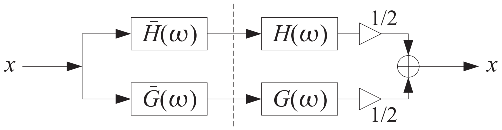


Fig. 10.8.2 Analysis and synthesis of single stage.

Because it is an identity in ω, the same relationship and block diagram will still be valid for the filter pairs $H(2\omega),G(2\omega)$ and $H(4\omega),G(4\omega)$ leading to an overall analysis and synthesis filter bank with perfect reconstruction as shown in Fig. 10.8.3.

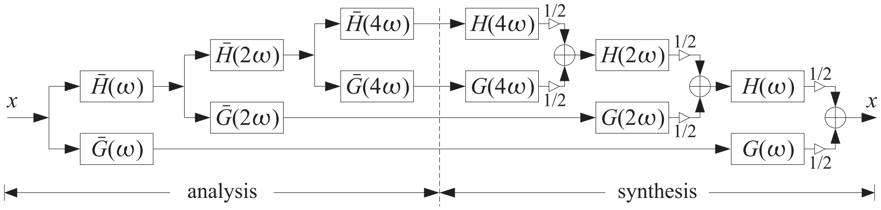


Fig. 10.8.3 Analysis and synthesis filter banks for the UWT.

Thus, it is possible with undecimated filtering operations to achieve (a) the desirable subband filter characteristics of the DWT, and (b) perfect reconstruction. To make the algorithm more concrete, first we recall that the filter with frequency response $H(2^r\omega)$ is the à trous filter defined in Eq. (10.2.20), that is,

$$h^{[r]}(k)=\sum_n h(n)\delta(k-2^r n) \quad\Leftrightarrow\quad H^{[r]}(\omega)=H(2^r\omega) \qquad (10.8.2)$$

Then, denoting the successive analysis bank output signals by $a_j(n),b_j(n)$, we obtain the following analysis and synthesis algorithm written in convolutional form:

$$\begin{aligned}
&\boxed{\begin{aligned}\mathbf{a}_{j-1}&=\bar{\mathbf{h}}^{[J-j]}*\mathbf{a}_j\\ \mathbf{b}_{j-1}&=\bar{\mathbf{g}}^{[J-j]}*\mathbf{a}_j\end{aligned}} \qquad J\ge j\ge J_0+1, \qquad \text{(analysis)}\\
&\boxed{\mathbf{a}_j=\frac{1}{2}\left[\mathbf{h}^{[J-j]}*\mathbf{a}_{j-1}+\mathbf{g}^{[J-j]}*\mathbf{b}_{j-1}\right]} \qquad J_0+1\le j\le J, \qquad \text{(synthesis)}
\end{aligned} \qquad (10.8.3)$$

where J,J_0 are the finest and coarsest desired resolution levels, and we must initialize the analysis algorithm by the overall input $a_J(n)=x(n)$. Different à trous filters are used in each stage, unlike the DWT that uses the same filters $\mathbf{h},\mathbf{g}$. The correctness of the algorithm can be verified by writing Eqs. (10.8.3) in the frequency domain and applying the identity (10.8.1).

To make the algorithm practical we may use mod-N circular convolutions, where $N=2^J$ is the length of the input signal block $\mathbf{x}$. The à trous filters $\mathbf{h}^{[r]},\mathbf{g}^{[r]}$ can be represented by $N{\times}N$ matrices H_r,G_r, which are the ordinary convolution matrices of $\mathbf{h}^{[r]},\mathbf{g}^{[r]}$ reduced modulo-N column-wise. Similarly, the time-reversed filters $\bar{\mathbf{h}}^{[r]},\bar{\mathbf{g}}^{[r]}$ will be represented by the transposed matrices H_r^T,G_r^T. The construction of these matrices is straightforward, for example,

```
h = h(:);  g = cmf(h);                               % h, g filters

hr = upr(h,r);        gr = upr(g,r);                 % upsample by 2^r
Hr = convmtx(hr, N); Gr = convmtx(gr, N);            % ordinary convolution matrices
Hr = modwrap(Hr, N); Gr = modwrap(Gr, N);            % wrapped mod-N column-wise
```

These steps have been incorporated into the function `uwtmat`, except the function `convmat` is used in place of `convmtx` to make the matrices sparse:

```
[Hr,Gr] = uwtmat(h,N,r);          % undecimated wavelet transform matrices
```

The matrices H_r,G_r satisfy the matrix version of Eq. (10.8.1):

$$\frac{1}{2}\left[H_rH_r^T+G_rG_r^T\right]=I_N \qquad (10.8.4)$$

The concrete matrix realization of the UWT can be stated then as follows:

$$\begin{aligned}
&\boxed{\begin{aligned}\mathbf{a}_{j-1}&=H_{J-j}^T\mathbf{a}_j\\ \mathbf{b}_{j-1}&=G_{J-j}^T\mathbf{a}_j\end{aligned}} \qquad J\ge j\ge J_0+1, \qquad \text{(analysis)}\\
&\boxed{\mathbf{a}_j=\frac{1}{2}\left[H_{J-j}\mathbf{a}_{j-1}+G_{J-j}\mathbf{b}_{j-1}\right]} \qquad J_0+1\le j\le J, \qquad \text{(synthesis)}
\end{aligned} \qquad (10.8.5)$$

where all the vectors are N-dimensional, initialized at $\mathbf{a}_J=\mathbf{x}$. The algorithm is illustrated in Fig. 10.8.4. The UWT is the $N{\times}(J{-}J_0{+}1)$ matrix U defined column-wise by:

$$U=[\mathbf{a}_{J_0},\,\mathbf{b}_{J_0},\,\mathbf{b}_{J_0+1},\,\dots,\,\mathbf{b}_{J-1}] \quad \text{(UWT)} \qquad (10.8.6)$$

The MATLAB functions `uwtm` and `iuwtm` implement the algorithms in Eq. (10.8.5):

```
U = uwtm(x,h,J0);        % UWT in matrix form
x = iuwtm(U,h);          % inverse UWT in matrix form
```

An example is as follows:

```
h = daub(3); x = [1 2 3 4 5 6 7 8]';
J0=0; U = uwtm(x,h,J0);                     % or, set J0 = 1 and J0 = 2
xinv = iuwtm(U,h); norm(x-xinv)
```

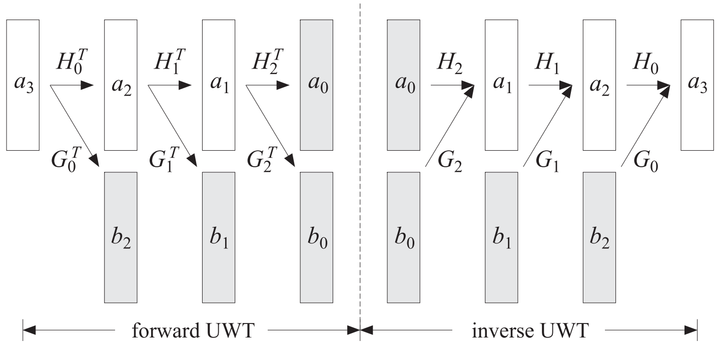


Fig. 10.8.4 Undecimated wavelet transform.

which generates, for $J_0 = 2, 1, 0$,

$$\texttt{U = uwtm(x,h,2)} = \begin{bmatrix} 2.5702 & 0 \\ 3.9844 & 0 \\ 5.3986 & 0 \\ 6.5310 & 2.6614 \\ 8.6288 & -3.7938 \\ 11.1231 & -0.1147 \\ 8.8583 & 0.9653 \\ 3.8173 & 0.2818 \end{bmatrix} = [\mathbf{a}_2, \mathbf{b}_2]$$

$$\texttt{U = uwtm(x,h,1)} = \begin{bmatrix} 7.9539 & -4.4090 & 0 \\ 11.0848 & -1.5166 & 0 \\ 12.3278 & 0.0351 & 0 \\ 12.1992 & 0.4022 & 2.6614 \\ 10.0461 & 2.2467 & -3.7938 \\ 6.9152 & 4.8818 & -0.1147 \\ 5.6722 & 2.1272 & 0.9653 \\ 5.8008 & -3.7674 & 0.2818 \end{bmatrix} = [\mathbf{a}_1, \mathbf{b}_1, \mathbf{b}_2] \tag{10.8.7}$$

$$\texttt{U = uwtm(x,h,0)} = \begin{bmatrix} 12.7279 & -1.4794 & -4.4090 & 0 \\ 12.7279 & 2.9484 & -1.5166 & 0 \\ 12.7279 & 4.7063 & 0.0351 & 0 \\ 12.7279 & 4.5243 & 0.4022 & 2.6614 \\ 12.7279 & 1.4794 & 2.2467 & -3.7938 \\ 12.7279 & -2.9484 & 4.8818 & -0.1147 \\ 12.7279 & -4.7063 & 2.1272 & 0.9653 \\ 12.7279 & -4.5243 & -3.7674 & 0.2818 \end{bmatrix} = [\mathbf{a}_0, \mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2]$$

The functions `uwtm` and `iuwtm` are somewhat slow because they generate the required matrices H_r, G_r on the fly at each stage. Of course, it would be possible to precompute the matrices and save them in a cell or a three-dimensional array as was done in the function `fwtm`. The functions `uwt` and `iuwt` are much faster versions that produce the same results and are implemented using circular convolutions:

```
U = uwt(x,h,J0);     % UWT in convolutional form
x = iuwt(U,h);       % inverse UWT
```

The following MATLAB code shows a possible implementation of the analysis part:

```
M=length(h)-1;
g=cmf(h); hR=flip(h); gR=flip(g);         % construct reversed filters
a = x;                                    % x is N = 2^J dimensional column vector
for r=0:J-J0-1,                           % à trous interpolation factor is 2^r, level j = J − r
   a = advance(a, 2^r*M);                 % establishes equivalence with matrix form
   hRr = upr(hR,r);                       % reversed filters upsampled by 2^r
   gRr = upr(gR,r);
   b = circonv(gRr,a,N);                  % modulo-N circular convolution
   a = circonv(hRr,a,N);
   U = [b,U];                             % accumulate the columns of U
end
U = [a,U];
```

The time-advancing operation is necessary to compensate for the use of the reversed filters rather than the time-reversed ones. Although this algorithm works, it is wasteful because the à trous filters $\mathbf{h}^{[r]}$ have length $2^r(M+1)$ consisting mostly of zeros and only $(M+1)$ nonzero coefficients, where M is the filter order of $\mathbf{h}$. The computational cost of the indicated circular convolution operations is of the order of $2^r(M+1)N$. It is possible to restructure these operations so that only the nonzero filter coefficients are used, thereby reducing the computational cost to $(M+1)N$. Ordinary and mod-N circular convolution by the à trous filter (10.8.2) can be written as follows:

$$\begin{aligned} y(n) &= \sum_k h^{[r]}(k)x(n-k) = \sum_m h_m x(n - 2^r m) \\ \tilde{y}(n)_{\text{mod-}N} &= \sum_p y(n+pN) = \sum_{p,m} h_m x(n + pN - 2^r m) \end{aligned}$$

We assume that $N = 2^J$ and that the à trous factor is such that $r \le J$, so that we may write $N = 2^r L$, where $L = 2^{J-r}$. Thus, a length-N block can be divided into 2^r sub-blocks of length L. We show below that the mod-N circular convolution can be replaced by 2^r mod-L circular convolutions. The total computational cost reduces then to $2^r L(M+1) = N(M+1)$. Setting $n = 2^r i + k$, with $0 \le k \le 2^r - 1$, we may define the k-th sub-block input and output signals:

$$x_k(i) = x(2^r i + k)\,, \quad y_k(i) = y(2^r i + k)\,, \qquad 0 \le k \le 2^r - 1$$

It follows then that $y_k(i)$ is the convolution of $x_k(i)$ with the original filter h_m, and that the mod-N circular convolution output can be obtained by mod-L reduction:

$$\begin{aligned} y_k(i) = y(2^r i + k) &= \sum_m h_m x(2^r i + k - 2^r m) = \sum_m h_m x_k(i-m) \\ \tilde{y}(2^r i + k)_{\text{mod-}N} &= \sum_{p,m} h_m x(2^r i + k + 2^r pL - 2^r m) = \sum_{p,m} h_m x_k(i + pL - m) \\ &= \sum_p y_k(i + pL) = \tilde{y}_k(i)_{\text{mod-}L} \end{aligned}$$

These operations have been incorporated into the MATLAB function `convat`,

```
y = convat(h,x,r);        % convolution à trous
```

which is equivalent to the mod-N operation, where N is the length of x:

```
y = circonv(upr(h,r),x,N);
```

The essential part of the function uwt is then,

```
M=length(h)-1;
g=cmf(h); hR=flip(h); gR=flip(g);        % construct reversed filters
a = x;                                   % x is N = 2^J dimensional column vector
for r=0:J-J0-1,                          % à trous interpolation factor is 2^r, level j = J - r
   a = advance(a, 2^r*M);                % establishes equivalence with matrix form
   b = convat(gR, a, r);                 % convolution à trous
   a = convat(hR, a, r);
   U = [b,U];                            % accumulate the columns of U
end
U = [a,U];
```

Because all stages of the analysis and synthesis filter banks in Fig. 10.8.3 operate at the same sampling rate, the UWT satisfies a time-invariance property, in the sense that a time-delay in the input will cause the same delay in the outputs from all stages. Hence, the alternative name "stationary" or "translation-invariant" wavelet transform. Such time-invariance property is not shared by the ordinary DWT.

As an example, the DWTs and UWTs of a signal and its circularly-delayed version by three time units are as follows, using the D_3 scaling filter and coarsest level $J_0 = 0$:

$$\mathbf{x} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \end{bmatrix} \;\Rightarrow\; \mathbf{w} = \begin{bmatrix} 12.7279 \\ -1.4794 \\ -4.4090 \\ 2.2467 \\ 0 \\ 0 \\ -3.7938 \\ 0.9653 \end{bmatrix} \;\Rightarrow\; U = \begin{bmatrix} 12.7279 & -1.4794 & -4.4090 & 0 \\ 12.7279 & 2.9484 & -1.5166 & 0 \\ 12.7279 & 4.7063 & 0.0351 & 0 \\ 12.7279 & 4.5243 & 0.4022 & 2.6614 \\ 12.7279 & 1.4794 & 2.2467 & -3.7938 \\ 12.7279 & -2.9484 & 4.8818 & -0.1147 \\ 12.7279 & -4.7063 & 2.1272 & 0.9653 \\ 12.7279 & -4.5243 & -3.7674 & 0.2818 \end{bmatrix}$$

$$\mathbf{x} = \begin{bmatrix} 6 \\ 7 \\ 8 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{bmatrix} \;\Rightarrow\; \mathbf{w} = \begin{bmatrix} 12.7279 \\ -2.9484 \\ 4.8818 \\ -1.5166 \\ -0.1147 \\ 0.2818 \\ 0 \\ 2.6614 \end{bmatrix} \;\Rightarrow\; U = \begin{bmatrix} 12.7279 & -2.9484 & 4.8818 & -0.1147 \\ 12.7279 & -4.7063 & 2.1272 & 0.9653 \\ 12.7279 & -4.5243 & -3.7674 & 0.2818 \\ 12.7279 & -1.4794 & -4.4090 & 0 \\ 12.7279 & 2.9484 & -1.5166 & 0 \\ 12.7279 & 4.7063 & 0.0351 & 0 \\ 12.7279 & 4.5243 & 0.4022 & 2.6614 \\ 12.7279 & 1.4794 & 2.2467 & -3.7938 \end{bmatrix}$$

We note that every column of U gets delayed circularly by three time units.

Multiresolution Decomposition with the UWT

The synthesis filter bank or the synthesis algorithm for the UWT can be viewed as a system with $J-J_0+1$ inputs, i.e., the columns of $U = [\mathbf{a}_{J_0}, \mathbf{b}_{J_0}, \mathbf{b}_{J_0+1}, \dots, \mathbf{b}_{J-1}]$, and

one output, the signal $\mathbf{x}$. The UWT multiresolution decomposition resolves $\mathbf{x}$ into components arising from the individual inputs when the other inputs are zero:

$$\begin{aligned} U = [\mathbf{a}_{J_0}, \mathbf{b}_{J_0}, \mathbf{b}_{J_0+1}, \dots, \mathbf{b}_{J-1}] &\xrightarrow{\text{IUWT}} \mathbf{x} \\ = [\mathbf{a}_{J_0}, 0, \quad 0, \quad \dots, \quad 0\ ] &\xrightarrow{\text{IUWT}} \mathbf{x}_{J_0} \\ +\,[\ 0, \quad \mathbf{b}_{J_0}, \quad 0, \quad \dots, \quad 0\ ] &\xrightarrow{\text{IUWT}} \bar{\mathbf{x}}_{J_0} \\ +\,[\ 0, \quad 0, \quad \mathbf{b}_{J_0+1}, \dots, \quad 0\ ] &\xrightarrow{\text{IUWT}} \bar{\mathbf{x}}_{J_0+1} \\ \cdots \qquad & \qquad \cdots \\ +\,[\ 0, \quad 0, \quad 0, \quad \dots, \mathbf{b}_{J-1}] &\xrightarrow{\text{IUWT}} \bar{\mathbf{x}}_{J-1} \end{aligned}$$

so that we have the sum,

$$\boxed{\mathbf{x} = \mathbf{x}_{J_0} + \bar{\mathbf{x}}_{J_0} + \bar{\mathbf{x}}_{J_0+1} + \cdots + \bar{\mathbf{x}}_{J-1}} \tag{10.8.8}$$

This is similar to the DWT decomposition (10.6.2), with each term reflecting a different resolution level, except that the terms are not mutually orthogonal. The MATLAB function uwtdec implements this decomposition:

```
X = uwtdec(x,h,J0);        % UWT multiresolution decomposition
```

where X consists of the columns, $X = [\mathbf{x}_{J_0}, \bar{\mathbf{x}}_{J_0}, \bar{\mathbf{x}}_{J_0+1}, \dots, \bar{\mathbf{x}}_{J-1}]$.

Fig. 10.8.5 shows an application to the monthly housing starts from January 1988 to April 2009 (i.e., 256 months), using the symmlet S_8 scaling filter and going down to resolution level $J_0 = 5$. This a subset of the dataset that we used repeatedly in Chap. 9.

The upper left graph shows the smooth component arising from the UWT coefficients $\mathbf{a}_{J_0}$. The remaining graphs arise from the detail coefficients. $\mathbf{b}_j$, $J_0 \le j \le J-1$. The sum of the four components is equal to the original data (dotted line in the upper-left graph.) The following MATLAB code generates the four graphs:

```
Y = loadfile('newhouse.dat');                % data file in OSP toolbox

y = Y(349:end,1);                            % selects Jan.88 - Apr.09 = 256 months
t = taxis(y,12,1988)';                       % adjust time axis

h=daub(8,2); J0=5;                           % symmlet S_8, note N = 256 = 2^8  ⇒  J = 8
X = uwtdec(y,h,J0);                          % UWT decomposition, try also X = dwtdec(y,h,J0)

figure; plot(t,y,':', t,X(:,1), '-');                  % upper left graph
figure; plot(t,X(:,2));                                % upper right
figure; plot(t,X(:,3)); figure; plot(t,X(:,4));        % lower graphs
```

See Fig. 10.9.1 for an alternative way of plotting the UWT (or DWT) decomposition and the UWT (or DWT) wavelet coefficients using the function plotdec.

Wavelet Denoising with the UWT

The application of the UWT to denoising applications follows the same approach as the DWT. The detail columns $\mathbf{b}_j$ of U get thresholded by a possibly level-dependent threshold and the inverse UWT is constructed. The procedure is depicted below:

$$\mathbf{x} \xrightarrow{\text{UWT}} U = [\mathbf{a}_{J_0}, \mathbf{b}_{J_0}, \dots, \mathbf{b}_{J-1}] \xrightarrow{\text{thresh}} U^{\text{thr}} = [\mathbf{a}_{J_0}, \mathbf{b}_{J_0}^{\text{thr}}, \dots, \mathbf{b}_{J-1}^{\text{thr}}] \xrightarrow{\text{IUWT}} \mathbf{x}_{\text{thr}}$$

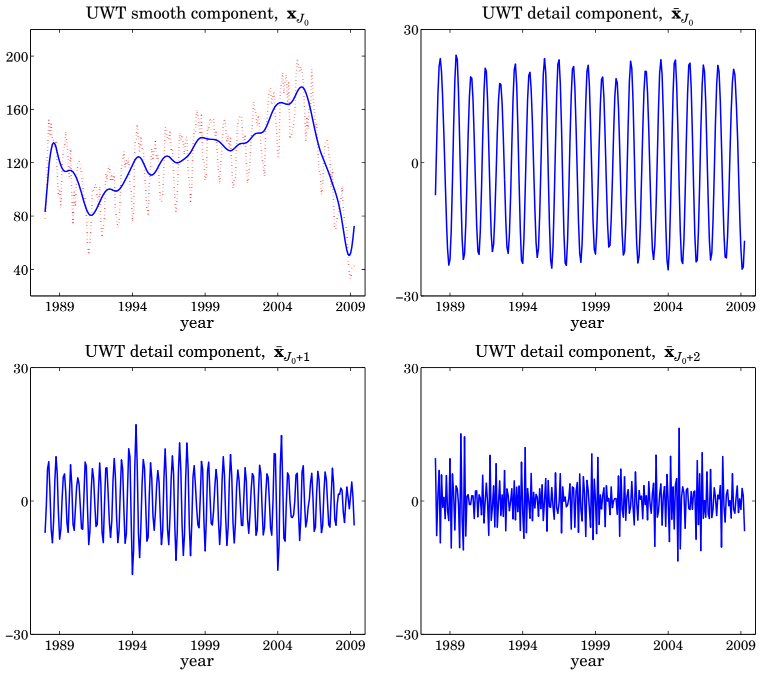


Fig. 10.8.5 UWT decomposition of monthly housing data, using S_8 with $J = 8$ and $J_0 = 5$.

The MATLAB function wduwt implements this denoising procedure using the universal threshold (10.7.3) and soft or hard thresholding:

```
y = wduwt(x,h,J0,type);                % wavelet denoising with UWT
```

Fig. 10.8.6 shows the same denoising example as that in Fig. 10.7.1, but denoised using the UWT. The following MATLAB code generates the top-row graphs:

```
F = inline('1./(1 + abs(x)).^4');                 % bumps function

N = 2048; t = (0:N-1)'/N; x = zeros(size(t));     % normalize time to 0 ≤ t ≤ 1

t0 = [10 13 15 23 25 40 44 65 76 78 81]/100;      % signal parameters
a = [40,50,30,40,50,42,21,43,31,51,42] * 1.0523;
w = [5,5,6,10,10,30,10,10,5,8,5]/1000;

for i=1:length(a),                                % construct noise-free signal
   x = x + a(i) * F((t-t0(i))/w(i));
end

seed=2009; randn('state',seed); v = randn(size(t));   % generate noise

y = x + v;                                        % noisy signal with SNR, σx/σv = 7
```

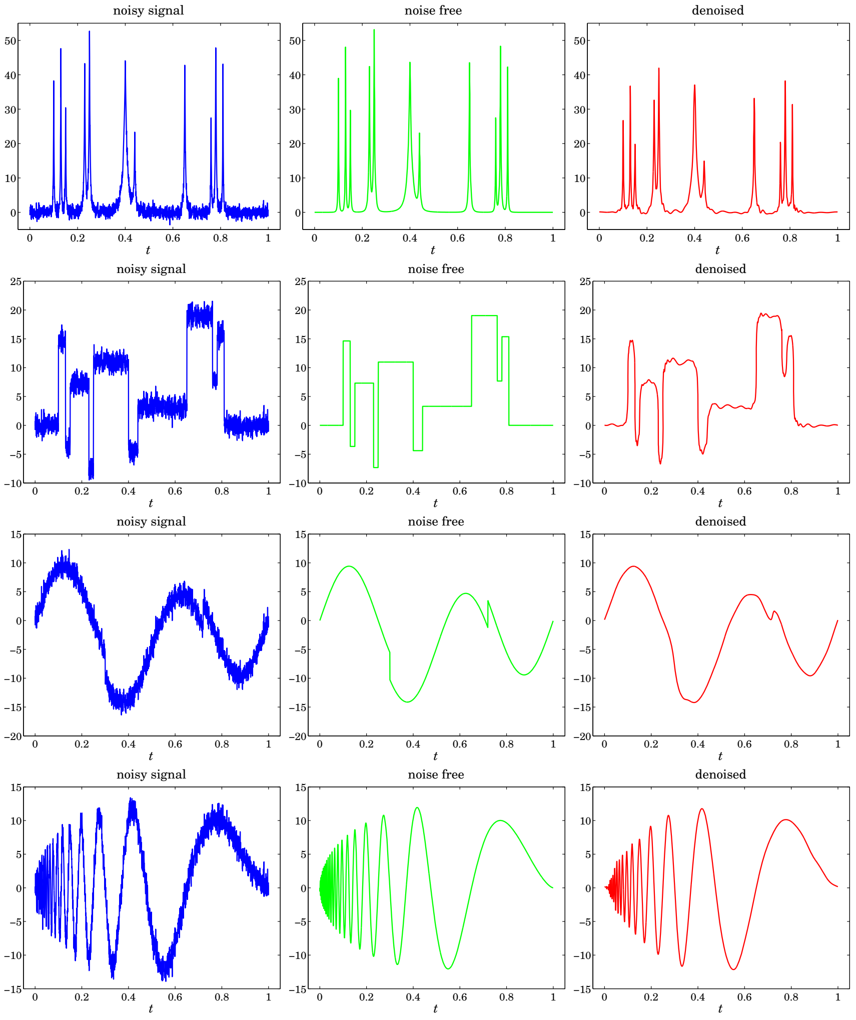


Fig. 10.8.6 Wavelet denoising with the UWT.

```
h = daub(8,2); J0=5; type=1;                      % use Symmlet-8 and soft thresholding
xd = wduwt(y,h,J0,type);                          % wavelet denoising using the UWT

figure; plot(t,y,'-'); figure; plot(t,x,'-'); figure; plot(t,xd,'r-');   % top row
```

Comparing Figs. 10.7.1 and 10.8.6, we note that the UWT outperforms the DWT, an observation that has been made repeatedly in the denoising literature.

10.9 MATLAB Functions

We summarize the MATLAB functions that we discussed in this chapter, and give few more details about `plotdec` that can be used to plot wavelet decompositions and wavelet coefficients.

Wavelet functions

`advance`	– circular time-advance (left-shift) of a vector
`casc`	– cascade algorithm for ϕ and ψ wavelet functions
`circonv`	– circular convolution
`cmf`	– conjugate mirror of a filter
`convat`	– convolution à trous
`convmat`	– sparse convolution matrix
`daub`	– Daubechies scaling filters (daublets, symmlets, coiflets)
`dn2`	– downsample by a factor of 2
`dwtcell`	– create cell array of sparse discrete wavelet transform matrices
`dwtdec`	– DWT decomposition into orthogonal multiresolution components
`dwtmat`	– discrete wavelet transform matrices - sparse
`dwtmat2`	– discrete wavelet transform matrices - nonsparse
`dwtwrap`	– wrap a DWT matrix into a lower DWT matrix
`flip`	– flip a column, a row, or both
`fwt`	– fast wavelet transform using convolution and downsampling
`fwtm`	– fast wavelet transform in matrix form
`fwtmat`	– overall DWT orthogonal matrix
`ifwt`	– inverse fast wavelet transform - convolutional form
`ifwtm`	– inverse fast wavelet transform - matrix form
`iuwt`	– inverse undecimated wavelet transform - convolutional form
`iuwtm`	– inverse undecimated wavelet transform - matrix form
`modwrap`	– wrap matrix column-wise mod-N
`phinit`	– eigenvector initialization of scaling function ϕ
`plotdec`	– plot DWT/UWT decomposition or DWT/UWT coefficients
`up2`	– upsample a vector by factor of 2
`upr`	– upsample a vector by factor of 2^r
`uwt`	– undecimated wavelet transform - convolutional form
`uwtdec`	– UWT multiresolution decomposition
`uwtm`	– undecimated wavelet transform - matrix form
`uwtmat`	– undecimated wavelet transform matrices - sparse
`uwtmat2`	– undecimated wavelet transform matrices - nonsparse
`w2V`	– wavelet vector to wavelet matrix
`wcoeff`	– extract wavelet coefficients from DWT at given level
`wdenoise`	– Donoho & Johnstone's VisuShrink denoising procedure
`wduwt`	– wavelet denoising with undecimated wavelet transform
`wthr`	– soft/hard level-dependent wavelet thresholding

The function `plotdec` allows a compact display of a DWT or UWT decomposition, or the display of the DWT/UWT wavelet smooth and detail coefficients:

```
plotdec(X,type,lin,Jmax);      % plot DWT/UWT decomposition or DWT/UWT coefficients
```

with inputs:

`X` = $N\times(J-J_0+1)$ matrix of DWT/UWT decomposition signals or DWT/UWT coefficients, $N = 2^J$
`type` = `'xs','xd','ws','wd'` (x=decomposition, w=wavelet coeffs, s=include smooth, d=details only)
`Jmax` = highest resolution level to plot, $J_{\max} \le J-1$, minimum is determined from $J_0 = J+1-\text{size}(X,2)$
`lin` = `'l','s'` for line or stem plot

See the help for this function for several usage examples. Fig. 10.9.1 shows an alternative plot of the UWT decomposition of Fig. 10.8.5, showing only the detail components, including a plot of the UWT wavelet coefficients. The MATLAB code used to generate the four graphs was as follows:

```
h=daub(8,2); J0=5;
Y = loadfile('newhouse.dat');          % load housing data - file in OSP toolbox
y = Y(349:end,1);                      % selects Jan.88 - Apr.09 = 256 months

Xdwt = dwtdec(y,h,J0);                 % DWT decomposition
[w,V] = fwt(y,h,J0);                   % DWT coefficients
Xuwt = uwtdec(y,h,J0);                 % UWT decomposition
U = uwt(y,h,J0);                       % UWT coefficients

figure; plotdec(Xdwt,'xd');  figure; plotdec(V,'wd');      % upper graphs
figure; plotdec(Xuwt,'xd');  figure; plotdec(U,'wd');      % lower graphs
```

10.10 Problems

10.1 An alternative way of determining the Daubechies D_2 scaling filter is to assume that its transfer function has the form (with $K = 2$ zeros at Nyquist):

$$H(z) = h_0(1+z^{-1})^2(1-z_1z^{-1})$$

Show that z_1 must the be a solution of the quadratic equation $z^2 - 4z + 1 = 0$. Pick that solution that has $|z_1| < 1$ and verify that the resulting filter $H(z)$ meets all the design constraints (10.3.10).

10.2 To determining the Daubechies D_3 scaling filter assume that its its transfer function has the following form with $K = 3$ zeros at Nyquist:

$$H(z) = h_0(1+z^{-1})^3\big(1-(a+jb)z^{-1}\big)\big(1-(a-jb)z^{-1}\big)$$

including a complex zero $z_1 = a + jb$, constrained such that $|z_1| < 1$. Show that the design constraints (10.3.12) imply that b is given by

$$b = \sqrt{\frac{a + a^3 - 3a^2}{3-a}}$$

and that a is obtained as the following solution of the quartic equation:

$$12a^4 - 72a^3 + 152a^2 - 132a + 27 = 0 \quad\Rightarrow\quad a = \frac{3}{2} - \frac{1}{6}\sqrt{15+12\sqrt{10}}$$

Verify that the zero $z_1 = a + jb$ coincides with that in Eq. (10.3.18). By expanding $H(z)$, express the filter coefficients h_n in terms of a, b, and normalize them to add up to $\sqrt{2}$.

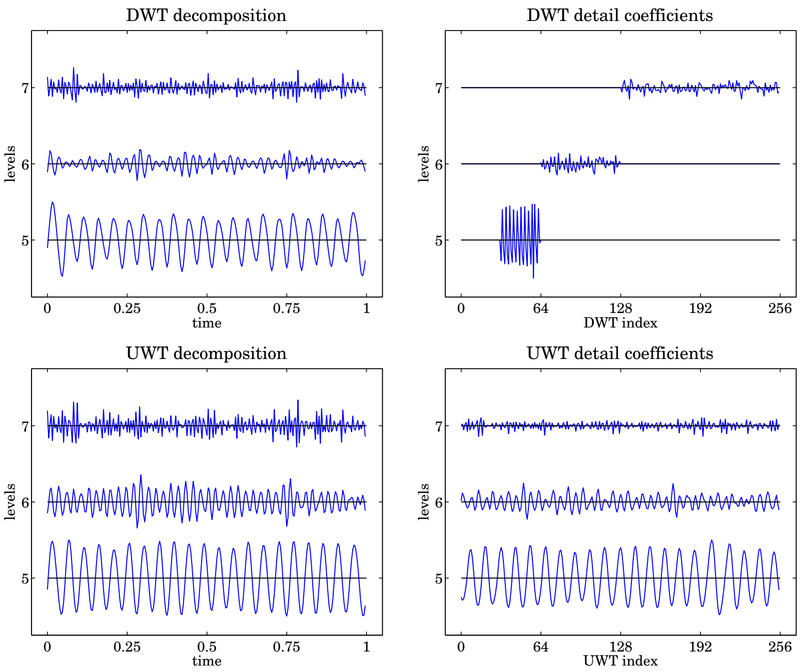


Fig. 10.9.1 UWT/DWT decompositions and wavelet coefficients of housing data.

10.3 Prove the downsampling replication property (10.4.11) by working backwards, that is, start from the Fourier transform expression and show that

$$\frac{1}{L}\sum_{m=0}^{L-1} X(f - mf_s^{\text{down}}) = \sum_k s(k)x(k)e^{-2\pi jfk/f_s} = \sum_n x(nL)e^{-2\pi jfnL/f_s} = Y_{\text{down}}(f)$$

where $s(k)$ is the periodic "sampling function" with the following representations:

$$s(k) = \frac{1}{L}\sum_{m=0}^{L-1} e^{-2\pi jkm/L} = \frac{1}{L}\,\frac{1-e^{-2\pi jk}}{1-e^{-2\pi jk/L}} = \sum_n \delta(k-nL)$$

Moreover, show that the above representations are nothing but the inverse L-point DFT of the DFT of one period of the periodic pulse train:

$$s(k) = [\dots, 1, \underbrace{0, 0, \dots, 0}_{L-1 \text{ zeros}}, 1, \underbrace{0, 0, \dots, 0}_{L-1 \text{ zeros}}, 1, \underbrace{0, 0, \dots, 0}_{L-1 \text{ zeros}}, \dots] = \sum_n \delta(k-nL)$$

10.4 Show that the solution to the optimization problem (10.7.7) is the soft-thresholding rule of Eq. (10.7.8).

10.5 Study the "Tikhonov regularizer" wavelet thresholding function:

$$d_{\text{thr}} = f(d, \lambda, a) = d\,\frac{|d|^a}{|d|^a + \lambda^a}\,, \quad a > 0,\ \lambda > 0$$

11

Wiener Filtering

The problem of estimating one signal from another is one of the most important in signal processing. In many applications, the desired signal is not available or observable directly. Instead, the observable signal is a degraded or distorted version of the original signal. The signal estimation problem is to recover, in the best way possible, the desired signal from its degraded replica.

We mention some typical examples: (1) The desired signal may be corrupted by strong additive noise, such as weak evoked brain potentials measured against the strong background of ongoing EEGs; or weak radar returns from a target in the presence of strong clutter. (2) An antenna array designed to be sensitive towards a particular "look" direction may be vulnerable to strong jammers from other directions due to sidelobe leakage; the signal processing task here is to null the jammers while at the same time maintaining the sensitivity of the array towards the desired look direction. (3) A signal transmitted over a communications channel can suffer phase and amplitude distortions and can be subject to additive channel noise; the problem is to recover the transmitted signal from the distorted received signal. (4) A Doppler radar processor tracking a moving target must take into account dynamical noise—such as small purely random accelerations—affecting the dynamics of the target, as well as measurement errors. (5) An image recorded by an imaging system is subject to distortions such as blurring due to motion or to the finite aperture of the system, or other geometric distortions; the problem here is to undo the distortions introduced by the imaging system and restore the original image. A related problem, of interest in medical image processing, is that of reconstructing an image from its projections. (6) In remote sensing and inverse scattering applications, the basic problem is, again, to infer one signal from another; for example, to infer the temperature profile of the atmosphere from measurements of the spectral distribution of infrared energy; or to deduce the structure of a dielectric medium, such as the ionosphere, by studying its response to electromagnetic wave scattering; or, in oil exploration to infer the layered structure of the earth by measuring its response to an impulsive input near its surface.

In this chapter, we pose the signal estimation problem and discuss some of the criteria used in the design of signal estimation algorithms.

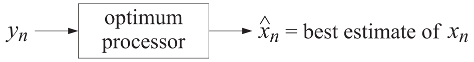
We do not present a complete discussion of all methods of signal recovery and estimation that have been invented for applications as diverse as those mentioned above.

Our emphasis is on traditional linear least-squares estimation methods, not only because they are widely used, but also because they have served as the motivating force for the development of other estimation techniques and as the yardstick for evaluating them.

We develop the theoretical solution of the Wiener filter both in the stationary and nonstationary cases, and discuss its connection to the orthogonal projection, Gram-Schmidt constructions, and correlation canceling ideas of Chap. 1. By means of an example, we introduce Kalman filtering concepts and discuss their connection to Wiener filtering and to signal modeling. Practical implementations of the Wiener filter are discussed in Chapters 12 and 16. Other signal recovery methods for deconvolution applications that are based on alternative design criteria are briefly discussed in Chap. 12, where we also discuss some interesting connections between Wiener filtering/linear prediction methods and inverse scattering methods.

11.1 Linear and Nonlinear Estimation of Signals

The signal estimation problem can be stated as follows: We wish to estimate a random signal x_n on the basis of available observations of a related signal y_n. The available signal y_n is to be processed by an optimal processor that produces the best possible estimate of x_n:



The resulting estimate $\hat{x}_n$ will be a function of the observations y_n. If the optimal processor is linear, such as a linear filter, then the estimate $\hat{x}_n$ will be a linear function of the observations. We are going to concentrate mainly on linear processors. However, we would like to point out that, depending on the estimation criterion, there are cases where the estimate $\hat{x}_n$ may turn out to be a nonlinear function of the y_ns.

We discuss briefly four major estimation criteria for designing such optimal processors. They are:

(1) The maximum a posteriori (MAP) criterion.
(2) The maximum likelihood (ML) criterion.
(3) The mean square (MS) criterion.
(4) The linear mean-square (LMS) criterion.

The LMS criterion is a special case of the MS criterion. It requires, *a priori*, that the estimate $\hat{x}_n$ be a *linear* function of the y_ns.[†] The main advantage of the LMS processor is that it requires only knowledge of second order statistics for its design, whereas the other, nonlinear, processors require more detailed knowledge of probability densities.

To explain the various estimation criteria, let us assume that the desired signal x_n is to be estimated over a finite time interval $n_a \le n \le n_b$ Without loss of generality, we may assume that the observed signal y_n is also available over the same interval. Define the vectors

[†] Note that the acronym LMS is also used in the context of adaptive filtering, for *least mean-square*.

$$\mathbf{x} = \begin{bmatrix} x_{n_a} \\ x_{n_a+1} \\ \vdots \\ x_{n_b} \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_{n_a} \\ y_{n_a+1} \\ \vdots \\ y_{n_b} \end{bmatrix}$$

For each value of n, we seek the functional dependence

$$\hat{x}_n = \hat{x}_n(\mathbf{y})$$

of $\hat{x}_n$ on the given observation vector $\mathbf{y}$ that provides the *best estimate* of x_n.

1. The criterion for the MAP estimate is to maximize the a posteriori conditional density of x_n given that $\mathbf{y}$ already occurred; namely,

$$p(x_n|\mathbf{y}) = \text{maximum} \tag{11.1.1}$$

in other words, the optimal estimate $\hat{x}_n$ is that x_n that maximizes this quantity for the given vector $\mathbf{y}$; $\hat{x}_n$ is therefore the most probable choice resulting from the given observations $\mathbf{y}$.

2. The ML criterion, on the other hand, selects $\hat{x}_n$ to maximize the conditional density of $\mathbf{y}$ given x_n, that is,

$$p(\mathbf{y}|x_n) = \text{maximum} \tag{11.1.2}$$

This criterion selects $\hat{x}_n$ as though the already collected observations $\mathbf{y}$ were the most likely to occur.

3. The MS criterion minimizes the mean-square estimation error

$$\mathcal{E} = E[e_n^2] = \text{min}, \quad \text{where} \quad e_n = x_n - \hat{x}_n \tag{11.1.3}$$

that is, the best choice of the functional dependence $\hat{x}_n = \hat{x}_n(\mathbf{y})$ is sought that minimizes this expression. We know from our results of Sec. 1.4 that the required solution is the corresponding *conditional mean*

$$\hat{x}_n = E[x_n|\mathbf{y}] = \text{MS estimate} \tag{11.1.4}$$

computed with respect to the conditional density $p(x_n|\mathbf{y})$.

4. Finally, the LMS criterion requires the estimate to be a linear function of the observations

$$\hat{x}_n = \sum_{i=n_a}^{n_b} h(n,i)y_i \tag{11.1.5}$$

For each n, the weights $h(n,i)$, $n_a \le i \le n_b$ are selected to *minimize* the mean-square estimation error

$$\mathcal{E} = E[e_n^2] = E\big[(x_n - \hat{x}_n)^2\big] = \text{minimum} \tag{11.1.6}$$

With the exception of the LMS estimate, all other estimates $\hat{x}_n(\mathbf{y})$ are, in general, nonlinear functions of $\mathbf{y}$.

Example 11.1.1: If both x_n and $\mathbf{y}$ are zero-mean and jointly gaussian, then Examples 1.4.1 and 1.4.2 imply that the MS and LMS estimates of x_n are the *same*. Furthermore, since $p(x_n|\mathbf{y})$ is gaussian it will be symmetric about its maximum, which occurs at its mean, that is, at $E[x_n|\mathbf{y}]$. Therefore, the MAP estimate of x_n is equal to the MS estimate. In conclusion, for *zero-mean jointly gaussian* x_n and $\mathbf{y}$, the three estimates MAP, MS, and LMS coincide. □

Example 11.1.2: To see the nonlinear character and the differences among the various estimates, consider the following example: A discrete-amplitude, constant-in-time signal x can take on the three values

$$x=-1\,,\quad x=0\,,\quad x=1$$

each with probability of $1/3$. This signal is placed on a known carrier waveform c_n and transmitted over a noisy channel. The received samples are of the form

$$y_n = c_n x + v_n\,,\quad n = 1,2,\dots,M$$

where v_n are zero-mean white gaussian noise samples of variance σ_v^2, assumed to be independent of x. The above set of measurements can be written in an obvious vector notation

$$\mathbf{y} = \mathbf{c}\,x + \mathbf{v}$$

(a) Determine the conditional densities $p(\mathbf{y}|x)$ and $p(x|\mathbf{y})$.

(b) Determine and compare the four alternative estimates MAP, ML, MS, and LMS.

Solution: To compute $p(\mathbf{y}|x)$, note that if x is given, then the only randomness left in $\mathbf{y}$ arises from the noise term $\mathbf{v}$. Since v_n are uncorrelated and gaussian, they will be independent; therefore,

$$\begin{aligned}
p(\mathbf{y}|x) &= p(\mathbf{v}) = \prod_{n=1}^{M} p(v_n) = (2\pi\sigma_v^2)^{-M/2}\exp\left[-\frac{1}{2\sigma_v^2}\sum_{n=1}^{M}v_n^2\right] \\
&= (2\pi\sigma_v^2)^{-M/2}\exp\left[-\frac{1}{2\sigma_v^2}\mathbf{v}^2\right] = (2\pi\sigma_v^2)^{-M/2}\exp\left[-\frac{1}{2\sigma_v^2}(\mathbf{y}-\mathbf{c}\,x)^2\right]
\end{aligned}$$

Using Bayes' rule we find $p(x|\mathbf{y}) = p(\mathbf{y}|x)p(x)/p(\mathbf{y})$. Since

$$p(x) = \frac{1}{3}\big[\delta(x-1)+\delta(x)+\delta(x+1)\big]$$

we find

$$p(x|\mathbf{y}) = \frac{1}{A}\big[p(\mathbf{y}|1)\delta(x-1)+p(\mathbf{y}|0)\delta(x)+p(\mathbf{y}|-1)\delta(x+1)\big]$$

where the constant A is

$$A = 3p(\mathbf{y}) = 3\int p(\mathbf{y}|x)p(x)\,dx = p(\mathbf{y}|1)+p(\mathbf{y}|0)+p(\mathbf{y}|-1)$$

To find the MAP estimate of x, the quantity $p(x|\mathbf{y})$ must be maximized with respect to x. Since the expression for $p(x|\mathbf{y})$ forces x to be one of the three values $+1, 0, -1$, it follows that the maximum among the three coefficients $p(\mathbf{y}|1)$, $p(\mathbf{y}|0)$, $p(\mathbf{y}|-1)$ will determine the value of x. Thus, for a given $\mathbf{y}$ we select that x that

$$p(\mathbf{y}|x) = \text{maximum of}\{p(\mathbf{y}|1),\,p(\mathbf{y}|0),\,p(\mathbf{y}|-1)\}$$

Using the gaussian nature of $p(\mathbf{y}|x)$, we find equivalently

$$(\mathbf{y}-\mathbf{c}\,x)^2 = \text{minimum of}\{(\mathbf{y}-\mathbf{c})^2,\,\mathbf{y}^2,\,(\mathbf{y}+\mathbf{c})^2\}$$

Subtracting $\mathbf{y}^2$ from both sides, dividing by $\mathbf{c}^T\mathbf{c}$, and denoting

$$\bar{y} = \frac{\mathbf{c}^T\mathbf{y}}{\mathbf{c}^T\mathbf{c}}$$

we find the equivalent equation

$$x^2 - 2x\bar{y} = \min\{1-2\bar{y},\,0,\,1+2\bar{y}\}$$

and in particular, applying these for $+1, 0, -1$, we find

$$\hat{x}_{\text{MAP}} = \begin{cases} 1, & \text{if } \bar{y} > \dfrac{1}{2} \\ 0, & \text{if } -\dfrac{1}{2} < \bar{y} < \dfrac{1}{2} \\ -1, & \text{if } \bar{y} < -\dfrac{1}{2} \end{cases}$$

To determine the ML estimate, we must maximize $p(\mathbf{y}|x)$ with respect to x. The ML estimate does not require knowledge of the a priori probability density $p(x)$ of x. Therefore, differentiating $p(\mathbf{y}|x)$ with respect to x and setting the derivative to zero gives

$$\frac{\partial}{\partial x}p(\mathbf{y}|x) = 0 \quad\text{or}\quad \frac{\partial}{\partial x}\ln p(\mathbf{y}|x) = 0 \quad\text{or}\quad \frac{\partial}{\partial x}(\mathbf{y}-\mathbf{c}\,x)^2 = 0$$

which gives

$$\hat{x}_{\text{ML}} = \frac{\mathbf{c}^T\mathbf{y}}{\mathbf{c}^T\mathbf{c}} = \bar{y}$$

The MS estimate is obtained by computing the conditional mean

$$\begin{aligned}
E[x|\mathbf{y}] &= \int x p(x|\mathbf{y})\,dx = \int x\frac{1}{A}\big[p(\mathbf{y}|1)\delta(x-1)+p(\mathbf{y}|0)\delta(x)+p(\mathbf{y}|-1)\delta(x+1)\big]dx \\
&= \frac{1}{A}\big[p(\mathbf{y}|1)-p(\mathbf{y}|-1)\big]\,,\quad\text{or,}
\end{aligned}$$

$$\hat{x}_{\text{MS}} = \frac{p(\mathbf{y}|1)-p(\mathbf{y}|-1)}{p(\mathbf{y}|1)+p(\mathbf{y}|0)+p(\mathbf{y}|-1)}$$

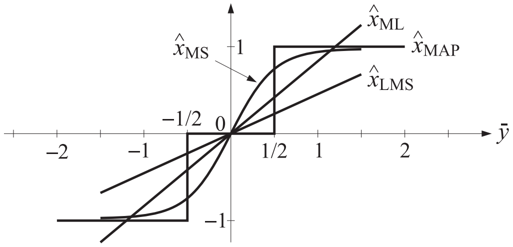
Canceling some common factors from the numerator and denominator, we find the simpler expression

$$\hat{x}_{\rm MS} = \frac{2\sinh(2a\bar{y})}{e^a + 2\cosh(2a\bar{y})}\,, \quad \text{where} \quad a = \frac{\mathbf{c}^T\mathbf{c}}{2\sigma_v^2}$$

Finally, the LMS estimate can be computed as in Example 1.4.3. We find

$$\hat{x}_{\rm LMS} = \frac{\mathbf{c}^T\mathbf{y}}{\dfrac{\sigma_v^2}{\sigma_x^2} + \mathbf{c}^T\mathbf{c}} = \frac{\mathbf{c}^T\mathbf{c}}{\dfrac{\sigma_v^2}{\sigma_x^2} + \mathbf{c}^T\mathbf{c}}\,\bar{y}$$

All four estimates have been expressed in terms of $\bar{y}$. Note that the ML estimate is linear but has a different slope than the LMS estimate. The nonlinearity of the various estimates is best seen in the following figure:



11.2 Orthogonality and Normal Equations

From now on, we will concentrate on the optimal linear estimate defined by Eqs. (11.1.5) and (11.1.6). For each time instant n at which an estimate $\hat{x}_n$ is sought, the optimal weights $h(n,i)$, $n_a \leq i \leq n_b$ must be determined that minimize the error criterion (11.1.6). In general, a new set of optimal weights must be computed for each time instant n. In the special case when the processes x_n and y_n are stationary and the observations are available for a long time, that is, $n_a = -\infty$, the weights become time-invariant in the sense that $h(n,i) = h(n-i)$, and the linear processor becomes an ordinary *time-invariant linear filter*. We will discuss the solution for $h(n,i)$ both for the time-invariant and the more general cases.

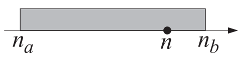
The problem of determining the optimal weights $h(n,i)$ according to the mean-square error minimization criterion (11.1.6) is in general referred to as the *Wiener filtering problem* [849–866]. An interesting historical account of the development of this problem and its ramifications is given in the review article by Kailath [866]. Wiener filtering problems are conventionally divided into three types:

1. The optimal *smoothing* problem,
2. The optimal *filtering* problem, and
3. The optimal *prediction* problem.

In all cases, the optimal estimate of x_n at a given time instant n is given by an expression of the form (11.1.5), as a linear combination of the available observations y_n in the interval $n_a \leq n \leq n_b$. The division into three types of problems depends on which of the available observations in that interval are taken into account in making up the linear combination (11.1.5).

In the smoothing problem, *all* the observations in the interval $[n_a, n_b]$ are taken into account. The shaded part in the following figure denotes the range of observations that are used in the summation of Eq. (11.1.5):

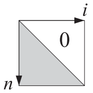
$$\hat{x}_n = \sum_{i=n_a}^{n_b} h(n,i)y_i$$



Since some of the observations are to the future of x_n, the linear operation is not causal. This does not present a problem if the sequence y_n is already available and stored in memory.

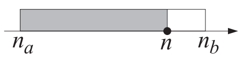
The optimal filtering problem, on the other hand, requires the linear operation (11.1.5) to be *causal*, that is, only those observations that are in the present and past of the current sample x_n must be used in making up the estimate $\hat{x}_n$. This requires that the matrix of optimal weights $h(n,i)$ be lower triangular, that is,

$$h(n,i) = 0, \quad \text{for} \quad n < i$$



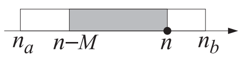
Thus, in reference to the figure below, only the shaded portion of the observation interval is used at the current time instant:

$$\hat{x}_n = \sum_{i=n_a}^{n} h(n,i)y_i$$



The estimate $\hat{x}_n$ depends on the present and all the past observations, from the fixed starting point n_a to the current time instant n. As n increases, more and more observations are taken into account in making up the estimate, and the actual computation of $\hat{x}_n$ becomes less and less efficient. It is desirable, then, to be able to recast the expression for $\hat{x}_n$ a time-recursive form. This is what is done in Kalman filtering. But, there is another way to make the Wiener filter computationally manageable. Instead of allowing a growing number of observations, only the current and the past M observations y_i, $i = n, n-1, \dots, n-M$ are taken into account. In this case, only $(M+1)$ filter weights are to be computed at each time instant n. This is depicted below:

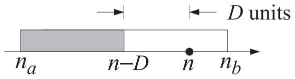
$$\hat{x}_n = \sum_{i=n-M}^{n} h(n,i)y_i = \sum_{m=0}^{M} h(n,n-m)y_{n-m}$$



This is referred to as the *finite impulse response* (FIR) Wiener filter. Because of its simple implementation, the FIR Wiener filter has enjoyed widespread popularity. Depending on the particular application, the practical implementation of the filter may vary. In Sec. 11.3 we present the theoretical formulation that applies to the stationary case; in Chap. 12 we reconsider it as a waveshaping and spiking filter and discuss a number of deconvolution applications. In Chap. 16, we consider its adaptive implementation using the Widrow-Hoff LMS algorithm and discuss a number of applications such as channel equalization and echo cancellation; we also discuss two alternative adaptive implementations—the so-called “gradient lattice,” and the “recursive least-squares.”

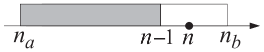
Finally, the linear prediction problem is a special case of the optimal filtering problem with the additional stipulation that observations only up to time instant $n-D$ must be used in obtaining the current estimate $\hat{x}_n$; this is equivalent to the problem of predicting D units of time into the future. The range of observations used in this case is shown below:

$$\hat{x}_n = \sum_{i=n_a}^{n-D} h(n,i)y_i$$



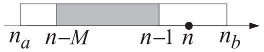
Of special interest to us will be the case of one-step prediction, corresponding to the choice $D = 1$. This is depicted below:

$$\hat{x}_n = \sum_{i=n_a}^{n-1} h(n,i)y_i$$



If we demand that the prediction be based only on the past M samples (from the current sample), we obtain the FIR version of the prediction problem, referred to as *linear prediction based on the past M samples*, which is depicted below:

$$\hat{x}_n = \sum_{i=n-M}^{n-1} h(n,i)y_i = \sum_{m=1}^{M} h(n,n-m)y_{n-m}$$



Next, we set up the orthogonality and normal equations for the optimal weights. We begin with the smoothing problem. The estimation error is in this case

$$e_n = x_n - \hat{x}_n = x_n - \sum_{i=n_a}^{n_b} h(n,i)y_i \tag{11.2.1}$$

Differentiating the mean-square estimation error (11.1.6) with respect to each weight $h(n,i)$, $n_a \le i \le n_b$, and setting the derivative to zero, we obtain the orthogonality equations that are enough to determine the weights:

$$\frac{\partial\mathcal{E}}{\partial h(n,i)} = 2E\left[e_n\frac{\partial e_n}{\partial h(n,i)}\right] = -2E[e_n y_i] = 0\,, \quad \text{for } n_a \le i \le n_b\,, \quad \text{or,}$$

$$R_{ey}(n,i) = E[e_n y_i] = 0 \quad \text{(orthogonality equations)} \tag{11.2.2}$$

for $n_a \le i \le n_b$. Thus, the estimation error e_n is orthogonal (uncorrelated) to each observation y_i used in *making up* the estimate $\hat{x}_n$. The orthogonality equations provide exactly as many equations as there are unknown weights.

Inserting Eq. (11.2.1) for e_n, the orthogonality equations may be written in an equivalent form, known as the *normal equations*

$$E\Big[\big(x_n - \sum_{k=n_a}^{n_b} h(n,k)y_k\big)y_i\Big] = 0\,, \quad \text{or,}$$

$$E[x_n y_i] = \sum_{k=n_a}^{n_b} h(n,k)E[y_k y_i] \quad \text{(normal equations)} \tag{11.2.3}$$

These determine the optimal weights at the current time instant n. In the vector notation of Sec. 11.1, we write Eq. (11.2.3) as

$$E[\mathbf{x}\mathbf{y}^T] = HE[\mathbf{y}\mathbf{y}^T]$$

where H is the matrix of weights $h(n,i)$. The optimal H and the estimate are then

$$\hat{\mathbf{x}} = H\mathbf{y} = E[\mathbf{x}\mathbf{y}^T]E[\mathbf{y}\mathbf{y}^T]^{-1}\mathbf{y}$$

This is identical to the correlation canceler of Sec. 1.4. The orthogonality equations (11.2.2) are precisely the correlation cancellation conditions. Extracting the nth row of this matrix equation, we find an explicit expression for the nth estimate $\hat{x}_n$

$$\hat{x}_n = E[x_n\mathbf{y}^T]E[\mathbf{y}\mathbf{y}^T]^{-1}\mathbf{y}$$

which is recognized as the projection of the random variable x_n onto the subspace spanned by the available observations; namely, $Y = \{y_{n_a}, y_{n_a+1}, \dots, y_{n_b}\}$. This is a general result: The minimum mean-square linear estimate $\hat{x}_n$ is the projection of x_n onto the subspace spanned by all the observations that are used to make up that estimate. This result is a direct consequence of the quadratic minimization criterion (11.1.6) and the orthogonal projection theorem discussed in Sec. 1.6.

Using the methods of Sec. 1.4, the minimized estimation error at time instant n is easily computed by

$$\begin{aligned}\mathcal{E}_n &= E[e_n e_n] = E[e_n x_n] = E\Big[\big(x_n - \sum_{i=n_a}^{n_b} h(n,i)y_i\big)x_n\Big] \\ &= E[x_n^2] - \sum_{i=n_a}^{n_b} h(n,i)E[y_i x_n] = E[x_n^2] - E[x_n\mathbf{y}^T]E[\mathbf{y}\mathbf{y}^T]^{-1}E[\mathbf{y}x_n]\end{aligned}$$

which corresponds to the diagonal entries of the covariance matrix of the estimation error $\mathbf{e}$:

$$R_{ee} = E[\mathbf{e}\mathbf{e}^T] = E[\mathbf{x}\mathbf{x}^T] - E[\mathbf{x}\mathbf{y}^T]E[\mathbf{y}\mathbf{y}^T]^{-1}E[\mathbf{y}\mathbf{x}^T]$$

The *optimum filtering* problem is somewhat more complicated because of the causality condition. In this case, the estimate at time n is given by

$$\hat{x}_n = \sum_{i=n_a}^{n} h(n,i)y_i \tag{11.2.4}$$

Inserting this into the minimization criterion (11.1.6) and differentiating with respect to $h(n,i)$ for $n_a \le i \le n$, we find again the orthogonality conditions

$$R_{ey}(n,i) = E[e_n y_i] = 0 \quad \text{for } n_a \le i \le n \tag{11.2.5}$$

where the most important difference from Eq. (11.2.2) is the restriction on the range of i, that is, e_n is decorrelated only from the present and past values of y_i. Again, the estimation error e_n is orthogonal to each observation y_i that is being used to make up the estimate. The orthogonality equations can be converted into the normal equations as follows:

$$E[e_n y_i]=E\big[\big(x_n-\sum_{k=n_a}^{n}h(n,k)y_k\big)y_i\big]=0\,,\quad \text{or,}$$

$$E[x_n y_i]=\sum_{k=n_a}^{n}h(n,k)E[y_k y_i]\quad \text{for}\;\; n_a\le i\le n\,,\quad \text{or,}\tag{11.2.6}$$

$$R_{xy}(n,i)=\sum_{k=n_a}^{n}h(n,k)R_{yy}(k,i)\quad \text{for}\;\; n_a\le i\le n\tag{11.2.7}$$

Such equations are generally known as *Wiener-Hopf* equations. Introducing the vector of observations *up to* the current time n, namely,

$$\mathbf{y}_n=[y_{n_a},y_{n_a+1},\dots,y_n]^T$$

we may write Eq. (11.2.6) in vector form as

$$E[x_n\mathbf{y}_n^T]=[h(n,n_a),h(n,n_a+1),\dots,h(n,n)]E[\mathbf{y}_n\mathbf{y}_n^T]$$

which can be solved for the vector of weights

$$[h(n,n_a),h(n,n_a+1),\dots,h(n,n)]=E[x_n\mathbf{y}_n^T]E[\mathbf{y}_n\mathbf{y}_n^T]^{-1}$$

and for the estimate $\hat{x}_n$:

$$\hat{x}_n=E[x_n\mathbf{y}_n^T]E[\mathbf{y}_n\mathbf{y}_n^T]^{-1}\mathbf{y}_n\tag{11.2.8}$$

Again, $\hat{x}_n$ is recognized as the *projection* of x_n onto the space spanned by the observations that are used in making up the estimate; namely, $Y_n=\{y_{n_a},y_{n_a+1},\dots,y_n\}$. This solution of Eqs. (11.2.5) and (11.2.7) will be discussed in more detail in Sec. 11.8, using covariance factorization methods.

11.3 Stationary Wiener Filter

In this section, we make two assumptions that simplify the structure of Eqs. (11.2.6) and (11.2.7). The first is to assume *stationarity* for all signals so that the cross-correlation and autocorrelation appearing in Eq. (11.2.7) become functions of the *differences* of their arguments. The second assumption is to take the initial time n_a to be the *infinite past*, $n_a=-\infty$, that is, the observation interval is $Y_n=\{y_i,\,-\infty<i\le n\}$.

The assumption of stationarity can be used as follows: Suppose we have the solution of $h(n,i)$ of Eq. (11.2.7) for the best weights to estimate x_n, and wish to determine the best weights $h(n+d,i)$, $n_a\le i\le n+d$ for estimating the sample x_{n+d} at the future time $n+d$. Then, the new weights will satisfy the same equations as (11.2.7) with the changes

$$R_{xy}(n+d,i)=\sum_{k=n_a}^{n+d}h(n+d,k)R_{yy}(k,i),\quad \text{for}\;\; n_a\le i\le n+d\tag{11.3.1}$$

Making a change of variables $i\to i+d$ and $k\to k+d$, we rewrite Eq. (11.3.1) as

$$R_{xy}(n+d,i+d)=\sum_{k=n_a-d}^{n}h(n+d,k+d)R_{yy}(k+d,i+d),\quad \text{for}\;\; n_a-d\le i\le n\tag{11.3.2}$$

Now, if we assume stationarity, Eqs. (11.2.7) and (11.3.2) become

$$\begin{aligned}R_{xy}(n-i)&=\sum_{k=n_a}^{n}h(n,k)R_{yy}(k-i)\,,\quad \text{for}\;\; n_a\le i\le n\\ R_{xy}(n-i)&=\sum_{k=n_a-d}^{n}h(n+d,k+d)R_{yy}(k-i)\,,\quad \text{for}\;\; n_a-d\le i\le n\end{aligned}\tag{11.3.3}$$

If it were not for the differences in the ranges of i and k, these two equations would be the same. But this is exactly what happens when we make the second assumption that $n_a=-\infty$. Therefore, by uniqueness of the solution, we find in this case

$$h(n+d,k+d)=h(n,k)$$

and since d is arbitrary, it follows that $h(n,k)$ must be a function of the difference of its arguments, that is,

$$h(n,k)=h(n-k)\tag{11.3.4}$$

Thus, the optimal linear processor becomes a *shift-invariant causal linear filter* and the estimate is given by

$$\hat{x}_n=\sum_{i=-\infty}^{n}h(n-i)y_i=\sum_{i=0}^{\infty}h(i)y_{n-i}\tag{11.3.5}$$

and Eq. (11.3.3) becomes in this case

$$R_{xy}(n-i)=\sum_{k=-\infty}^{n}h(n,k)R_{yy}(k-i)\,,\quad \text{for}\;\; -\infty<i\le n$$

With the change of variables $n-i\to n$ and $n-k\to k$, we find

$$R_{xy}(n)=\sum_{k=0}^{\infty}R_{yy}(n-k)h(k)\,,\quad \text{for}\;\; n\ge 0\tag{11.3.6}$$

and written in matrix form

$$\begin{bmatrix}R_{yy}(0)&R_{yy}(1)&R_{yy}(2)&R_{yy}(3)&\cdots\\ R_{yy}(1)&R_{yy}(0)&R_{yy}(1)&R_{yy}(2)&\cdots\\ R_{yy}(2)&R_{yy}(1)&R_{yy}(0)&R_{yy}(1)&\cdots\\ R_{yy}(3)&R_{yy}(2)&R_{yy}(1)&R_{yy}(0)&\cdots\\ \vdots&\vdots&\vdots&\vdots&\end{bmatrix}\begin{bmatrix}h(0)\\ h(1)\\ h(2)\\ h(3)\\ \vdots\end{bmatrix}=\begin{bmatrix}R_{xy}(0)\\ R_{xy}(1)\\ R_{xy}(2)\\ R_{xy}(3)\\ \vdots\end{bmatrix}\tag{11.3.7}$$

These are the *discrete-time Wiener-Hopf equations*. Were it not for the restriction $n\ge 0$ (which reflects the requirement of causality), they could be solved easily by z-transform methods. As written above, they require methods of *spectral factorization* for their solution.

Before we discuss such methods, we mention in passing the continuous-time version of the Wiener-Hopf equation:

$$R_{xy}(t)=\int_0^{\infty}R_{yy}(t-t')h(t')\,dt'\,,\quad t\geq 0$$

We also consider the FIR Wiener filtering problem in the stationary case. The observation interval in this case is $Y_n=\{y_i,\; n-M\leq i\leq n\}$. Using the same arguments as above we have $h(n,i)=h(n-i)$, and the estimate $\hat{x}_n$ is obtained by an ordinary FIR linear filter

$$\hat{x}_n=\sum_{i=n-M}^{n}h(n-i)y_i=h(0)y_n+h(1)y_{n-1}+\cdots+h(M)y_{n-M} \tag{11.3.8}$$

where the $(M+1)$ filter weights $h(0),h(1),\dots,h(M)$ are obtained by the $(M+1)\times(M+1)$ matrix version of the Wiener-Hopf normal equations:

$$\begin{bmatrix} R_{yy}(0) & R_{yy}(1) & R_{yy}(2) & \cdots & R_{yy}(M) \\ R_{yy}(1) & R_{yy}(0) & R_{yy}(1) & \cdots & R_{yy}(M-1) \\ R_{yy}(2) & R_{yy}(1) & R_{yy}(0) & \cdots & R_{yy}(M-2) \\ \vdots & \vdots & \vdots & & \vdots \\ R_{yy}(M) & R_{yy}(M-1) & R_{yy}(M-2) & \cdots & R_{yy}(0) \end{bmatrix} \begin{bmatrix} h(0) \\ h(1) \\ h(2) \\ \vdots \\ h(M) \end{bmatrix} = \begin{bmatrix} R_{xy}(0) \\ R_{xy}(1) \\ R_{xy}(2) \\ \vdots \\ R_{xy}(M) \end{bmatrix} \tag{11.3.9}$$

Exploiting the Toeplitz property of the matrix R_{yy}, the above matrix equation can be solved efficiently using Levinson's algorithm. This will be discussed in Chap. 12. In Chap. 16, we will consider adaptive implementations of the FIR Wiener filter which produce the optimal filter weights adaptively without requiring prior knowledge of the autocorrelation and cross-correlation matrices R_{yy} and R_{xy} and without requiring any matrix inversion.

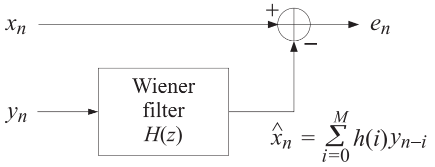


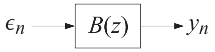
Fig. 11.3.1 Time-Invariant Wiener Filter.

We summarize our results on the stationary Wiener filter in Fig. 11.3.1. The optimal filter weights $h(n)$, $n=0,1,2,\dots$ are computed from Eq. (11.3.7) or Eq. (11.3.9). The action of the filter is precisely that of the correlation canceler: The filter processes the observation signal y_n *causally* to produce the best possible estimate $\hat{x}_n$ of x_n, and then it proceeds to cancel it from the output e_n. As a result, the output e_n is no longer correlated with any of the present and past values of y_n, that is, $E[e_ny_{n-i}]=0$, for $i=0,1,2,\dots$. As we remarked in Sec. 1.4, it is better to think of $\hat{x}_n$ as the optimal estimate of *that part* of the primary signal x_n which happens to be correlated with the secondary signal y_n. This follows from the property that if $x_n=x_1(n)+x_2(n)$ with $R_{x_2y}=0$, then $R_{xy}=R_{x_1y}$. Therefore, the solution of Eq. (11.3.7) for the best weights to estimate x_n is also the solution for the best weights to estimate $x_1(n)$. The filter may also be thought of as the *optimal signal separator* of the two signal components $x_1(n)$ and $x_2(n)$.

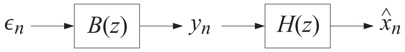
11.4 Construction of the Wiener Filter by Prewhitening

The normal equations (11.3.6) would have a trivial solution if the sequence y_n were a white-noise sequence with delta-function autocorrelation. Thus, the solution procedure is first to whiten the sequence y_n and then solve the normal equations. To this end, let y_n have a signal model, as guaranteed by the spectral factorization theorem

$$S_{yy}(z)=\sigma_\epsilon^2B(z)B(z^{-1}) \tag{11.4.1}$$

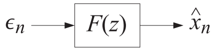


where ϵ_n is the driving white noise, and $B(z)$ a minimal-phase filter. The problem of estimating x_n in terms of the sequence y_n becomes equivalent to the problem of estimating x_n in terms of the white-noise sequence ϵ_n:



If we could determine the combined filter

$$F(z)=B(z)H(z)$$



we would then solve for the desired Wiener filter $H(z)$

$$H(z)=\frac{F(z)}{B(z)} \tag{11.4.2}$$

Since $B(z)$ is minimum-phase, the indicated inverse $1/B(z)$ is guaranteed to be *stable and causal.* Let f_n be the causal impulse response of $F(z)$. Then, it satisfies the normal equations of the type of Eq. (11.3.6):

$$R_{x\epsilon}(n)=\sum_{i=0}^{\infty}f_iR_{\epsilon\epsilon}(n-i)\,,\quad n\geq 0 \tag{11.4.3}$$

Since $R_{\epsilon\epsilon}(n-i)=\sigma_\epsilon^2\delta(n-i)$, Eq. (11.4.3) collapses to

$$R_{x\epsilon}(n)=\sigma_\epsilon^2f_n\,,\quad n\geq 0\,,\quad \text{or,}$$

$$f_n=\frac{R_{x\epsilon}(n)}{\sigma_\epsilon^2}\,,\quad \text{for }\; n\geq 0 \tag{11.4.4}$$

Next, we compute the corresponding z-transform $F(z)$

$$F(z)=\sum_{n=0}^{\infty}f_nz^{-n}=\frac{1}{\sigma_\epsilon^2}\sum_{n=0}^{\infty}R_{x\epsilon}(n)z^{-n}=\frac{1}{\sigma_\epsilon^2}\big[S_{x\epsilon}(z)\big]_+ \tag{11.4.5}$$

where $[S_{x\epsilon}(z)]_+$ denotes the *causal part* of the double-sided z-transform $S_{x\epsilon}(z)$. Generally, the causal part of a z-transform

$$G(z) = \sum_{n=-\infty}^{\infty} g_n z^{-n} = \sum_{n=-\infty}^{-1} g_n z^{-n} + \sum_{n=0}^{\infty} g_n z^{-n}$$

is defined as

$$[G(z)]_+ = \sum_{n=0}^{\infty} g_n z^{-n}$$

The causal instruction in Eq. (11.4.5) was necessary since the above solution for f_n was valid only for $n \geq 0$. Since y_n is the output of the filter $B(z)$ driven by ϵ_n, it follows that

$$S_{xy}(z) = S_{x\epsilon}(z)B(z^{-1}) \quad \text{or} \quad S_{x\epsilon}(z) = \frac{S_{xy}(z)}{B(z^{-1})}$$

Combining Eqs. (11.4.2) and (11.4.5), we finally find

$$H(z) = \frac{1}{\sigma_\epsilon^2 B(z)}\left[\frac{S_{xy}(z)}{B(z^{-1})}\right]_+ \quad \text{(Wiener filter)} \tag{11.4.6}$$

Thus, the construction of the optimal filter first requires the spectral factorization of $S_{yy}(z)$ to obtain $B(z)$, and then use of the above formula. This is the optimal *realizable* Wiener filter based on the *infinite past*. If the causal instruction is ignored, one obtains the optimal *unrealizable* Wiener filter

$$H_{\text{unreal}}(z) = \frac{S_{xy}(z)}{\sigma_\epsilon^2 B(z)B(z^{-1})} = \frac{S_{xy}(z)}{S_{yy}(z)} \tag{11.4.7}$$

The *minimum* value of the mean-square estimation error can be conveniently expressed by a contour integral, as follows

$$\begin{aligned}
\mathcal{E} &= E[e_n^2] = E\big[e_n(x_n - \hat{x}_n)\big] = E[e_n x_n] - E[e_n \hat{x}_n] = E[e_n x_n] = R_{ex}(0) \\
&= \oint_{\text{u.c.}} S_{ex}(z)\,\frac{dz}{2\pi jz} = \oint_{\text{u.c.}} \big[S_{xx}(z) - S_{\hat{x}x}(z)\big]\,\frac{dz}{2\pi jz}\,, \quad \text{or,}
\end{aligned}$$

$$\mathcal{E} = \oint_{\text{u.c.}} \big[S_{xx}(z) - H(z)S_{yx}(z)\big]\,\frac{dz}{2\pi jz} \tag{11.4.8}$$

11.5 Wiener Filter Example

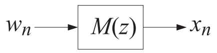
This example, in addition to illustrating the above ideas, will also serve as a short introduction to *Kalman filtering*. It is desired to estimate the signal x_n on the basis of noisy observations

$$y_n = x_n + v_n$$

where v_n is white noise of unit variance, $\sigma_v^2 = 1$, uncorrelated with x_n. The signal x_n is a first order Markov process, having a signal model

$$x_{n+1} = 0.6x_n + w_n$$

where w_n is white noise of variance $\sigma_w^2 = 0.82$. Enough information is given above to determine the required power spectral densities $S_{xy}(z)$ and $S_{yy}(z)$. First, we note that the signal generator transfer function for x_n is



$$M(z) = \frac{1}{z - 0.6}$$

so that

$$S_{xx}(z) = \sigma_w^2 M(z)M(z^{-1}) = \frac{0.82}{(z-0.6)(z^{-1}-0.6)} = \frac{0.82}{(1-0.6z^{-1})(1-0.6z)}$$

Then, we find

$$\begin{aligned}
S_{xy}(z) &= S_{x(x+v)}(z) = S_{xx}(z) + S_{xv}(z) = S_{xx}(z) = \frac{0.82}{(1-0.6z^{-1})(1-0.6z)} \\
S_{yy}(z) &= S_{(x+v)(x+v)}(z) = S_{xx}(z) + S_{xv}(z) + S_{vx}(z) + S_{vv}(z) = S_{xx}(z) + S_{vv}(z) \\
&= \frac{0.82}{(1-0.6z^{-1})(1-0.6z)} + 1 = \frac{0.82 + (1-0.6z^{-1})(1-0.6z)}{(1-0.6z^{-1})(1-0.6z)} \\
&= \frac{2(1-0.3z^{-1})(1-0.3z)}{(1-0.6z^{-1})(1-0.6z)} = 2\cdot\frac{1-0.3z^{-1}}{1-0.6z^{-1}}\cdot\frac{1-0.3z}{1-0.6z} \\
&= \sigma_\epsilon^2 B(z)B(z^{-1})
\end{aligned}$$

Then according to Eq. (11.4.6), we must compute the causal part of

$$G(z) = \frac{S_{xy}(z)}{B(z^{-1})} = \frac{\dfrac{0.82}{(1-0.6z^{-1})(1-0.6z)}}{\dfrac{1-0.3z}{1-0.6z}} = \frac{0.82}{(1-0.6z^{-1})(1-0.3z)}$$

This may be done by partial fraction expansion, but the fastest way is to use the contour inversion formula to compute g_k for $k \geq 0$, and then resum the z-transform:

$$\begin{aligned}
g_k &= \oint_{\text{u.c.}} G(z)z^k\,\frac{dz}{2\pi jz} = \oint_{\text{u.c.}} \frac{0.82z^k}{(1-0.3z)(z-0.6)}\,\frac{dz}{2\pi j} \\
&= (\text{residue at } z = 0.6) = \frac{0.82(0.6)^k}{1-(0.3)(0.6)} = (0.6)^k\,, \quad k \geq 0
\end{aligned}$$

Resumming, we find the causal part

$$[G(z)]_+ = \sum_{k=0}^{\infty} g_k z^{-k} = \frac{1}{1-0.6z^{-1}}$$

Finally, the optimum Wiener estimation filter is

$$H(z) = \frac{1}{\sigma_\epsilon^2 B(z)}\left[\frac{S_{xy}(z)}{B(z^{-1})}\right]_+ = \frac{[G(z)]_+}{\sigma_\epsilon^2 B(z)} = \frac{0.5}{1-0.3z^{-1}} \tag{11.5.1}$$

which can be realized as the difference equation

$$\hat{x}_n = 0.3\hat{x}_{n-1} + 0.5y_n \qquad y_n \longrightarrow \boxed{H(z)} \longrightarrow \hat{x}_n \tag{11.5.2}$$

The estimation error is also easily computed using the contour formula of Eq. (11.4.8):

$$\mathcal{E} = E[e_n^2] = \sigma_e^2 = \oint_{\text{u.c.}} \big[S_{xx}(z) - H(z)S_{yx}(z)\big]\frac{dz}{2\pi jz} = 0.5$$

To appreciate the improvement afforded by filtering, this error must be compared with the error in case no processing is made and y_n is itself taken to represent a noisy estimate of x_n. The estimation error in the latter case is $y_n - x_n = v_n$, so that $\sigma_v^2 = 1$. Thus, the gain afforded by processing is

$$\frac{\sigma_e^2}{\sigma_v^2} = 0.5 \quad \text{or} \quad 3 \text{ dB}$$

11.6 Wiener Filter as Kalman Filter

We would like to cast this example in a Kalman filter form. The difference equation Eq. (11.5.2) for the Wiener filter seems to have the "wrong" state transition matrix; namely, 0.3 instead of 0.6, which is the state matrix for the state model of x_n. However, it is not accidental that the Wiener filter difference equation may be rewritten in the alternative form

$$\hat{x}_n = 0.6\hat{x}_{n-1} + 0.5(y_n - 0.6\hat{x}_{n-1})$$

The quantity $\hat{x}_n$ is the best estimate of x_n, at time n, based on all the observations up to that time, that is, $Y_n = \{y_i,\ -\infty < i \leq n\}$. To simplify the subsequent notation, we denote it by $\hat{x}_{n/n}$. It is the projection of x_n on the space Y_n. Similarly, $\hat{x}_{n-1}$ denotes the best estimate of x_{n-1}, based on the observations up to time $n-1$, that is, $Y_{n-1} = \{y_i,\ -\infty < i \leq n-1\}$. The above filtering equation is written in this notation as

$$\hat{x}_{n/n} = 0.6\hat{x}_{n-1/n-1} + 0.5(y_n - 0.6\hat{x}_{n-1/n-1}) \tag{11.6.1}$$

It allows the computation of the *current best* estimate $\hat{x}_{n/n}$, in terms of the previous best estimate $\hat{x}_{n-1/n-1}$ and the new observation y_n that becomes available at the current time instant n.

The various terms of Eq. (11.6.1) have nice interpretations: Suppose that the best estimate $\hat{x}_{n-1/n-1}$ of the previous sample x_{n-1} is available. Even before the next observation y_n comes in, we may use this estimate to make a reasonable prediction as to what the next best estimate ought to be. Since we know the system dynamics of x_n, we may try to "boost" $\hat{x}_{n-1/n-1}$ to the next time instant n according to the system dynamics, that is, we take

$$\hat{x}_{n/n-1} = 0.6\hat{x}_{n-1/n-1} = \text{prediction of } x_n \text{ on the basis of } Y_{n-1} \tag{11.6.2}$$

Since $y_n = x_n + v_n$, we may use this prediction of x_n to make a prediction of the next measurement y_n, that is, we take

$$\hat{y}_{n/n-1} = \hat{x}_{n/n-1} = \text{prediction of } y_n \text{ on the basis of } Y_{n-1} \tag{11.6.3}$$

If this prediction were perfect, and if the next observation y_n were noise free, then this would be the value that we would observe. Since we actually observe y_n, the observation or innovations residual will be

$$\alpha_n = y_n - \hat{y}_{n/n-1} \tag{11.6.4}$$

This quantity represents that part of y_n that *cannot* be predicted on the basis of the previous observations Y_{n-1}. It represents the truly new information contained in the observation y_n. Actually, if we are making the best prediction possible, then the most we can expect of our prediction is to make the innovations residual a white-noise (uncorrelated) signal, that is, what remains after we make the best possible prediction should be unpredictable. According to the general discussion of the relationship between signal models and linear prediction given in Sec. 1.17, it follows that if $\hat{y}_{n/n-1}$ is the best predictor of y_n then α_n must be the whitening sequence that drives the signal model of y_n. We shall verify this fact shortly. This establishes an intimate connection between the *Wiener/Kalman filtering* problem and the *signal modeling* problem. If we overestimate the observation y_n the innovation residual will be negative; and if we underestimate it, the residual will be positive. In either case, we would like to correct our tentative estimate in the right direction. This may be accomplished by

$$\hat{x}_{n/n} = \hat{x}_{n/n-1} + G(y_n - \hat{y}_{n/n-1}) = 0.6\hat{x}_{n-1/n-1} + G(y_n - 0.6\hat{x}_{n-1/n-1}) \tag{11.6.5}$$

where the gain G, known as the *Kalman gain*, should be a positive quantity. The *prediction/correction* procedure defined by Eqs. (11.6.2) through (11.6.5) is known as the *Kalman filter*. It should be clear that any value for the gain G will provide an estimate, even if suboptimal, of x_n. Our solution for the Wiener filter has precisely the above structure with a gain $G = 0.5$. This value is optimal for the given example. It is a very instructive exercise to show this in two ways: First, with G arbitrary, the estimation filter of Eq. (11.6.5) has transfer function

$$H(z) = \frac{G}{1 - 0.6(1-G)z^{-1}} \qquad y_n \longrightarrow \boxed{H(z)} \longrightarrow \hat{x}_{n/n}$$

Insert this expression into the mean-square estimation error $\mathcal{E} = E[e_n^2]$, where $e_n = x_n - \hat{x}_{n/n}$, and minimize it with respect to the parameter G. This should give $G = 0.5$.

Alternatively, G should be such that to render the innovations residual (11.6.4) a white noise signal. In requiring this, it is useful to use the spectral factorization model for y_n, that is, the fact that y_n is the output of $B(z)$ when driven by the white noise signal ϵ_n. Working with z-transforms, we have:

$$\begin{aligned}
\alpha(z) &= Y(z) - 0.6z^{-1}\hat{X}(z) = Y(z) - 0.6z^{-1}H(z)Y(z) \\
&= \left[1 - 0.6z^{-1}\frac{G}{1 - 0.6(1-G)z^{-1}}\right]Y(z) = \left[\frac{1 - 0.6z^{-1}}{1 - 0.6(1-G)z^{-1}}\right]Y(z) \\
&= \left[\frac{1 - 0.6z^{-1}}{1 - 0.6(1-G)z^{-1}}\right]\left[\frac{1 - 0.3z^{-1}}{1 - 0.6z^{-1}}\right]\epsilon(z) = \left[\frac{1 - 0.3z^{-1}}{1 - 0.6(1-G)z^{-1}}\right]\epsilon(z)
\end{aligned}$$

Since ϵ_n is white, it follows that the transfer function relationship between α_n and ϵ_n must be trivial; otherwise, there will be sequential correlations present in α_n. Thus, we must have $0.6(1-G)=0.3$, or $G=0.5$; and in this case, $\alpha_n=\epsilon_n$. It is also possible to set $0.6(1-G)=1/0.3$, but this would correspond to an unstable filter.

We have obtained a most interesting result; namely, that when the Wiener filtering problem is recast into its Kalman filter form given by Eq. (11.6.1), then the innovations residual α_n, which is computable on line with the estimate $\hat{x}_{n/n}$, is identical to the whitening sequence ϵ_n of the signal model of y_n. In other words, the Kalman filter can be thought of as the *whitening filter* for the observation signal y_n.

To appreciate further the connection between Wiener and Kalman filters and between Kalman filters and the whitening filters of signal models, we consider a generalized version of the above example and cast it in standard Kalman filter notation.

It is desired to estimate x_n from y_n. The signal model for x_n is taken to be the first-order autoregressive model

$$x_{n+1}=ax_n+w_n \qquad \text{(state model)} \tag{11.6.6}$$

with $|a|<1$. The observation signal y_n is related to x_n by

$$y_n=cx_n+v_n \qquad \text{(measurement model)} \tag{11.6.7}$$

It is further assumed that the state and measurement noises, w_n and v_n, are zero-mean, mutually uncorrelated, white noises of variances Q and R, respectively, that is,

$$E[w_nw_i]=Q\delta_{ni}\,,\quad E[v_nv_i]=R\delta_{ni}\,,\quad E[w_nv_i]=0 \tag{11.6.8}$$

We also assume that v_n is uncorrelated with the initial value of x_n so that v_n and x_n will be uncorrelated for all n. The parameters a,c,Q,R are assumed to be known. Let $x_1(n)$ be the time-advanced version of x_n:

$$x_1(n)=x_{n+1}$$

and consider the two related Wiener filtering problems of estimating x_n and $x_1(n)$ on the basis of $Y_n=\{y_i,\ -\infty<i\le n\}$, depicted below

$$y_n \longrightarrow \boxed{H(z)} \longrightarrow \hat{x}_{n/n} \qquad\qquad y_n \longrightarrow \boxed{H_1(z)} \longrightarrow \hat{x}_1(n)=\hat{x}_{n+1/n}$$

The problem of estimating $x_1(n)=x_{n+1}$ is equivalent to the problem of one-step prediction into the future on the basis of the past and present. Therefore, we will denote this estimate by $\hat{x}_1(n)=\hat{x}_{n+1/n}$. The state equation (11.6.6) determines the spectral density of x_n:

$$S_{xx}(z)=\frac{1}{(z-a)(z^{-1}-a)}\,S_{ww}(z)=\frac{Q}{(1-az^{-1})(1-az)}$$

The observation equation (11.6.7) determines the cross-densities

$$\begin{aligned}S_{xy}(z)&=cS_{xx}(z)+S_{xv}(z)=cS_{xx}(z)\\ S_{x_1y}(z)&=zS_{xy}(z)=zcS_{xx}(z)\end{aligned}$$

where we used the filtering equation $X_1(z)=zX(z)$. The spectral density of y_n can be factored as follows:

$$\begin{aligned}S_{yy}(z)&=c^2S_{xx}(z)+S_{vv}(z)=\frac{c^2Q}{(1-az^{-1})(1-az)}+R\\ &=\frac{c^2Q+R(1-az^{-1})(1-az)}{(1-az^{-1})(1-az)}\equiv\sigma_\epsilon^2\left(\frac{1-fz^{-1}}{1-az^{-1}}\right)\left(\frac{1-fz}{1-az}\right)\end{aligned}$$

where f and σ_ϵ^2 satisfy the equations

$$f\sigma_\epsilon^2=aR \tag{11.6.9}$$

$$(1+f^2)\sigma_\epsilon^2=c^2Q+(1+a^2)R \tag{11.6.10}$$

and f has magnitude less than one. Thus, the corresponding signal model for y_n is

$$B(z)=\frac{1-fz^{-1}}{1-az^{-1}} \tag{11.6.11}$$

Next, we compute the causal parts as required by Eq. (11.4.6):

$$\begin{aligned}\left[\frac{S_{xy}(z)}{B(z^{-1})}\right]_+&=\left[\frac{cQ}{(1-az^{-1})(1-fz)}\right]_+=\frac{cQ}{1-fa}\,\frac{1}{1-az^{-1}}\\ \left[\frac{S_{x_1y}(z)}{B(z^{-1})}\right]_+&=\left[\frac{cQz}{(1-az^{-1})(1-fz)}\right]_+=\frac{cQa}{1-fa}\,\frac{1}{1-az^{-1}}\end{aligned}$$

Using Eq. (11.4.6), we determine the Wiener filters $H(z)$ and $H_1(z)$ as follows:

$$H(z)=\frac{1}{\sigma_\epsilon^2B(z)}\left[\frac{S_{xy}(z)}{B(z^{-1})}\right]_+=\frac{\dfrac{cQ/(1-fa)}{(1-az^{-1})}}{\sigma_\epsilon^2\left(\dfrac{1-fz^{-1}}{1-az^{-1}}\right)}=\frac{\left(\dfrac{cQ}{\sigma_\epsilon^2(1-fa)}\right)}{1-fz^{-1}}$$

or, defining the gain G by

$$G=\frac{cQ}{\sigma_\epsilon^2(1-fa)} \tag{11.6.12}$$

we finally find

$$H(z)=\frac{G}{1-fz^{-1}} \tag{11.6.13}$$

$$H_1(z)=aH(z)=\frac{K}{1-fz^{-1}} \tag{11.6.14}$$

where in Eq. (11.6.14) we defined a related gain, also called the Kalman gain, as follows:

$$K=aG=\frac{cQa}{\sigma_\epsilon^2(1-fa)} \tag{11.6.15}$$

Eq. (11.6.14) immediately implies that

$$\hat{x}_{n+1/n}=a\hat{x}_{n/n} \tag{11.6.16}$$

which is the precise justification of Eq. (11.6.2). The difference equations of the two filters are

$$\begin{aligned}\hat{x}_{n+1/n} &= f\hat{x}_{n/n-1} + Ky_n \\ \hat{x}_{n/n} &= f\hat{x}_{n-1/n-1} + Gy_n\end{aligned} \tag{11.6.17}$$

Using the results of Problem 1.50, we may express all the quantities f, σ_ϵ^2, K, and G in terms of a single positive quantity P which satisfies the *algebraic Riccati equation*:

$$Q = P - \frac{PRa^2}{R + c^2P} \tag{11.6.18}$$

Then, we find the interrelationships

$$K = aG = \frac{acP}{R + c^2P}\,, \quad \sigma_\epsilon^2 = R + c^2P\,, \quad f = a - cK = \frac{Ra}{R + c^2P} \tag{11.6.19}$$

It is left as an exercise to show that the minimized mean-square estimation errors are given in terms of P by

$$E[e_{n/n-1}^2] = P\,, \quad E[e_{n/n}^2] = \frac{RP}{R + c^2P}$$

where

$$e_{n/n-1} = x_n - \hat{x}_{n/n-1}\,, \quad e_{n/n} = x_n - \hat{x}_{n/n}$$

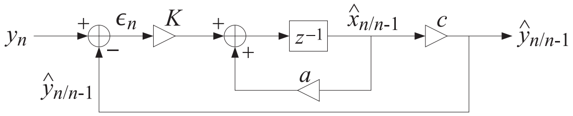
are the corresponding estimation errors for the optimally predicted and filtered estimates, respectively. Using Eq. (11.6.19)), we may rewrite the filtering equation (11.6.17) in the following forms:

$$\begin{aligned}\hat{x}_{n+1/n} &= (a - cK)\hat{x}_{n/n-1} + Ky_n\,, \quad \text{or,} \\ \hat{x}_{n+1/n} &= a\hat{x}_{n/n-1} + K(y_n - c\hat{x}_{n/n-1})\,, \quad \text{or,} \\ \hat{x}_{n+1/n} &= a\hat{x}_{n/n-1} + K(y_n - \hat{y}_{n/n-1})\end{aligned} \tag{11.6.20}$$

where we set

$$\hat{y}_{n/n-1} = c\hat{x}_{n/n-1} \tag{11.6.21}$$

A realization of the estimation filter based on (11.6.20) is shown below:



Replacing $K = aG$ and using Eq. (11.6.16) in (11.6.20), we also find

$$\hat{x}_{n/n} = \hat{x}_{n/n-1} + G(y_n - \hat{y}_{n/n-1}) \tag{11.6.22}$$

The quantity $\hat{y}_{n/n-1}$ defined in Eq. (11.6.21) is the best estimate of y_n based on its past Y_{n-1}. This can be seen in two ways: First, using the results of Problem 1.7 on the linearity of the estimates, we find

$$\hat{y}_{n/n-1} = \widehat{cx_n + v_n} = c\hat{x}_{n/n-1} + \hat{v}_{n/n-1} = c\hat{x}_{n/n-1}$$

where the term $\hat{v}_{n/n-1}$ was dropped. This term represents the estimate of v_n on the basis of the past ys; that is, Y_{n-1}. Since v_n is white and also uncorrelated with x_n, it follows that it will be uncorrelated with all past ys; therefore, $\hat{v}_{n/n-1} = 0$. The second way to show that $\hat{y}_{n/n-1}$ is the best prediction of y_n is to show that the innovations residual

$$\alpha_n = y_n - \hat{y}_{n/n-1} = y_n - c\hat{x}_{n/n-1} \tag{11.6.23}$$

is a white-noise sequence and coincides with the whitening sequence ϵ_n of y_n. Indeed, working in the z-domain and using Eq. (11.6.17) and the signal model of y_n we find

$$\begin{aligned}\alpha(z) &= Y(z) - cz^{-1}\hat{X}_1(z) = Y(z) - cz^{-1}H_1(z)Y(z) \\ &= \left[1 - cz^{-1}\frac{K}{1 - fz^{-1}}\right]Y(z) = \left[\frac{1 - (f + cK)z^{-1}}{1 - fz^{-1}}\right]Y(z) \\ &= \left[\frac{1 - az^{-1}}{1 - fz^{-1}}\right]Y(z) = \frac{1}{B(z)}Y(z) = \epsilon(z)\end{aligned}$$

which implies that

$$\alpha_n = \epsilon_n$$

Finally, we note that the recursive updating of the estimate of x_n given by Eq. (11.6.22) is identical to the result of Problem 1.11.

Our purpose in presenting this example was to tie together a number of ideas from Chapter 1 (correlation canceling, estimation, Gram-Schmidt orthogonalization, linear prediction, and signal modeling) to ideas from this chapter on Wiener filtering and its recursive reformulation as a Kalman filter.

We conclude this section by presenting a simulation of this example defined by the following choice of parameters:

$$a = 0.95\,, \quad c = 1\,, \quad Q = 1 - a^2\,, \quad R = 1$$

The above choice for Q normalizes the variance of x_n to unity. Solving the Riccati equation (11.6.18) and using Eq. (11.6.19), we find

$$P = 0.3122\,, \quad K = 0.2261\,, \quad G = 0.2380\,, \quad f = a - cK = 0.7239$$

Fig. 11.6.1 shows 100 samples of the observed signal y_n together with the desired signal x_n. The signal y_n processed through the Wiener filter $H(z)$ defined by the above parameters is shown in Fig. 11.6.2 together with x_n. The tracking properties of the filter are evident from the graph. It should be emphasized that this is the best one can do by means of ordinary causal linear filtering.

11.7 Construction of the Wiener Filter by the Gapped Function

Next, we would like to give an alternative construction of the optimal Wiener filter based on the concept of the gapped function. This approach is especially useful in linear prediction. The gapped function is defined as the cross-correlation between the estimation error e_n and the observation sequence y_n, as follows:

$$g(k) = R_{ey}(k) = E[e_ny_{n-k}]\,, \quad \text{for} \quad -\infty < k < \infty \tag{11.7.1}$$

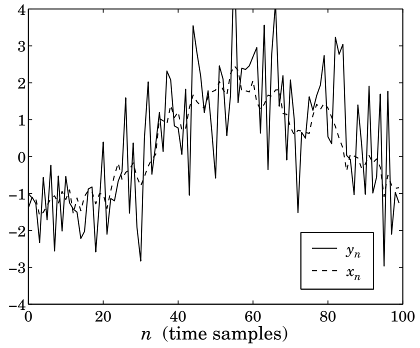


Fig. 11.6.1 Desired signal and its noisy observation.

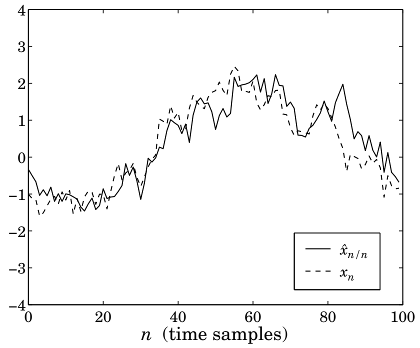
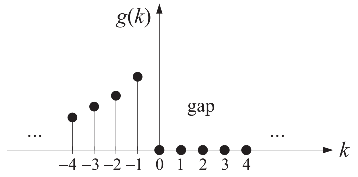


Fig. 11.6.2 Best estimate of desired signal.

This definition is motivated by the orthogonality equations which state that the prediction error e_n must be orthogonal to all of the available observations; namely, $Y_n = \{y_i\,,\ -\infty < i \le n\} = \{y_{n-k}\,,\ k \ge 0\}$. That is, for the optimal set of filter weights we must have

$$g(k) = R_{ey}(k) = E[e_n y_{n-k}] = 0\,, \quad \text{for } k \ge 0 \tag{11.7.2}$$



and $g(k)$ develops a right-hand side gap. On the other hand, $g(k)$ may be written in

the alternative form

$$g(k) = E[e_n y_{n-k}] = E\big[(x_n - \sum_{i=0}^{\infty} h_i y_{n-i})y_{n-k}\big] = R_{xy}(k) - \sum_{i=0}^{\infty} h_i R_{yy}(k-i)\,, \quad \text{or,}$$

$$g(k) = R_{ey}(k) = R_{xy}(k) - \sum_{i=0}^{\infty} h_i R_{yy}(k-i) \tag{11.7.3}$$

Taking z-transforms of both sides we find

$$G(z) = S_{ey}(z) = S_{xy}(z) - H(z)S_{yy}(z)$$

Because of the gap conditions, the left-hand side contains only positive powers of z, whereas the right-hand side contains both positive and negative powers of z. Thus, the non-positive powers of z must drop out of the right side. This condition precisely determines $H(z)$. Introducing the spectral factorization of $S_{yy}(z)$ and dividing both sides by $B(z^{-1})$ we find

$$\begin{aligned} G(z) &= S_{xy}(z) - H(z)S_{yy}(z) = S_{xy}(z) - H(z)\sigma_\epsilon^2 B(z)B(z^{-1}) \\ \frac{G(z)}{B(z^{-1})} &= \frac{S_{xy}(z)}{B(z^{-1})} - \sigma_\epsilon^2 H(z)B(z) \end{aligned}$$

The z-transform $B(z^{-1})$ is anticausal and, because of the gap conditions, so is the ratio $G(z)/B(z^{-1})$. Therefore, taking causal parts of both sides and noting that the product $H(z)B(z)$ is already causal, we find

$$0 = \left[\frac{S_{xy}(z)}{B(z^{-1})}\right]_+ - \sigma_\epsilon^2 H(z)B(z)$$

which may be solved for $H(z)$ to give Eq. (11.4.6).

11.8 Construction of the Wiener Filter by Covariance Factorization

In this section, we present a generalization of the gapped-function method to the more general non-stationary and/or finite-past Wiener filter. This is defined by the Wiener-Hopf equations (11.2.7), which are equivalent to the orthogonality equations (11.2.5). The latter are the non-stationary versions of the gapped function of the previous section. The best way to proceed is to cast Eqs. (11.2.5) in matrix form as follows: Without loss of generality we may take the starting point $n_a = 0$. The final point n_b is left arbitrary. Introduce the vectors

$$\mathbf{x} = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n_b} \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n_b} \end{bmatrix}$$

and the corresponding correlation matrices

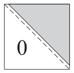
$$R_{xy} = E[\mathbf{x}\mathbf{y}^T]\,, \quad R_{yy} = E[\mathbf{y}\mathbf{y}^T]$$

The filtering equation (11.2.4) may be written in vector form as

$$\hat{\mathbf{x}} = H\mathbf{y} \tag{11.8.1}$$

where H is the matrix of optimal weights $\{h(n,i)\}$. The *causality* of the filtering operation (11.8.1), requires H to be *lower-triangular.* The minimization problem becomes equivalent to the problem of minimizing the mean-square estimation error subject to the constraint that H be lower-triangular. The minimization conditions are the normal equations (11.2.5) which, in this matrix notation, state that the matrix R_{ey} has no lower-triangular (causal) part; or, equivalently, that R_{ey} is *strictly* upper-triangular (i.e., even the main diagonal of R_{ey} is zero), therefore

$$R_{ey} = \text{strictly upper triangular} \tag{11.8.2}$$



Inserting Eq. (11.8.1) into R_{ey} we find

$$R_{ey} = E[\mathbf{e}\mathbf{y}^T] = E\big[(\mathbf{x} - H\mathbf{y})\mathbf{y}^T\big]\,, \quad \text{or,}$$

$$R_{ey} = R_{xy} - HR_{yy} \tag{11.8.3}$$

The minimization conditions (11.8.2) require H to be that lower-triangular matrix which renders the combination (11.8.3) upper-triangular. In other words, H should be such that the lower triangular part of the right-hand side must vanish. To solve Eqs. (11.8.2) and (11.8.3), we introduce the *LU Cholesky factorization* of the covariance matrix R_{yy} given by

$$R_{yy} = BR_{\epsilon\epsilon}B^T \tag{11.8.4}$$

where B is unit lower-triangular, and $R_{\epsilon\epsilon}$ is diagonal. This was discussed in Sec. 1.6. Inserting this into Eq. (11.8.3) we find

$$R_{ey} = R_{xy} - HR_{yy} = R_{xy} - HBR_{\epsilon\epsilon}B^T \tag{11.8.5}$$

Multiplying by the inverse transpose of B we obtain

$$R_{ey}B^{-T} = R_{xy}B^{-T} - HBR_{\epsilon\epsilon} \tag{11.8.6}$$

Now, the matrix B^{-T} is unit upper-triangular, but R_{ey} is strictly upper, therefore, the product $R_{xy}B^{-T}$ will be strictly upper. This can be verified easily for any two such matrices. Extracting the lower-triangular parts of both sides of Eq. (11.8.6) we find

$$0 = \big[R_{xy}B^{-T}\big]_+ - HBR_{\epsilon\epsilon}$$

where we used the fact that the left-hand side was strictly upper and that the term $HBR_{\epsilon\epsilon}$ was already lower-triangular. The notation $[\;]_+$ denotes the lower triangular part of a matrix including the diagonal. We find finally

$$H = \big[R_{xy}B^{-T}\big]_+ R_{\epsilon\epsilon}^{-1}B^{-1} \tag{11.8.7}$$

This is the most general solution of the Wiener filtering problem [18, 19]. It includes the results of the stationary case, as a special case. Indeed, if all the signals are stationary, then the matrices R_{xy}, B, and B^T become Toeplitz and have a z-transform associated with them as discussed in Problem 1.51. Using the results of that problem, it is easily seen that Eq. (11.8.7) is the time-domain equivalent of Eq. (11.4.6).

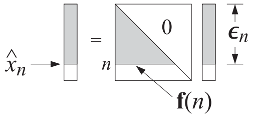
The prewhitening approach of Sec. 11.4 can also be understood in the present matrix framework. Making the change of variables

$$\mathbf{y} = B\boldsymbol{\epsilon}$$

we find that $R_{xy} = E[\mathbf{x}\mathbf{y}^T] = E[\mathbf{x}\boldsymbol{\epsilon}^T]B^T = R_{x\epsilon}B^T$, and therefore, $R_{xy}B^{-T} = R_{x\epsilon}$ and the filter H becomes $H = [R_{x\epsilon}]_+R_{\epsilon\epsilon}^{-1}B^{-1}$. The corresponding estimate is then

$$\hat{\mathbf{x}} = H\mathbf{y} = HB\boldsymbol{\epsilon} = F\boldsymbol{\epsilon}\,, \quad \text{where } \; F = HB = [R_{x\epsilon}]_+R_{\epsilon\epsilon}^{-1} \tag{11.8.8}$$

This is the matrix equivalent of Eq. (11.4.5). The matrix F is lower-triangular by construction. Therefore, to extract the nth component $\hat{x}_n$ of Eq. (11.8.8), it is enough to consider the $n{\times}n$ submatrices as shown below:



The nth row of F is $\mathbf{f}(n)^T = E[x_n\boldsymbol{\epsilon}_n^T]E[\boldsymbol{\epsilon}_n\boldsymbol{\epsilon}_n^T]^{-1}$. Therefore, the nth estimate becomes

$$\hat{x}_n = \mathbf{f}(n)^T\boldsymbol{\epsilon}_n = E[x_n\boldsymbol{\epsilon}_n^T]E[\boldsymbol{\epsilon}_n\boldsymbol{\epsilon}_n^T]^{-1}\boldsymbol{\epsilon}_n$$

which may also be written in the recursive form

$$\hat{x}_{n/n} = \sum_{i=0}^{n} E[x_n\epsilon_i]E[\epsilon_i\epsilon_i]^{-1}\epsilon_i = \sum_{i=0}^{n-1} E[x_n\epsilon_i]E[\epsilon_i\epsilon_i]^{-1}\epsilon_i + G_n\epsilon_n\,, \quad \text{or,}$$

$$\hat{x}_{n/n} = \hat{x}_{n/n-1} + G_n\epsilon_n \tag{11.8.9}$$

where we made an obvious change in notation, and $G_n = E[x_n\epsilon_n]E[\epsilon_n\epsilon_n]^{-1}$. This is identical to Eq. (11.6.22); in the stationary case, G_n is a constant, independent of n. We can also recast the nth estimate in "batch" form, expressed directly in terms of the observation vector $\mathbf{y}_n = [y_0, y_1, \dots, y_n]^T$. By considering the $n{\times}n$ subblock part of the Gram-Schmidt construction, we may write $\mathbf{y}_n = B_n\boldsymbol{\epsilon}_n$, where B_n is unit lower-triangular. Then, $\hat{x}_n$ can be expressed as

$$\hat{x}_n = E[x_n\boldsymbol{\epsilon}_n^T]E[\boldsymbol{\epsilon}_n\boldsymbol{\epsilon}_n^T]^{-1}\boldsymbol{\epsilon}_n = E[x_n\mathbf{y}_n^T]E[\mathbf{y}_n\mathbf{y}_n^T]^{-1}\mathbf{y}_n$$

which is identical to Eq. (11.2.8).

11.9 The Kalman Filter

The Kalman filter discussion of Sec. 11.6 and its equivalence to the Wiener filter was based on the asymptotic Kalman filter for which the observations were available from the infinite past to the present, namely, $\{y_i\,,\ -\infty < i \le n\}$. In Sec. 11.7, we solved the most general Wiener filtering problem based on the finite past for which the observation space was

$$Y_n = \{y_0, y_1, \dots, y_n\} \tag{11.9.1}$$

Here, we recast these results in a time-recursive form and obtain the time-varying Kalman filter for estimating x_n based on the finite observation subspace Y_n. We also discuss its asymptotic properties for large n and show that it converges to the steady-state Kalman filter of Sec. 11.6.

Our discussion is based on Eq. (11.8.9), which is essentially the starting point in Kalman's original derivation [852]. To make Eq. (11.8.9) truly recursive, we must have a means of recursively computing the required gain G_n from one time instant to the next. As in Sec. 11.8, we denote by $\hat{x}_{n/n}$ and $\hat{x}_{n/n-1}$ the optimal estimates of x_n based on the observation subspaces Y_n and Y_{n-1}, defined in Eq. (11.9.1), with the initial condition $\hat{x}_{0/-1} = 0$. Iterating the state and measurement models (11.6.6) and (11.6.7) starting at $n = 0$, we obtain the following two results, previously derived for the steady-state case

$$\hat{x}_{n+1/n} = a\hat{x}_{n/n}\,, \quad \hat{y}_{n/n-1} = c\hat{x}_{n/n-1} \tag{11.9.2}$$

The proof of both is based on the linearity property of estimates; for example,

$$\hat{x}_{n+1/n} = \widehat{ax_n + w_n} = a\hat{x}_{n/n} + \hat{w}_{n/n} = a\hat{x}_{n/n}$$

where $\hat{w}_{n/n}$ was set to zero because w_n does not depend on any of the observations Y_n. This is seen as follows. The iteration of the state equation (11.6.6) leads to the expression $x_n = a^n x_0 + a^{n-1}w_0 + a^{n-2}w_1 + \cdots + aw_{n-2} + w_{n-1}$. It follows from this and Eq. (11.6.7) that the observation subspace Y_n will depend only on

$$\{x_0, w_0, w_1, \dots, w_{n-1}, v_0, v_1, \dots, v_n\}$$

Making the additional assumption that x_0 is uncorrelated with w_n it follows that w_n will be uncorrelated with all random variables in the above set, and thus, with Y_n. The second part of Eq. (11.9.2) is shown by similar arguments. Next, we develop the recursions for the gain G_n. Using Eq. (11.8.9), the estimation and prediction errors may be related as follows

$$e_{n/n} = x_n - \hat{x}_{n/n} = x_n - \hat{x}_{n/n-1} - G_n\epsilon_n = e_{n/n-1} - G_n\epsilon_n$$

Taking the correlation of both sides with x_n we find

$$E[e_{n/n}x_n] = E[e_{n/n-1}x_n] - G_nE[\epsilon_n x_n] \tag{11.9.3}$$

Using the orthogonality properties $E[e_{n/n}\hat{x}_{n/n}] = 0$ and $E[e_{n/n-1}\hat{x}_{n/n-1}] = 0$, which follow from the optimality of the two estimates $\hat{x}_{n/n}$ and $\hat{x}_{n/n-1}$, we can write the mean-square estimation and prediction errors as

$$P_{n/n} = E[e_{n/n}^2] = E[e_{n/n}x_n]\,, \quad P_{n/n-1} = E[e_{n/n-1}^2] = E[e_{n/n-1}x_n] \tag{11.9.4}$$

We find also

$$\epsilon_n = y_n - \hat{y}_{n/n-1} = (cx_n + v_n) - c\hat{x}_{n/n-1} = ce_{n/n-1} + v_n$$

Using the fact that $e_{n/n-1}$ depends only on x_n and Y_{n-1}, it follows that the two terms in the right-hand side are uncorrelated with each other. Thus,

$$E[\epsilon_n^2] = c^2E[e_{n/n-1}^2] + E[v_n^2] = c^2P_{n/n-1} + R \tag{11.9.5}$$

also

$$E[\epsilon_n x_n] = cE[e_{n/n-1}x_n] + E[v_n x_n] = cP_{n/n-1} \tag{11.9.6}$$

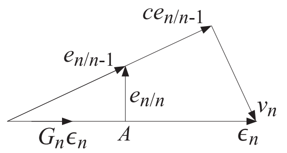
Therefore, the gain G_n is computable by

$$G_n = \frac{E[\epsilon_n x_n]}{E[\epsilon_n^2]} = \frac{cP_{n/n-1}}{R + c^2P_{n/n-1}} \tag{11.9.7}$$

Using Eqs. (11.9.4), (11.9.6), and (11.9.7) into Eq. (11.9.3), we obtain

$$P_{n/n} = P_{n/n-1} - G_ncP_{n/n-1} = P_{n/n-1} - \frac{c^2P_{n/n-1}}{R + c^2P_{n/n-1}} = \frac{RP_{n/n-1}}{R + c^2P_{n/n-1}} \tag{11.9.8}$$

The subtracted term in (11.9.8) represents the *improvement* in estimating x_n using $\hat{x}_{n/n}$ over using $\hat{x}_{n/n-1}$. Equations (11.9.3), (11.9.7), and (11.9.8) admit a nice geometrical interpretation [867]. The two right-hand side terms in $\epsilon_n = ce_{n/n-1} + v_n$ are orthogonal and can be represented by the orthogonal triangle



where the prediction error $e_{n/n-1}$ has been scaled up by the factor c. Thus, Eq. (11.9.5) is the statement of the Pythagorean theorem for this triangle. Next, write the equation $e_{n/n} = e_{n/n-1} - G_n\epsilon_n$ as

$$e_{n/n-1} = e_{n/n} + G_n\epsilon_n$$

Because $e_{n/n}$ is orthogonal to all the observations in Y_n and ϵ_n is a linear combination of the same observations, it follows that the two terms in the right-hand side will be orthogonal. Thus, $e_{n/n-1}$ may be resolved in two orthogonal parts, one being in the direction of ϵ_n. This is represented by the smaller orthogonal triangle in the previous diagram. Clearly, the length of the side $e_{n/n}$ is minimized at right angles at point A. It follows from the similarity of the two orthogonal triangles that

$$\frac{G_n\sqrt{E[\epsilon_n^2]}}{\sqrt{E[e_{n/n-1}^2]}} = \frac{c\sqrt{E[e_{n/n-1}^2]}}{\sqrt{E[\epsilon_n^2]}}$$

which is equivalent to Eq. (11.9.7). Finally, the Pythagorean theorem applied to the smaller triangle implies $E[e_{n/n-1}^2] = E[e_{n/n}^2] + G_n^2E[\epsilon_n^2]$, which is equivalent to Eq. (11.9.8).

To obtain a truly recursive scheme, we need next to find a relationship between $P_{n/n}$ and the next prediction error $P_{n+1/n}$. It is found as follows. From the state model (11.6.6) and (11.9.2), we have

$$e_{n+1/n} = x_{n+1} - \hat{x}_{n+1/n} = (ax_n + w_n) - a\hat{x}_{n/n} = ae_{n/n} + w_n$$

Because $e_{n/n}$ depends only on x_n and Y_n, it follows that the two terms in the right-hand side will be uncorrelated. Therefore, $E[e_{n+1/n}^2] = a^2E[e_{n/n}^2] + E[w_n^2]$, or,

$$P_{n+1/n} = a^2P_{n/n} + Q \tag{11.9.9}$$

The first term corresponds to the propagation of the estimate $\hat{x}_{n/n}$ forward in time according to the system dynamics; the second term represents the worsening of the estimate due to the presence of the dynamical noise w_n. The Kalman filter algorithm is now complete. It is summarized below:

0. Initialize by $\hat{x}_{0/-1} = 0$ and $P_{0/-1} = E[x_0^2]$.
1. At time n, $\hat{x}_{n/n-1}$, $P_{n/n-1}$, and the new measurement y_n are available.
2. Compute $\hat{y}_{n/n-1} = c\hat{x}_{n/n-1}$, $\epsilon_n = y_n - \hat{y}_{n/n-1}$, and the gain G_n using Eq. (11.9.7).
3. Correct the predicted estimate $\hat{x}_{n/n} = \hat{x}_{n/n-1} + G_n\epsilon_n$ and compute its mean-square error $P_{n/n}$, using Eq. (11.9.8).
4. Predict the next estimate $\hat{x}_{n+1/n} = a\hat{x}_{n/n}$, and compute the mean-square prediction error $P_{n+1/n}$, using Eq. (11.9.9).
5. Go to the next time instant, $n \to n + 1$.

The optimal predictor $\hat{x}_{n/n-1}$ satisfies the Kalman filtering equation

$$\hat{x}_{n+1/n} = a\hat{x}_{n/n} = a(\hat{x}_{n/n-1} + G_n\epsilon_n) = a\hat{x}_{n/n-1} + aG_n(y_n - c\hat{x}_{n/n-1}), \quad \text{or,}$$

$$\hat{x}_{n+1/n} = f_n\hat{x}_{n/n-1} + K_ny_n \tag{11.9.10}$$

where we defined

$$K_n = aG_n, \quad f_n = a - cK_n \tag{11.9.11}$$

These are the time-varying analogs of Eqs. (11.6.17) and (11.6.19). Equations (11.9.8) and (11.9.9) may be combined into one updating equation for $P_{n/n-1}$, known as the discrete Riccati *difference* equation

$$P_{n+1/n} = \frac{a^2RP_{n/n-1}}{R + c^2P_{n/n-1}} + Q \tag{11.9.12}$$

It is the time-varying version of Eq. (11.6.18). We note that in deriving all of the above results, we did not need to assume that the model parameters $\{a, c, Q, R\}$ were constants, independent of time. They can just as well be replaced by time-varying model parameters:

$$\{a_n, c_n, Q_n, R_n\}$$

The asymptotic properties of the Kalman filter depend, of course, on the particular time variations in the model parameters. In the time-invariant case, with $\{a, c, Q, R\}$ constant, we expect the solution of the Riccati equation (11.9.12) to converge, for large n, to some steady-state value $P_{n/n-1} \to P$. In this limit, the Riccati difference equation (11.9.12) tends to the steady-state algebraic Riccati equation (11.6.18), which determines the limiting value P. The Kalman filter parameters will converge to the limiting values $f_n \to f$, $K_n \to K$, and $G_n \to G$ given by Eq. (11.6.19).

It is possible to solve Eq. (11.9.12) in closed form and explicitly demonstrate these convergence properties. Using the techniques of [871,872], we obtain

$$P_{n/n-1} = P + \frac{f^{2n}E_0}{1 + S_nE_0}, \quad \text{for } n = 0, 1, 2, \dots, \tag{11.9.13}$$

where $E_0 = P_{0/-1} - P$ and

$$S_n = B\frac{1 - f^{2n}}{1 - f^2}, \quad B = \frac{c^2}{R + c^2P}$$

We have already mentioned (see Problem 1.50) that the stability of the signal model and the positivity of the asymptotic solution P imply the minimum phase condition $|f| < 1$. Thus, the second term of Eq. (11.9.13) converges to zero exponentially with a time constant determined by f.

Example 11.9.1: Determine the closed form solutions of the time-varying Kalman filter for the state and measurement models:

$$x_{n+1} = x_n + w_n, \quad y_n = x_n + v_n$$

with $Q = 0.5$ and $R = 1$. Thus, $a = 1$ and $c = 1$. The Riccati equations are

$$P_{n+1/n} = \frac{P_{n/n-1}}{1 + P_{n/n-1}} + 0.5, \quad P = \frac{P}{1 + P} + 0.5$$

The solution of the algebraic Riccati equation is $P = 1$. This implies that $f = aR/(R + c^2P) = 0.5$. To illustrate the solution (11.9.13), we take the initial condition to be zero $P_{0/-1} = 0$. We find $B = c^2/(R + c^2P) = 0.5$ and

$$S_n = \frac{2}{3}\left[1 - (0.5)^{2n}\right]$$

Thus,

$$P_{n/n-1} = 1 - \frac{(0.5)^{2n}}{1 - \frac{2}{3}\left[1 - (0.5)^{2n}\right]} = \frac{1 - (0.5)^{2n}}{1 + 2(0.5)^{2n}}$$

The first few values calculated from this formula are

$$P_{1/0} = \frac{1}{2}, \quad P_{2/1} = \frac{5}{6}, \quad P_{3/2} = \frac{21}{22}, \dots$$

and quickly converge to $P = 1$. They may also be obtained by iterating Eq. (11.9.12). □

11.10 Problems

11.1 Let $\mathbf{x} = [x_{n_a}, \dots, x_{n_b}]^T$ and $\mathbf{y} = [y_{n_a}, \dots, y_{n_b}]^T$ be the desired and available signal vectors. The relationship between $\mathbf{x}$ and $\mathbf{y}$ is assumed to be linear of the form

$$\mathbf{y} = C\mathbf{x} + \mathbf{v}$$

where C represents a linear degradation and $\mathbf{v}$ is a vector of zero-mean independent gaussian samples with a common variance σ_v^2. Show that the maximum likelihood (ME) estimation criterion is in this case equivalent to the following least-squares criterion, based on the quadratic vector norm:

$$\mathcal{E} = \|\mathbf{y} - C\mathbf{x}\|^2 = \text{minimum with respect to } \mathbf{x}$$

Show that the resulting estimate is given by

$$\hat{\mathbf{x}} = (C^T C)^{-1} C^T \mathbf{y}$$

11.2 Let $\hat{\mathbf{x}} = H\mathbf{y}$ be the optimal linear smoothing estimate of $\mathbf{x}$ given by Eq. (11.1.5). It is obtained by minimizing the mean-square estimation error $\mathcal{E}_n = E[e_n^2]$ for each n in the interval $[n_a, n_b]$.

(a) Show that the solution for H also minimizes the error covariance matrix

$$R_{ee} = E[\mathbf{e}\mathbf{e}^T]$$

where $\mathbf{e}$ is the vector of estimation errors $\mathbf{e} = [e_{n_a}, \dots, e_{n_b}]^T$.

(b) Show that H also minimizes every quadratic index of the form, for any positive semi-definite matrix Q:

$$E[\mathbf{e}^T Q \mathbf{e}] = \min$$

(c) Explain how the minimization of each $E[e_n^2]$ can be understood in terms of part (b).

11.3 Consider the smoothing problem of estimating the signal vector $\mathbf{x}$ from the signal vector $\mathbf{y}$. Assume that $\mathbf{x}$ and $\mathbf{y}$ are linearly related by

$$\mathbf{y} = C\mathbf{x} + \mathbf{v}$$

and that $\mathbf{v}$ and $\mathbf{x}$ are uncorrelated from each other, and that the covariance matrices of $\mathbf{x}$ and $\mathbf{v}$, R_{xx} and R_{vv}, are known. Show that the smoothing estimate of $\mathbf{x}$ is in this case

$$\hat{\mathbf{x}} = R_{xx} C^T [C R_{xx} C^T + R_{vv}]^{-1} \mathbf{y}$$

11.4 A stationary random signal has autocorrelation function $R_{xx}(k) = \sigma_x^2 a^{|k|}$, for all k. The observation signal is $y_n = x_n + v_n$, where v_n is a zero-mean, white noise sequence of variance σ_v^2, uncorrelated from x_n.

(a) Determine the optimal FIR Wiener filter of order $M = 1$ for estimating x_n from y_n.

(b) Repeat for the optimal linear predictor of order $M = 2$ for predicting x_n on the basis of the past two samples y_{n-1} and y_{n-2}.

11.5 A stationary random signal $x(n)$ has autocorrelation function $R_{xx}(k) = \sigma_x^2 a^{|k|}$, for all k. Consider a time interval $[n_a, n_b]$. The random signal $x(n)$ is known only at the end-points of that interval; that is, the only available observations are

$$y(n_a) = x(n_a), \quad y(n_b) = x(n_b)$$

Determine the optimal estimate of $x(n)$ based on just these two samples in the form

$$\hat{x}(n) = h(n, n_a) y(n_a) + h(n, n_b) y(n_b)$$

for the following values of n: (a) $n_a \le n \le n_b$, (b) $n \le n_a$, (c) $n \ge n_b$.

11.6 A stationary random signal x_n is to be estimated on the basis of the noisy observations

$$y_n = x_n + v_n$$

It is given that

$$S_{xx}(z) = \frac{1}{(1 - 0.5z^{-1})(1 - 0.5z)}, \quad S_{vv}(z) = 5, \quad S_{xv}(z) = 0$$

(a) Determine the optimal realizable Wiener filter for estimating the signal x_n on the basis of the observations $Y_n = \{y_i,\ i \le n\}$. Write the difference equation of this filter. Compute the mean-square estimation error.

(b) Determine the optimal realizable Wiener filter for predicting one step into the future; that is, estimate x_{n+1} on the basis of Y_n.

(c) Cast the results of (a) and (b) in a predictor/corrector Kalman filter form, and show explicitly that the innovations residual of the observation signal y_n is identical to the corresponding whitening sequence ϵ_n driving the signal model of y_n.

11.7 Repeat the previous problem for the following choice of state and measurement models

$$x_{n+1} = x_n + w_n, \quad y_n = x_n + v_n$$

where w_n and v_n have variances $Q = 0.5$ and $R = 1$, respectively.

11.8 Consider the state and measurement equations

$$x_{n+1} = a x_n + w_n, \quad y_n = c x_n + v_n$$

as discussed in Sec. 11.6. For *any* value of the Kalman gain K, consider the Kalman predictor/corrector algorithm defined by the equation

$$\hat{x}_{n+1/n} = a\hat{x}_{n/n-1} + K(y_n - c\hat{x}_{n/n-1}) = f\hat{x}_{n/n-1} + K y_n \tag{P.1}$$

where $f = a - cK$. The stability requirement of this estimation filter requires further that K be such that $|f| < 1$.

(a) Let $e_{n/n-1} = x_n - \hat{x}_{n/n-1}$ be the corresponding estimation error. Assuming that all signals are stationary, and working with z-transforms, show that the power spectral density of $e_{n/n-1}$ is given by

$$S_{ee}(z) = \frac{Q + K^2 R}{(1 - fz^{-1})(1 - fz)}$$

(b) Integrating $S_{ee}(z)$ around the unit circle, show that the mean-square value of the estimation error is given by

$$\mathcal{E} = E[e_{n/n-1}^2] = \frac{Q + K^2 R}{1 - f^2} = \frac{Q + K^2 R}{1 - (a - cK)^2} \tag{P.2}$$

(c) To select the optimal value of the Kalman gain K, differentiate $\mathcal{E}$ with respect to K and set the derivative to zero. Show that the resulting equation for K can be expressed in the form

$$K = \frac{caP}{R + c^2 P}$$

where P stands for the minimized value of $\mathcal{E}$; that is, $P = \mathcal{E}_{\min}$.

(d) Inserting this expression for K back into the expression (P.2) for $\mathcal{E}$, show that the quantity P must satisfy the algebraic Riccati equation

$$Q = P - \frac{a^2RP}{R + c^2P}$$

Thus, the resulting estimator filter is identical to the optimal one-step prediction filter discussed in Sec. 11.6.

11.9 Show that Eq. (P.2) of Problem 11.8 can be derived without using z-transforms, by using only stationarity, as suggested below: Using the state and measurement model equations and Eq. (P. l), show that the estimation error $e_{n/n-1}$ satisfies the difference equation

$$e_{n+1/n} = fe_{n/n-1} + w_n - Kv_n$$

Then, invoking stationarity, derive Eq. (P.2). Using similar methods, show that the mean-square estimation error is given by

$$E[e_{n/n}^2] = \frac{RP}{R + c^2P}$$

where $e_{n/n} = x_n - \hat{x}_{n/n}$ is the estimation error of the optimal filter (11.6.13).

11.10 Consider the general example of Sec. 11.6. It was shown there that the innovations residual was the same as the whitening sequence ϵ_n driving the signal model of y_n

$$\epsilon_n = y_n - \hat{y}_{n/n-1} = y_n - c\hat{x}_{n/n-1}$$

Show that it can be written as

$$\epsilon_n = ce_{n/n-1} + v_n$$

where $e_{n/n-1} = x_n - \hat{x}_{n/n-1}$ is the prediction error. Then, show that

$$\sigma_\epsilon^2 = E[\epsilon_n^2] = R + c^2P$$

11.11 *Computer Experiment.* Consider the signal and measurement model defined by Eqs. (11.6.6) through (11.6.8), with the choices $a = 0.9$, $c = 1$, $Q = 1 - a^2$, and $R = 1$. Generate 1500 samples of the random noises w_n and v_n. Generate the corresponding signals x_n and y_n according to the state and measurement equations. Determine the optimal Wiener filter of the form (11.6.13) for estimating x_n on the basis of y_n. Filter the sequence y_n through the Wiener filter to generate the sequence $\hat{x}_{n/n}$.

(a) On the same graph, plot the desired signal x_n and the available noisy version y_n for n ranging over the last 100 values (i.e., n = 1400–1500.)

(b) On the same graph, plot the recovered signal $\hat{x}_{n/n}$ together with the original signal x_n for n ranging over the last 100 values.

(c) Repeat (a) and (b) using a different realization of w_n and v_n.

(d) Repeat (a), (b), and (c) for the choice $a = -0.9$.

11.12 Consider the optimal Wiener filtering problem in its matrix formulation of Sec. 11.8. Let $\mathbf{e} = \mathbf{x} - \hat{\mathbf{x}} = \mathbf{x} - H\mathbf{y}$ be the estimation error corresponding to a particular choice of the lower-triangular matrix H. Minimize the error covariance matrix $R_{ee} = E[\mathbf{e}\mathbf{e}^T]$ with respect to H subject to the *constraint* that H be lower-triangular. These constraints are $H_{ni} = 0$ for $n < i$. To do this, introduce a set of Lagrange multipliers Λ_{ni} for $n < i$, one for each constraint equation, and incorporate them into an effective performance index

$$J = E[\mathbf{e}\mathbf{e}^T] + \Lambda H^T + H\Lambda^T = \min$$

where the matrix Λ is strictly upper-triangular. Show that this formulation of the minimization problem yields exactly the same solution as Eq. (11.8.7).

11.13 *Exponential Moving Average as Wiener Filter.* The single EMA filter for estimating the local level of a signal that we discussed in Chap. 6 admits a nice Wiener-Kalman filtering interpretation. Consider the noisy random walk signal model,

$$\begin{aligned} x_{n+1} &= x_n + w_n \\ y_n &= x_n + v_n \end{aligned} \tag{11.10.1}$$

where w_n, v_n are mutually uncorrelated, zero-mean, white noise signals of variances $Q = \sigma_w^2$ and $R = \sigma_v^2$. Based on the material in Section 12.6, show that the optimum Wiener/Kalman filter for predicting x_n from y_n is equivalent to the exponential smoother, that is, show that it is given by,

$$\hat{x}_{n+1/n} = f\hat{x}_{n/n-1} + (1 - f)y_n \tag{11.10.2}$$

so that the forgetting-factor parameter λ of EMA is identified as the closed-loop parameter f of the Kalman filter, and show further that f is given in terms of Q, R as follows,

$$1 - f = \frac{\sqrt{Q^2 + 4QR} - Q}{2R}$$

Show also the $\hat{x}_{n+1/n} = \hat{x}_{n/n}$.

a. For the following values $\sigma_w = 0.1$ and $\sigma_v = 1$, generate $N = 300$ samples of x_n, y_n from Eq. (11.10.1) and run y_n through the equivalent Kalman filter of Eq. (11.10.2) to compute $\hat{x}_{n/n-1}$. On the same graph, plot all three signals $y_n, x_n, \hat{x}_{n/n-1}$ versus $0 \le n \le N - 1$. An example graph is shown at the end.

b. A possible way to determine λ or f from the data y_n is as follows. Assume a tentative value for λ, compute $\hat{x}_{n/n-1}$, then the error $e_{n/n-1} = x_n - \hat{x}_{n/n-1}$, and the mean-square error:

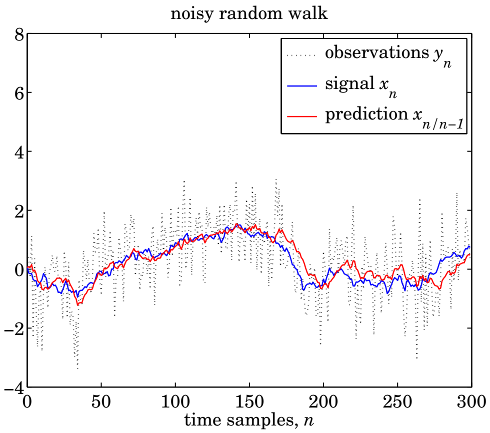
$$\text{MSE}(\lambda) = \sum_n e_{n/n-1}^2$$

Repeat the calculation of MSE(λ) over a range of λs, for example, $0.80 \le \lambda \le 0.95$, chosen such that the interval $[0.80, 0.95]$ contain the true λ. Then find that λ that minimizes MSE(λ), which should be close to the true value.

Because the estimated λ depends on the particular realization of the model (11.10.1), generate 20 different realizations of the pair x_n, y_n with the same Q, R, and for each realization carry out the estimate of λ as described above, and finally form the average of the 20 estimated λs. Discuss if this method generates an acceptable estimate of λ or f.

c. Repeat part (b), by replacing the MSE by the mean-absolute-error:

$$\text{MAE}(\lambda) = \sum_n |e_{n/n-1}|$$



12 Linear Prediction

12.1 Pure Prediction and Signal Modeling

In Sec. 1.17, we discussed the connection between linear prediction and signal modeling. Here, we rederive the same results by considering the linear prediction problem as a special case of the Wiener filtering problem, given by Eq. (11.4.6). Our aim is to cast the results in a form that will suggest a practical way to solve the prediction problem and hence also the modeling problem. Consider a stationary signal y_n having a signal model

$$S_{yy}(z)=\sigma_\epsilon^2 B(z)B(z^{-1}) \qquad \epsilon_n \longrightarrow \boxed{B(z)} \longrightarrow y_n \tag{12.1.1}$$

as guaranteed by the spectral factorization theorem. Let $R_{yy}(k)$ denote the autocorrelation of y_n:

$$R_{yy}(k)=E[y_{n+k}y_n]$$

The linear prediction problem is to predict the current value y_n on the basis of all the past values $Y_{n-1}=\{y_i\,,\ -\infty<i\le n-1\}$. If we define the delayed signal $y_1(n)=y_{n-1}$, then the linear prediction problem is equivalent to the optimal Wiener filtering problem of estimating y_n from the related signal $y_1(n)$. The optimal estimation filter $H(z)$ is given by Eq. (11.4.6), where we must identify x_n and y_n with y_n and $y_1(n)$ of the present notation. Using the filtering equation $Y_1(z)=z^{-1}Y(z)$, we find that y_n and $y_1(n)$ have the same spectral factor $B(z)$

$$S_{y_1y_1}(z)=(z^{-1})(z)S_{yy}(z)=S_{yy}(z)=\sigma_\epsilon^2 B(z)B(z^{-1})$$

and also that

$$S_{yy_1}(z)=S_{yy}(z)z=z\sigma_\epsilon^2 B(z)B(z^{-1})$$

Inserting these into Eq. (11.4.6), we find for the optimal filter $H(z)$

$$H(z)=\frac{1}{\sigma_\epsilon^2 B(z)}\left[\frac{S_{yy_1}(z)}{B(z^{-1})}\right]_+=\frac{1}{\sigma_\epsilon^2 B(z)}\left[\frac{z\sigma_\epsilon^2 B(z)B(z^{-1})}{B(z^{-1})}\right]_+ \,,\quad \text{or,}$$

$$H(z)=\frac{1}{B(z)}\big[zB(z)\big]_+ \tag{12.1.2}$$

The causal instruction can be removed as follows: Noting that $B(z)$ is a causal and stable filter, we may expand it in the power series

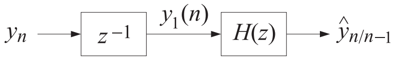
$$B(z)= 1 + b_1z^{-1} + b_2z^{-2} + b_3z^{-3} + \cdots$$

The causal part of $zB(z)$ is then

$$\begin{aligned}\bigl[zB(z)\bigr]_+ &= [z + b_1 + b_2z^{-1} + b_3z^{-2} + \cdots]_+ = b_1 + b_2z^{-1} + b_3z^{-2} + \cdots \\ &= z\bigl(b_1z^{-1} + b_2z^{-2} + b_3z^{-3} + \cdots\bigr) = z\bigl(B(z)-1\bigr)\end{aligned}$$

The prediction filter $H(z)$ then becomes

$$H(z)= \frac{1}{B(z)}\, z\bigl(B(z)-1\bigr) = z\left[1-\frac{1}{B(z)}\right] \tag{12.1.3}$$



The input to this filter is $y_1(n)$ and the output is the prediction $\hat{y}_{n/n-1}$.

Example 12.1.1: Suppose that y_n is generated by driving the all-pole filter

$$y_n = 0.9y_{n-1} - 0.2y_{n-2} + \epsilon_n$$

by zero-mean white noise ϵ_n. Find the best predictor $\hat{y}_{n/n-1}$. The signal model in this case is $B(z)= 1/(1-0.9z^{-1}+0.2z^{-2})$ and Eq. (12.1.3) gives

$$z^{-1}H(z)= 1-\frac{1}{B(z)} = 1-(1-0.9z^{-1}+0.2z^{-2})= 0.9z^{-1}-0.2z^{-2}$$

The I/O equation for the prediction filter is obtained by

$$\hat{Y}(z)= H(z)Y_1(z)= z^{-1}H(z)Y(z)= \bigl[0.9z^{-1}-0.2z^{-2}\bigr]Y(z)$$

and in the time domain

$$\hat{y}_{n/n-1} = 0.9y_{n-1} - 0.2y_{n-2}$$

Example 12.1.2: Suppose that

$$S_{yy}(z)= \frac{(1-0.25z^{-2})(1-0.25z^{2})}{(1-0.8z^{-1})(1-0.8z)}$$

Determine the best predictor $\hat{y}_{n/n-1}$. Here, the minimum phase factor is

$$B(z)= \frac{1-0.25z^{-2}}{1-0.8z^{-1}}$$

and therefore the prediction filter is

$$z^{-1}H(z)= 1-\frac{1}{B(z)} = 1-\frac{1-0.8z^{-1}}{1-0.25z^{-2}} = \frac{0.8z^{-1}-0.25z^{-2}}{1-.25z^{-2}}$$

The I/O equation of this filter is conveniently given recursively by the difference equation

$$\hat{y}_{n/n-1} = 0.25\hat{y}_{n-2/n-3} + 0.8y_{n-1} - 0.25y_{n-2}$$ □

The prediction error

$$e_{n/n-1} = y_n - \hat{y}_{n/n-1}$$

is identical to the whitening sequence ϵ_n driving the signal model (12.1.1) of y_n, indeed,

$$\begin{aligned}E(z) &= Y(z)-\hat{Y}(z)= Y(z)-H(z)Y_1(z)= Y(z)-H(z)z^{-1}Y(z) \\ &= \bigl[1-z^{-1}H(z)\bigr]Y(z)= \frac{1}{B(z)}Y(z)= \epsilon(z)\end{aligned}$$

Thus, in accordance with the results of Sec. 1.13 and Sec. 1.17

$$e_{n/n-1} = y_n - \hat{y}_{n/n-1} = \epsilon_n \tag{12.1.4}$$

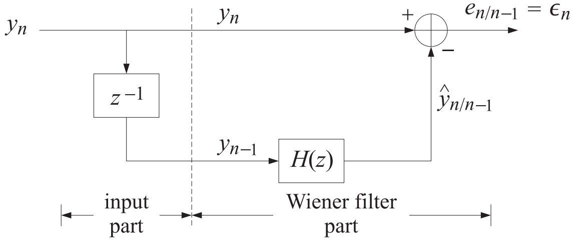


Fig. 12.1.1 Linear Predictor.

An overall realization of the linear predictor is shown in Fig. 12.1.1. The indicated dividing line separates the linear predictor into the Wiener filtering part and the input part which provides the proper input signals to the Wiener part. The transfer function from y_n to $e_{n/n-1}$ is the *whitening inverse filter*

$$A(z)= \frac{1}{B(z)} = 1-z^{-1}H(z)$$

which is *stable and causal* by the minimum-phase property of the spectral factorization (12.1.1). In the z-domain we have

$$E(z)= \epsilon(z)= A(z)Y(z)$$

and in the time domain

$$e_{n/n-1} = \epsilon_n = \sum_{m=0}^{\infty} a_m y_{n-m} = y_n + a_1y_{n-1} + a_2y_{n-2} + \cdots$$

The predicted estimate $\hat{y}_{n/n-1} = y_n - e_{n/n-1}$ is

$$\hat{y}_{n/n-1} = -\bigl[a_1y_{n-1} + a_2y_{n-2} + \cdots\bigr]$$

These results are identical to Eqs. (1.17.2) and (1.17.3). The relationship noted above between linear prediction and signal modeling can also be understood in terms of the gapped-function approach of Sec. 11.7. Rewriting Eq. (12.1.1) in terms of the prediction-error filter $A(z)$ we have

$$S_{yy}(z)=\frac{\sigma_\epsilon^2}{A(z)A(z^{-1})} \tag{12.1.5}$$

from which we obtain

$$A(z)S_{yy}(z)=\frac{\sigma_\epsilon^2}{A(z^{-1})} \tag{12.1.6}$$

Since we have the filtering equation $\epsilon(z)=A(z)Y(z)$, it follows that

$$S_{\epsilon y}(z)=A(z)S_{yy}(z)$$

and in the time domain

$$R_{ey}(k)=E[\epsilon_n y_{n-k}]=\sum_{i=0}^{\infty}a_i R_{yy}(k-i) \tag{12.1.7}$$

which is recognized as the gapped function (11.7.1). By construction, ϵ_n is the orthogonal complement of y_n with respect to the entire past subspace $Y_{n-1}=\{y_{n-k},\ k=1,2,\dots\}$, therefore, ϵ_n will be orthogonal to each y_{n-k} for $k=1,2,\dots$. These are precisely the gap conditions. Because the prediction is based on the entire past, the gapped function develops an infinite right-hand side gap. Thus, Eq. (12.1.7) implies

$$R_{ey}(k)=E[\epsilon_n y_{n-k}]=\sum_{i=0}^{\infty}a_i R_{yy}(k-i)=0\,,\quad \text{for all}\ \ k=1,2,\dots \tag{12.1.8}$$

The same result, of course, also follows from the z-domain equation (12.1.6). Both sides of the equation are stable, but since $A(z)$ is minimum-phase, $A(z^{-1})$ will be maximum phase, and therefore it will have a stable but anticausal inverse $1/A(z^{-1})$. Thus, the right-hand side of Eq. (12.1.6) has no strictly causal part. Equating to zero all the coefficients of positive powers of z^{-1} results in Eq. (12.1.8).

The value of the gapped function at $k=0$ is equal to σ_ϵ^2. Indeed, using the gap conditions (12.1.8) we find

$$\begin{aligned}\sigma_\epsilon^2 &= E[\epsilon_n^2]=E\big[\epsilon_n(y_n+a_1y_{n-1}+a_2y_{n-2}+\cdots)\big]\\ &= R_{\epsilon y}(0)+a_1R_{\epsilon y}(1)+a_2R_{\epsilon y}(2)+\cdots=R_{\epsilon y}(0)=E[\epsilon_n y_n]\end{aligned}$$

Using Eq. (12.1.7) with $k=0$ and the symmetry property $R_{yy}(i)=R_{yy}(-i)$, we find

$$\sigma_\epsilon^2=E[\epsilon_n^2]=E[\epsilon_n y_n]=R_{yy}(0)+a_1R_{yy}(1)+a_2R_{yy}(2)+\cdots \tag{12.1.9}$$

Equations (12.1.8) and (12.1.9) may be combined into one:

$$\sum_{i=0}^{\infty}a_iR_{yy}(k-i)=\sigma_\epsilon^2\delta(k)\,,\quad \text{for all}\ \ k\geq 0 \tag{12.1.10}$$

which can be cast in the matrix form:

$$\begin{bmatrix} R_{yy}(0) & R_{yy}(1) & R_{yy}(2) & R_{yy}(3) & \cdots\\ R_{yy}(1) & R_{yy}(0) & R_{yy}(1) & R_{yy}(2) & \cdots\\ R_{yy}(2) & R_{yy}(1) & R_{yy}(0) & R_{yy}(1) & \cdots\\ R_{yy}(3) & R_{yy}(2) & R_{yy}(1) & R_{yy}(0) & \cdots\\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}\begin{bmatrix}1\\ a_1\\ a_2\\ a_3\\ \vdots\end{bmatrix}=\begin{bmatrix}\sigma_\epsilon^2\\ 0\\ 0\\ 0\\ \vdots\end{bmatrix} \tag{12.1.11}$$

These equations are known as the *normal equations* of linear prediction [915–928]. They provide the solution to both the signal modeling and the linear prediction problems. They determine the model parameters $\{a_1,a_2,\dots;\sigma_\epsilon^2\}$ of the signal y_n directly in terms of the experimentally accessible quantities $R_{yy}(k)$. To render them computationally manageable, the infinite matrix equation (12.1.11) must be reduced to a finite one, and furthermore, the quantities $R_{yy}(k)$ must be estimated from actual data samples of y_n. We discuss these matters next.

12.2 Autoregressive Models

In general, the number of prediction coefficients $\{a_1,a_2,\dots\}$ is infinite since the predictor is based on the infinite past. However, there is an important exception to this; namely, when the process y_n is autoregressive. In this case, the signal model $B(z)$ is an all-pole filter of the type

$$B(z)=\frac{1}{A(z)}=\frac{1}{1+a_1z^{-1}+a_2z^{-2}+\cdots+a_pz^{-p}} \tag{12.2.1}$$

which implies that the prediction filter is a polynomial

$$A(z)=1+a_1z^{-1}+a_2z^{-2}+\cdots+a_pz^{-p} \tag{12.2.2}$$

The signal generator for y_n is the following difference equation, driven by the uncorrelated sequence ϵ_n:

$$y_n+a_1y_{n-1}+a_2y_{n-2}+\cdots+a_py_{n-p}=\epsilon_n \tag{12.2.3}$$

and the optimal prediction of y_n is simply given by:

$$\hat{y}_{n/n-1}=-\big[a_1y_{n-1}+a_2y_{n-2}+\cdots+a_py_{n-p}\big] \tag{12.2.4}$$

In this case, the best prediction of y_n depends only on the past p samples $\{y_{n-1},y_{n-2},\dots,y_{n-p}\}$. The infinite set of equations (12.1.10) or (12.1.11) are still satisfied even though only the first $p+1$ coefficients $\{1,a_1,a_2,\dots,a_p\}$ are nonzero.

The $(p+1)\times(p+1)$ portion of Eq. (12.1.11) is sufficient to determine the $(p+1)$ model parameters $\{a_1,a_2,\dots,a_p;\sigma_\epsilon^2\}$:

$$\begin{bmatrix} R_{yy}(0) & R_{yy}(1) & R_{yy}(2) & \cdots & R_{yy}(p)\\ R_{yy}(1) & R_{yy}(0) & R_{yy}(1) & \cdots & R_{yy}(p-1)\\ R_{yy}(2) & R_{yy}(1) & R_{yy}(0) & \cdots & R_{yy}(p-2)\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ R_{yy}(p) & R_{yy}(p-1) & R_{yy}(p-2) & \cdots & R_{yy}(0) \end{bmatrix}\begin{bmatrix}1\\ a_1\\ a_2\\ \vdots\\ a_p\end{bmatrix}=\begin{bmatrix}\sigma_\epsilon^2\\ 0\\ 0\\ \vdots\\ 0\end{bmatrix} \tag{12.2.5}$$

Such equations may be solved efficiently by Levinson's algorithm, which requires $O(p^2)$ operations and $O(p)$ storage locations to obtain the a_is instead of $O(p^3)$ and $O(p^2)$, respectively, that would be required if the inverse of the autocorrelation matrix

R_{yy} were to be computed. The finite set of model parameters $\{a_1, a_2, \dots, a_p; \sigma_\epsilon^2\}$ determines the signal model of y_n completely. Setting $z = e^{j\omega}$ into Eq. (12.1.5)) we find a simple *parametric representation* of the power spectrum of the AR signal y_n

$$S_{yy}(\omega) = \frac{\sigma_\epsilon^2}{|A(\omega)|^2} = \frac{\sigma_\epsilon^2}{|1 + a_1 e^{-j\omega} + a_2 e^{-2j\omega} + \cdots + a_p e^{-j\omega p}|^2} \tag{12.2.6}$$

In practice, the normal equations (12.2.5) provide a means of determining approximate estimates for the model parameters $\{a_1, a_2, \dots, a_p; \sigma_\epsilon^2\}$. Typically, a block of length N of recorded data is available

$$\boxed{y_0, y_1, y_2, \dots, y_{N-1}}$$

There are many different methods of extracting reasonable estimates of the model parameters using this block of data. We mention: (1) the autocorrelation or Yule-Walker method, (2) the covariance method, and (3) Burg's method. There are also some variations of these methods. The first method, the Yule-Walker method, is perhaps the most obvious and straightforward one. In the normal equations (12.2.5), one simply replaces the ensemble autocorrelations $R_{yy}(k)$ by the corresponding sample autocorrelations computed from the given block of data; that is,

$$\hat{R}_{yy}(k) = \frac{1}{N}\sum_{n=0}^{N-1-k} y_{n+k}y_n\,, \quad \text{for } 0 \le k \le p \tag{12.2.7}$$

where only the first $p + 1$ lags are needed in Eq. (12.2.5). We must have, of course, $p \le N - 1$. As discussed in Sec. 1.13, the resulting estimates of the model parameters $\{\hat{a}_1, \hat{a}_2, \dots, \hat{a}_p; \hat{\sigma}_\epsilon^2\}$ may be used now in a number of ways; examples include obtaining an *estimate* of the power spectrum of the sequence y_n

$$\hat{S}_{yy}(\omega) = \frac{\hat{\sigma}_\epsilon^2}{|\hat{A}(\omega)|^2} = \frac{\hat{\sigma}_\epsilon^2}{|1 + \hat{a}_1 e^{-j\omega} + \hat{a}_2 e^{-2j\omega} + \cdots + \hat{a}_p e^{-j\omega p}|^2}$$

or, representing the block of N samples y_n in terms of a few (i.e., $p+1$) filter parameters. To synthesize the original samples one would generate white noise ϵ_n of variance $\hat{\sigma}_\epsilon^2$ and send it through the generator filter whose coefficients are the estimated values; that is, the filter

$$\hat{B}(z) = \frac{1}{\hat{A}(z)} = \frac{1}{1 + \hat{a}_1 z^{-1} + \hat{a}_2 z^{-2} + \cdots + \hat{a}_p z^{-p}}$$

The Yule-Walker analysis procedure, also referred to as the autocorrelation method of linear prediction [917], is summarized in Fig. 12.2.1.

12.3 Linear Prediction and the Levinson Recursion

In the last section, we saw that if the signal being predicted is autoregressive of order p, then the optimal linear predictor collapses to a pth order predictor. The infinite dimensional Wiener filtering problem collapses to a finite dimensional one. A geometrical way to understand this property is to say that the projection of y_n on the subspace spanned by the entire past $\{y_{n-i}\,,\ 1 \le i < \infty\}$ is the same as the projection of y_n onto the subspace spanned only by the past p samples; namely, $\{y_{n-i}\,,\ 1 \le i \le p\}$. This is a consequence of the difference equation (12.2.3) generating y_n.

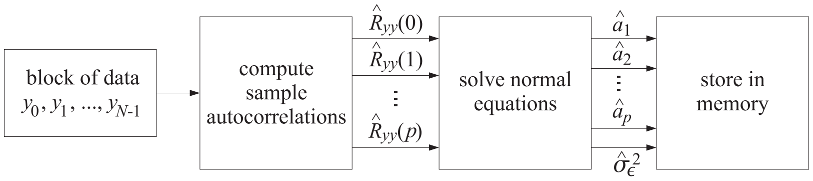


Fig. 12.2.1 Yule-Walker Analysis Algorithm.

If the process y_n is not autoregressive, these two projections will be different. For any given p, the projection of y_n onto the past p samples will still provide the best linear prediction of y_n that can be made on the basis of these p samples. As p increases, more and more past information is taken into account, and we expect the prediction of y_n to become better and better in the sense of yielding a smaller mean-square prediction error.

In this section, we consider the finite-past prediction problem and discuss its efficient solution via the Levinson recursion [915–928]. For sufficiently large values of p, it may be considered to be an adequate approximation to the full prediction problem and hence also to the modeling problem.

Consider a stationary time series y_n with autocorrelation function $R(k) = E[y_{n+k}y_n]$. For any given p, we seek the best linear predictor of the form

$$\hat{y}_n = -[a_1 y_{n-1} + a_2 y_{n-2} + \cdots + a_p y_{n-p}] \tag{12.3.1}$$

The p prediction coefficients $\{a_1, a_2, \dots, a_p\}$ are chosen to minimize the mean-square prediction error

$$\mathcal{E} = E[e_n^2] \tag{12.3.2}$$

where e_n is the prediction error

$$e_n = y_n - \hat{y}_n = y_n + a_1 y_{n-1} + a_2 y_{n-2} + \cdots + a_p y_{n-p} \tag{12.3.3}$$

Differentiating Eq. (12.3.2) with respect to each coefficient a_i, $i = 1, 2, \dots, p$, yields the orthogonality equations

$$E[e_n y_{n-i}] = 0\,, \quad \text{for } i = 1, 2, \dots, p \tag{12.3.4}$$

which express the fact that the optimal predictor $\hat{y}_n$ is the projection onto the span of the past p samples; that is, $\{y_{n-i}\,,\ i = 1, 2, \dots, p\}$. Inserting the expression (12.3.3) for e_n into Eq. (12.3.4), we obtain p linear equations for the coefficients

$$\sum_{j=0}^{p} a_j E[y_{n-j}y_{n-i}] = \sum_{j=0}^{p} R(i-j)a_j = 0\,, \quad \text{for } i = 1, 2, \dots, p \tag{12.3.5}$$

Using the conditions (12.3.4) we also find for the *minimized* value of

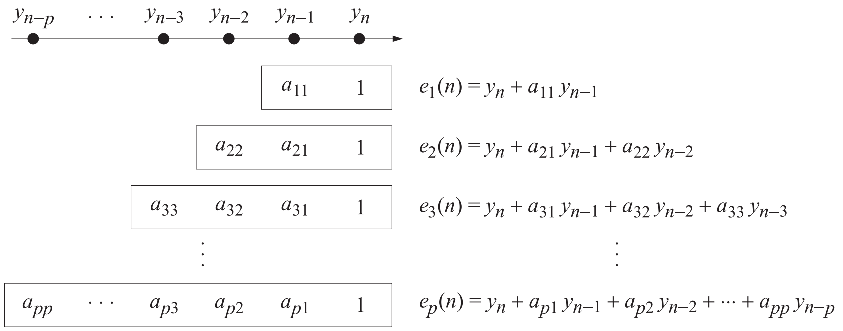
$$\sigma_e^2 = \mathcal{E} = E[e_n^2] = E[e_n y_n] = \sum_{j=0}^{p} R(j)a_j \tag{12.3.6}$$

Equations (12.3.5) and (12.3.6) can be combined into the $(p+1)\times(p+1)$ matrix equation

$$\begin{bmatrix} R(0) & R(1) & R(2) & \cdots & R(p) \\ R(1) & R(0) & R(1) & \cdots & R(p-1) \\ R(2) & R(1) & R(0) & \cdots & R(p-2) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ R(p) & R(p-1) & R(p-2) & \cdots & R(0) \end{bmatrix} \begin{bmatrix} 1 \\ a_1 \\ a_2 \\ \vdots \\ a_p \end{bmatrix} = \begin{bmatrix} \sigma_e^2 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{12.3.7}$$

which is identical to Eq. (12.2.5) for the autoregressive case. It is also the *truncated* version of the infinite matrix equation (12.1.11) for the full prediction problem.

Instead of solving the normal equations (12.3.7) directly, we would like to embed this problem into a whole class of similar problems; namely, those of determining the best linear predictors of orders $p = 1$, $p = 2$, $p = 3, \dots$, and so on. This approach will lead to Levinson's algorithm and to the so-called lattice realizations of linear prediction filters. Pictorially this class of problems is illustrated below



y_{n-p}
$\cdots$
y_{n-3}
y_{n-2}
y_{n-1}
y_n
a_{11} 1
$e_1(n)=y_n+a_{11}y_{n-1}$
a_{22} a_{21} 1
$e_2(n)=y_n+a_{21}y_{n-1}+a_{22}y_{n-2}$
a_{33} a_{32} a_{31} 1
$e_3(n)=y_n+a_{31}y_{n-1}+a_{32}y_{n-2}+a_{33}y_{n-3}$
$\vdots$
a_{pp} $\cdots$ a_{p3} a_{p2} a_{p1} 1
$e_p(n)=y_n+a_{p1}y_{n-1}+a_{p2}y_{n-2}+\cdots+a_{pp}y_{n-p}$

where $[1,a_{11}]$, $[1,a_{21},a_{22}]$, $[1,a_{31},a_{32},a_{33}],\dots$, represent the best predictors of orders $p = 1,2,3,\dots$, respectively. It was necessary to attach an extra index indicating the order of the predictor. Levinson's algorithm is an iterative procedure that constructs the next predictor from the previous one. In the process, all optimal predictors of lower orders are also computed. Consider the predictors of orders p and $p+1$, below

y_{n-p-1}	y_{n-p}	$\cdots$	y_{n-2}	y_{n-1}	y_n
	a_{pp}	$\cdots$	a_{p2}	a_{p1}	1
$a_{p+1,p+1}$	$a_{p+1,p}$	$\cdots$	$a_{p+1,2}$	$a_{p+1,1}$	1

$$e_p(n) = y_n + a_{p1}y_{n-1} + a_{p2}y_{n-2} + \cdots + a_{pp}y_{n-p}$$

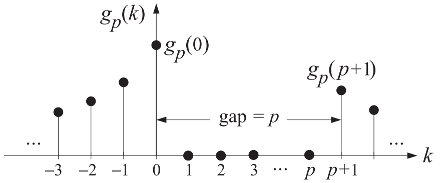
$$e_{p+1}(n) = y_n + a_{p+1,1}y_{n-1} + a_{p+1,2}y_{n-2} + \cdots + a_{p+1,p+1}y_{n-p-1}$$

Our objective is to construct the latter in terms of the former. We will use the approach of Robinson and Treitel, based on gapped functions [925]. Suppose that the best predictor of order p, $[1,a_{p1},a_{p2},\dots,a_{pp}]$, has already been constructed. The corresponding gapped function is

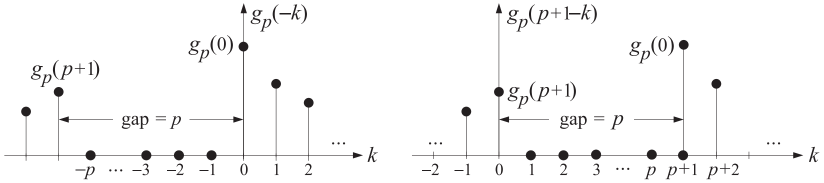
$$g_p(k) = E[e_p(n)y_{n-k}] = E\left[\left(\sum_{i=0}^{p} a_{pi}y_{n-i}\right)y_{n-k}\right] = \sum_{i=0}^{p} a_{pi}R(k-i) \tag{12.3.8}$$

It has a gap of length p as shown , that is,

$$g_p(k) = 0\,, \quad \text{for } 1 \le k \le p$$



These gap conditions are the same as the orthogonality equations (12.3.4). Using $g_p(k)$ we now construct a new gapped function $g_{p+1}(k)$ of gap $p+1$. To do this, first we reflect $g_p(k)$ about the origin; that is, $g_p(k)\to g_p(-k)$. The reflected function has a gap of length p but at negatives times. A delay of $(p+1)$ time units will realign this gap with the original gap. This follows because if $1 \le k \le p$, then $1 \le p+1-k \le p$. The reflected-delayed function will be $g_p(p+1-k)$. These operations are shown in the following figure



Since both $g_p(k)$ and $g_p(p+1-k)$ have exactly the same gap, it follows that so will any linear combination of them. Therefore,

$$g_{p+1}(k) = g_p(k) - \gamma_{p+1}g_p(p+1-k) \tag{12.3.9}$$

will have a gap of length at least p. We now select the parameter γ_{p+1} so that $g_{p+1}(k)$ acquires an *extra* gap point; its gap is now of length $p+1$. The extra gap condition is

$$g_{p+1}(p+1) = g_p(p+1) - \gamma_{p+1}g_p(0) = 0$$

which may be solved for

$$\gamma_{p+1} = \frac{g_p(p+1)}{g_p(0)}$$

Evaluating Eq. (12.3.8) at $k = p+1$, and using the fact that the value of the gapped function at $k = 0$ is the minimized value of the mean-squared error, that is,

$$E_p = E\big[e_p^2(n)\big] = E[e_p(n)y_n] = g_p(0) \tag{12.3.10}$$

we finally find

$$\gamma_{p+1} = \frac{\Delta_p}{E_p} \tag{12.3.11}$$

where we set $\Delta_p = g_p(p+1)$

$$\Delta_p = \sum_{i=0}^{p} a_{pi}R(p+1-i) = \big[R(p+1), R(p), R(p-1), \dots, R(1)\big]\begin{bmatrix}1\\ a_{p1}\\ a_{p2}\\ \vdots\\ a_{pp}\end{bmatrix} \tag{12.3.12}$$

The coefficients γ_{p+1} are called *reflection, PARCOR*, or *Schur coefficients*. This terminology will become clear later. Evaluating Eq. (12.3.9) at $k=0$ and using $g_p(p+1) = \gamma_{p+1}g_p(0)$, we also find a recursion for the quantity $E_{p+1} = g_{p+1}(0)$

$$E_{p+1} = g_{p+1}(0) = g_p(0) - \gamma_{p+1}g_p(p+1) = g_p(0) - \gamma_{p+1}\cdot\gamma_{p+1}g_p(0)\,, \quad \text{or,}$$

$$E_{p+1} = (1-\gamma_{p+1}^2)E_p \tag{12.3.13}$$

This represents the minimum value of the mean-square prediction error $E\big[e_{p+1}^2(n)\big]$ for the predictor of order $p+1$. Since both E_p and E_{p+1} are nonnegative, it follows that the factor $(1-\gamma_{p+1}^2)$ will be nonnegative and less than one. It represents the improvement in the prediction obtained by using a predictor of order $p+1$ instead of a predictor of order p. It also follows that γ_{p+1} has magnitude less than one, $|\gamma_{p+1}| \le 1$.

To find the new prediction coefficients, $a_{p+1,i}$, we use the fact that the gapped functions are equal to the convolution of the corresponding prediction-error filters with the autocorrelation function of y_n:

$$g_p(k) = \sum_{i=0}^{p} a_{pi}R(k-i) \quad \Rightarrow \quad G_p(z) = A_p(z)S_{yy}(z)$$

$$g_{p+1}(k) = \sum_{i=0}^{p+1} a_{p+1,i}R(k-i) \quad \Rightarrow \quad G_{p+1}(z) = A_{p+1}(z)S_{yy}(z)$$

where $S_{yy}(z)$ represents the power spectral density of y_n. Taking z-transforms of both sides of Eq. (12.3.9), we find

$$G_{p+1}(z) = G_p(z) - \gamma_{p+1}z^{-(p+1)}G_p(z^{-1})\,, \quad \text{or,}$$

$$A_{p+1}(z)S_{yy}(z) = A_p(z)S_{yy}(z) - \gamma_{p+1}z^{-(p+1)}A_p(z^{-1})S_{yy}(z^{-1})$$

where we used the fact that the reflected gapped function $g_p(-k)$ has z-transform $G_p(z^{-1})$, and therefore the delayed (by $p+1$) as well as reflected gapped function $g_p(p+1-k)$ has z-transform $z^{-(p+1)}G_p(z^{-1})$. Since $S_{yy}(z) = S_{yy}(z^{-1})$ because of the symmetry relations $R(k) = R(-k)$, it follows that $S_{yy}(z)$ is a common factor in all the terms. Therefore, we obtain a relationship between the new best prediction-error filter $A_{p+1}(z)$and the old one $A_p(z)$

$$A_{p+1}(z) = A_p(z) - \gamma_{p+1}z^{-(p+1)}A_p(z^{-1}) \qquad \text{(Levinson recursion)} \tag{12.3.14}$$

Taking inverse z-transforms, we find

$$\begin{bmatrix}1\\ a_{p+1,1}\\ a_{p+1,2}\\ \vdots\\ a_{p+1,p}\\ a_{p+1,p+1}\end{bmatrix} = \begin{bmatrix}1\\ a_{p1}\\ a_{p2}\\ \vdots\\ a_{pp}\\ 0\end{bmatrix} - \gamma_{p+1}\begin{bmatrix}0\\ a_{pp}\\ a_{p,p-1}\\ \vdots\\ a_{p1}\\ 1\end{bmatrix} \tag{12.3.15}$$

which can also be written as

$$a_{p+1,i} = a_{pi} - \gamma_{p+1}a_{p,p+1-i}\,, \quad \text{for } 1 \le i \le p$$

$$a_{p+1,p+1} = -\gamma_{p+1}$$

Introducing the *reverse* polynomial $A_p^R(z) = z^{-p}A_p(z^{-1})$, we may write Eq. (12.3.14) as

$$A_{p+1}(z) = A_p(z) - \gamma_{p+1}z^{-1}A_p^R(z) \tag{12.3.16}$$

Taking the reverse of both sides, we find

$$A_{p+1}(z^{-1}) = A_p(z^{-1}) - \gamma_{p+1}z^{p+1}A_p(z)$$

$$A_{p+1}^R(z) = z^{-(p+1)}A_{p+1}(z^{-1}) = z^{-(p+1)}A_p(z^{-1}) - \gamma_{p+1}A_p(z)\,, \quad \text{or,}$$

$$A_{p+1}^R(z) = z^{-1}A_p^R(z) - \gamma_{p+1}A_p(z) \tag{12.3.17}$$

Equation (12.3.17) is, in a sense, redundant, but it will prove convenient to think of the Levinson recursion as a recursion on both the forward, $A_p(z)$, and the reverse, $A_p^R(z)$, polynomials. Equations (12.3.16) and Eq. (12.3.17) may be combined into a 2×2 matrix recursion equation, referred to as the *forward Levinson recursion*:

$$\begin{bmatrix}A_{p+1}(z)\\ A_{p+1}^R(z)\end{bmatrix} = \begin{bmatrix}1 & -\gamma_{p+1}z^{-1}\\ -\gamma_{p+1} & z^{-1}\end{bmatrix}\begin{bmatrix}A_p(z)\\ A_p^R(z)\end{bmatrix} \qquad \text{(forward recursion)} \tag{12.3.18}$$

The recursion is initialized at $p=0$ by setting

$$A_0(z) = A_0^R(z) = 1 \quad \text{and} \quad E_0 = R(0) = E[y_n^2] \tag{12.3.19}$$

which corresponds to no prediction at all. We summarize the computational steps of the Levinson algorithm:

1. Initialize at $p=0$ using Eq. (12.3.19).
2. At stage p, the filter $A_p(z)$ and error E_p are available.
3. Using Eq. (12.3.11), compute γ_{p+1}.
4. Using Eq. (12.3.14) or Eq. (12.3.18), determine the new polynomial $A_{p+1}(z)$.
5. Using Eq. (12.3.13), update the mean-square prediction error to E_{p+1}.
6. Go to stage $p+1$.

The iteration may be continued until the final desired order is reached. The dependence on the autocorrelation $R(k)$ of the signal y_n is entered through Eq. (12.3.11) and $E_0 = R(0)$. To reach stage p, only the $p+1$ autocorrelation lags $\{R(0), R(1), \dots, R(p)\}$ are required. At the pth stage, the iteration already has provided all the prediction filters of lower order, and all the previous reflection coefficients. Thus, an alternative parametrization of the pth order predictor is in terms of the sequence of reflection coefficients $\{\gamma_1, \gamma_2, \dots, \gamma_p\}$ and the prediction error E_p

$$\{E_p, a_{p1}, a_{p2}, \dots, a_{pp}\} \quad \Leftrightarrow \quad \{E_p, \gamma_1, \gamma_2, \dots, \gamma_p\}$$

One may pass from one parameter set to another. And both sets are equivalent to the autocorrelation set $\{R(0), R(1), \dots, R(p)\}$. The alternative parametrization of the autocorrelation function $R(k)$ of a stationary random sequence in terms of the equivalent set of reflection coefficients is a general result [929,930], and has also been extended to the multichannel case [931].

If the process y_n is autoregressive of order p, then as soon as the Levinson recursion reaches this order, it will provide the autoregressive coefficients $\{a_1, a_2, \dots, a_p\}$ which are also the best prediction coefficients for the full (i.e., based on the infinite past) prediction problem. Further continuation of the Levinson recursion will produce nothing new—all prediction coefficients (and all reflection coefficients) of order higher than p will be zero, so that $A_q(z) = A_p(z)$ for all $q > p$.

The four functions **lev**, **frwlev**, **bkwlev**, and **rlev** allow the passage from one parameter set to another. The function **lev** is an implementation of the computational sequence outlined above. The input to the function is the final desired order of the predictor, say M, and the vector of autocorrelation lags $\{R(0), R(1), ..., R(M)\}$. Its output is the lower-triangular matrix L whose rows are the *reverse* of all the lower order prediction-error filters. For example, for $M = 4$ the matrix L would be

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ a_{11} & 1 & 0 & 0 & 0 \\ a_{22} & a_{21} & 1 & 0 & 0 \\ a_{33} & a_{32} & a_{31} & 1 & 0 \\ a_{44} & a_{43} & a_{42} & a_{41} & 1 \end{bmatrix} \tag{12.3.20}$$

The first column of L contains the negatives of all the reflection coefficients. This follows from the Levinson recursion (12.3.14) which implies that the negative of the highest coefficient of the pth prediction-error filter is the pth reflection coefficient; namely,

$$\gamma_p = -a_{pp}\,, \quad p = 1, 2, \dots, M \tag{12.3.21}$$

This choice for L is justified below and in Sec. 12.9. The function **lev** also produces the vector of mean-square prediction errors $\{E_0, E_1, \dots, E_M\}$ according to the recursion (12.3.13).

The function **frwlev** is an implementation of the forward Levinson recursion (12.3.18) or (12.3.15). Its input is the set of reflection coefficients $\{\gamma_1, \gamma_2, \dots, \gamma_M\}$ and its output is the set of all prediction-error filters up to order M, that is, $A_p(z)$, $p = 1, 2, \dots, M$. Again, this output is arranged into the matrix L.

The function **bkwlev** is the inverse operation to **frwlev**. Its input is the vector of prediction-error filter coefficients $[1, a_{M1}, a_{M2}, \dots, a_{MM}]$ of the final order M, and its output is the matrix L containing all the lower order prediction-error filters. The set of reflection coefficients are extracted from the first column of L. This function is based on the inverse of the matrix equation (12.3.18). Shifting p down by one unit, we write Eq. (12.3.18) as

$$\begin{bmatrix} A_p(z) \\ A_p^R(z) \end{bmatrix} = \begin{bmatrix} 1 & -\gamma_p z^{-1} \\ -\gamma_p & z^{-1} \end{bmatrix} \begin{bmatrix} A_{p-1}(z) \\ A_{p-1}^R(z) \end{bmatrix} \tag{12.3.22}$$

Its inverse is

$$\begin{bmatrix} A_{p-1}(z) \\ A_{p-1}^R(z) \end{bmatrix} = \frac{1}{1-\gamma_p^2} \begin{bmatrix} 1 & \gamma_p \\ \gamma_p z & z \end{bmatrix} \begin{bmatrix} A_p(z) \\ A_p^R(z) \end{bmatrix} \quad \text{(backward recursion)} \tag{12.3.23}$$

At each stage p, start with $A_p(z)$ and extract $\gamma_p = -a_{pp}$ from the highest coefficient of $A_p(z)$. Then, use Eq. (12.3.23) to obtain the polynomial $A_{p-1}(z)$. The iteration begins at the given order M and proceeds downwards to $p = M-1, M-2, \dots, 1, 0$.

The function **rlev** generates the set of autocorrelation lags $\{R(0), R(1), ..., R(M)\}$ from the knowledge of the final prediction-error filter $A_M(z)$ and final prediction error E_M. It calls **bkwlev** to generate all the lower order prediction-error filters, and then it reconstructs the autocorrelation lags using the gapped function condition $g_p(p) = \sum_{i=0}^{p} a_{pi} R(p-i) = 0$, which may be solved for $R(p)$ in terns of $R(p-i)$, $i = 1, 2, \dots, p$, as follows:

$$R(p) = -\sum_{i=1}^{p} a_{pi} R(p-i)\,, \quad p = 1, 2, \dots, M \tag{12.3.24}$$

For example, the first few iterations of Eq. (12.3.24) will be:

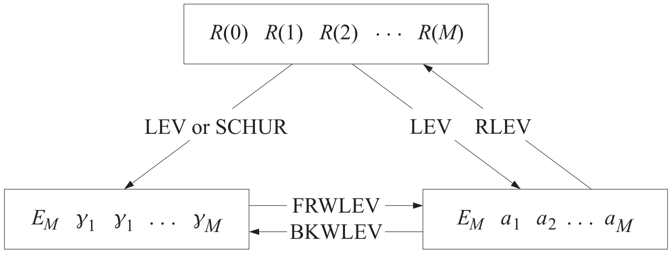
$$\begin{aligned} R(1) &= -\big[a_{11}R(0)\big] \\ R(2) &= -\big[a_{21}R(1) + a_{22}R(0)\big] \\ R(3) &= -\big[a_{31}R(2) + a_{32}R(1) + a_{33}R(0)\big] \end{aligned}$$

To get this recursion started, the value of $R(0)$ may be obtained from Eq. (12.3.13). Using Eq. (12.3.13) repeatedly, and $E_0 = R(0)$ we find

$$E_M = (1-\gamma_1^2)(1-\gamma_2^2)\cdots(1-\gamma_M^2)R(0) \tag{12.3.25}$$

Since the reflection coefficients are already known (from the call to **bkwlev**) and E_M is given, this equation provides the right value for $R(0)$.

The function **schur**, based on the Schur algorithm and discussed in Sec. 12.10, is an alternative to **lev**. The logical interconnection of these functions is shown below.



Example 12.3.1: Given the autocorrelation lags

$$\{R(0),R(1),R(2),R(3),R(4)\}=\{128,-64,80,-88,89\}$$

Find all the prediction-error filters $A_p(z)$ up to order four, the four reflection coefficients, and the corresponding mean-square prediction errors. Below, we simply state the results obtained using the function **lev**:

$$\begin{aligned}A_1(z)&=1+0.5z^{-1}\\A_2(z)&=1+0.25z^{-1}-0.5z^{-2}\\A_3(z)&=1-0.375z^{-2}+0.5z^{-3}\\A_4(z)&=1-0.25z^{-1}-0.1875z^{-2}+0.5z^{-3}-0.5z^{-4}\end{aligned}$$

The reflection coefficients are the negatives of the highest coefficients; thus,

$$\{\gamma_1,\gamma_2,\gamma_3,\gamma_4\}=\{-0.5,\ 0.5,\ -0.5,\ 0.5\}$$

The vector of mean-squared prediction errors is given by

$$\{E_0,E_1,E_2,E_3,E_4\}=\{128,\ 96,\ 72,\ 54,\ 40.5\}$$

Sending the above vector of reflection coefficients through the function **frwlev** would generate the above set of polynomials. Sending the coefficients of $A_4(z)$ through **bkwlev** would generate the same set of polynomials. Sending the coefficients of $A_4(z)$ and $E_4=40.5$ through **rlev** would recover the original autocorrelation lags $R(k)$, $k=0,1,2,3,4$. □

The *Yule-Walker* method (see Sec. 12.2) can be used to extract the linear prediction parameters from a given set of signal samples. From a given length-N block of data

$$\boxed{y_0,y_1,y_2,\dots,y_{N-1}}$$

compute the sample autocorrelations $\{\hat R(0),\hat R(1),\dots,\hat R(M)\}$ using, for example, Eq. (12.2.7), and send them through the Levinson recursion. The **yw** implements the Yule-Walker method. The input to the function is the data vector of samples $\{y_0,y_1,\dots,y_{N-1}\}$ and the desired final order M of the predictor. Its output is the set of all prediction-error filters up to order M, arranged in the matrix L, and the vector of mean-squared prediction errors up to order M, that is, $\{E_0,E_1,\dots,E_M\}$

Example 12.3.2: Given the signal samples

$$\{y_0,y_1,y_2,y_3,y_4\}=\{1,1,1,1,1\}$$

determine all the prediction-error filters up to order four. Using the fourth order predictor, predict the sixth value in the above sequence, i.e., the value of y_5.

The sample autocorrelation of the above signal is easily computed using the methods of Chapter 1. We find (ignoring the $1/N$ normalization factor):

$$\{\hat R(0),\hat R(1),\hat R(2),\hat R(3),\{\hat R(4)\}=\{5,4,3,2,1\}$$

Sending these lags through the function **lev** we find the prediction-error filters:

$$\begin{aligned}A_1(z)&=1-0.8z^{-1}\\A_2(z)&=1-0.889z^{-1}+0.111z^{-2}\\A_3(z)&=1-0.875z^{-1}+0.125z^{-3}\\A_4(z)&=1-0.857z^{-1}+0.143z^{-4}\end{aligned}$$

Therefore, the fourth order prediction of y_n given by Eq. (12.3.1) is

$$\hat y_n=0.857y_{n-1}-0.143y_{n-4}$$

which gives $\hat y_5=0.857-0.143=0.714$. □

The results of this section can also be derived from those of Sec. 1.8 by invoking stationarity and making the proper identification of the various quantities. The data vector $\mathbf{y}$ and the subvectors $\bar{\mathbf{y}}$ and $\tilde{\mathbf{y}}$ are identified with $\mathbf{y}=\mathbf{y}_{p+1}(n)$, $\bar{\mathbf{y}}=\mathbf{y}_p(n)$, and $\tilde{\mathbf{y}}=\mathbf{y}_p(n-1)$, where

$$\mathbf{y}_{p+1}(n)=\begin{bmatrix}y_n\\y_{n-1}\\\vdots\\y_{n-p}\\y_{n-p-1}\end{bmatrix},\quad\mathbf{y}_p(n)=\begin{bmatrix}y_n\\y_{n-1}\\\vdots\\y_{n-p}\end{bmatrix},\quad\mathbf{y}_p(n-1)=\begin{bmatrix}y_{n-1}\\y_{n-2}\\\vdots\\y_{n-p-1}\end{bmatrix}\qquad(12.3.26)$$

It follows from stationarity that the autocorrelation matrices of these vectors are independent of the absolute time instant n; therefore, we write

$$R_p=E[\mathbf{y}_p(n)\mathbf{y}_p(n)^T]=E[\mathbf{y}_p(n-1)\mathbf{y}_p(n-1)^T],\quad R_{p+1}=E[\mathbf{y}_{p+1}(n)\mathbf{y}_{p+1}(n)^T]$$

It is easily verified that R_p is the order-p autocorrelation matrix defined in Eq. (12.3.7) and that the order-$(p+1)$ autocorrelation matrix R_{p+1} admits the block decompositions

$$R_{p+1}=\left[\begin{array}{c|ccc}R(0)&R(1)&\cdots&R(p+1)\\\hline R(1)&&&\\\vdots&&R_p&\\R(p+1)&&&\end{array}\right]=\left[\begin{array}{ccc|c}&&&R(p+1)\\&R_p&&\vdots\\&&&R(1)\\\hline R(p+1)&\cdots&R(1)&R(0)\end{array}\right]$$

It follows, in the notation of Sec. 1.8, that $\bar R=\tilde R=R_p$ and $\rho_a=\rho_b=R(0)$, and

$$\mathbf{r}_a=\begin{bmatrix}R(1)\\\vdots\\R(p+1)\end{bmatrix},\quad\mathbf{r}_b=\begin{bmatrix}R(p+1)\\\vdots\\R(1)\end{bmatrix}$$

Thus, $\mathbf{r}_a$ and $\mathbf{r}_b$ are the reverse of each other. It follows that the backward predictors are the reverse of the forward ones. Therefore, Eq. (12.3.14) is the same as Eq. (1.8.40), with the identifications

$$\mathbf{a}=\mathbf{a}_{p+1}\,,\quad\mathbf{b}=\mathbf{b}_{p+1}\,,\quad\bar{\mathbf{a}}=\tilde{\mathbf{a}}=\mathbf{a}_p\,,\quad\bar{\mathbf{b}}=\tilde{\mathbf{b}}=\mathbf{b}_p$$

where

$$\mathbf{a}_{p+1} = \begin{bmatrix} 1 \\ a_{p+1,1} \\ \vdots \\ a_{p+1,p} \\ a_{p+1,p+1} \end{bmatrix}, \quad \mathbf{b}_{p+1} = \begin{bmatrix} a_{p+1,p+1} \\ a_{p+1,p} \\ \vdots \\ a_{p+1,1} \\ 1 \end{bmatrix} \quad \mathbf{a}_p = \begin{bmatrix} 1 \\ a_{p1} \\ \vdots \\ a_{pp} \end{bmatrix}, \quad \mathbf{b}_p = \begin{bmatrix} a_{pp} \\ \vdots \\ a_{p1} \\ 1 \end{bmatrix}$$

Symbolically, $\mathbf{b}_p = \mathbf{a}_p^R$, $\mathbf{b}_{p+1} = \mathbf{a}_{p+1}^R$. We have $\bar{E}_a = \tilde{E}_b = E_p$ and $\gamma_a = \gamma_b = \gamma_{p+1}$. Thus, Eq. (12.3.15) may be written as

$$\mathbf{a}_{p+1} = \begin{bmatrix} \mathbf{a}_p \\ 0 \end{bmatrix} - \gamma_{p+1} \begin{bmatrix} 0 \\ \mathbf{b}_p \end{bmatrix} = \begin{bmatrix} \mathbf{a}_p \\ 0 \end{bmatrix} - \gamma_{p+1} \begin{bmatrix} 0 \\ \mathbf{a}_p^R \end{bmatrix} \tag{12.3.27}$$

The normal Eqs. (12.3.7) can be written for orders p and $p+1$ in the compact form of Eqs. (1.8.38) and (1.8.12)

$$R_p\mathbf{a}_p = E_p\mathbf{u}_p\,, \quad R_{p+1}\mathbf{a}_{p+1} = E_{p+1}\mathbf{u}_{p+1}\,, \quad \mathbf{u}_p = \begin{bmatrix} 1 \\ \mathbf{0} \end{bmatrix}, \quad \mathbf{u}_{p+1} = \begin{bmatrix} \mathbf{u}_p \\ 0 \end{bmatrix} \tag{12.3.28}$$

Recognizing that Eq. (12.3.12) can be written as $\Delta_p = \mathbf{a}_p^T\mathbf{r}_b$, it follows that the reflection coefficient equation (12.3.11) is the same as (1.8.42). The rows of the matrix L defined by Eq. (12.3.20) are the reverse of the forward predictors; that is, the backward predictors of successive orders. Thus, L is the same as that defined in Eq. (1.8.13). The rows of the matrix U defined in Eq. (1.8.30) are the forward predictors, with the first row being the predictor of highest order. For example,

$$U = \begin{bmatrix} 1 & a_{41} & a_{42} & a_{43} & a_{44} \\ 0 & 1 & a_{31} & a_{32} & a_{33} \\ 0 & 0 & 1 & a_{21} & a_{22} \\ 0 & 0 & 0 & 1 & a_{11} \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Comparing L with U, we note that one is obtained from the other by reversing its rows and then its columns; formally, $U = JLJ$, where J is the corresponding reversing matrix.

12.4 Levinson's Algorithm in Matrix Form

In this section, we illustrate the mechanics of the Levinson recursion—cast in matrix form—by explicitly carrying out a few of the recursions given in Eq. (12.3.15). The objective of such recursions is to solve normal equations of the type

$$\begin{bmatrix} R_0 & R_1 & R_2 & R_3 \\ R_1 & R_0 & R_1 & R_2 \\ R_2 & R_1 & R_0 & R_1 \\ R_3 & R_2 & R_1 & R_0 \end{bmatrix} \begin{bmatrix} 1 \\ a_{31} \\ a_{32} \\ a_{33} \end{bmatrix} = \begin{bmatrix} E_3 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

for the unknowns $\{E_3, a_{31}, a_{32}, a_{33}\}$. The corresponding prediction-error filter is

$$A_3(z) = 1 + a_{31}z^{-1} + a_{32}z^{-2} + a_{33}z^{-3}$$

and the minimum value of the prediction error is E_3. The solution is obtained in an iterative manner, by solving a family of similar matrix equations of lower dimensionality. Starting at the upper left corner,

$$\begin{array}{|c|c|c|c|} \hline R_0 & R_1 & R_2 & R_3 \\ \hline R_1 & R_0 & R_1 & R_2 \\ \hline R_2 & R_1 & R_0 & R_1 \\ \hline R_3 & R_2 & R_1 & R_0 \\ \hline \end{array}$$

the R matrices are successively enlarged until the desired dimension is reached (4×4 in this example). Therefore, one successively solves the matrix equations

$$[R_0][1] = [E_0]\,, \quad \begin{bmatrix} R_0 & R_1 \\ R_1 & R_0 \end{bmatrix} \begin{bmatrix} 1 \\ a_{11} \end{bmatrix} = \begin{bmatrix} E_1 \\ 0 \end{bmatrix}, \quad \begin{bmatrix} R_0 & R_1 & R_2 \\ R_1 & R_0 & R_1 \\ R_2 & R_1 & R_0 \end{bmatrix} \begin{bmatrix} 1 \\ a_{21} \\ a_{22} \end{bmatrix} = \begin{bmatrix} E_2 \\ 0 \\ 0 \end{bmatrix}$$

The solution of each problem is obtained *in terms of* the solution of the previous one. In this manner, the final solution is gradually built up. In the process, one also finds all the lower order prediction-error filters.

The iteration is based on two key properties of the autocorrelation matrix: first, the autocorrelation matrix of a given size contains as subblocks all the lower order autocorrelation matrices; and second, the autocorrelation matrix is reflection invariant. That is, it remains invariant under interchange of its columns and then its rows. This interchanging operation is equivalent to the similarity transformation by the "reversing" matrix J having 1's along its anti-diagonal, e.g.,

$$J = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \tag{12.4.1}$$

The invariance property means that the autocorrelation matrix commutes with the matrix J

$$JRJ^{-1} = R \tag{12.4.2}$$

This property immediately implies that if the matrix equation is satisfied:

$$\begin{bmatrix} R_0 & R_1 & R_2 & R_3 \\ R_1 & R_0 & R_1 & R_2 \\ R_2 & R_1 & R_0 & R_1 \\ R_3 & R_2 & R_1 & R_0 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

then the following equation is also satisfied:

$$\begin{bmatrix} R_0 & R_1 & R_2 & R_3 \\ R_1 & R_0 & R_1 & R_2 \\ R_2 & R_1 & R_0 & R_1 \\ R_3 & R_2 & R_1 & R_0 \end{bmatrix} \begin{bmatrix} a_3 \\ a_2 \\ a_1 \\ a_0 \end{bmatrix} = \begin{bmatrix} b_3 \\ b_2 \\ b_1 \\ b_0 \end{bmatrix}$$

The steps of the Levinson algorithm are explicitly as follows:

Step 0

Solve $R_0 \cdot 1 = E_0$. This defines E_0. Then enlarge to the next size by padding a zero, that is,

$$\begin{bmatrix} R_0 & R_1 \\ R_1 & R_0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} E_0 \\ \Delta_0 \end{bmatrix}, \quad \text{this defines } \Delta_0\text{. Then, also}$$

$$\begin{bmatrix} R_0 & R_1 \\ R_1 & R_0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \Delta_0 \\ E_0 \end{bmatrix}, \quad \text{by reversal invariance}$$

These are the preliminaries to Step 1.

Step 1

We wish to solve

$$\begin{bmatrix} R_0 & R_1 \\ R_1 & R_0 \end{bmatrix} \begin{bmatrix} 1 \\ a_{11} \end{bmatrix} = \begin{bmatrix} E_1 \\ 0 \end{bmatrix} \tag{12.4.3}$$

Try an expression of the form

$$\begin{bmatrix} 1 \\ a_{11} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} - \gamma_1 \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Acting on both sides by $\begin{bmatrix} R_0 & R_1 \\ R_1 & R_0 \end{bmatrix}$ and using the results of Step 0, we obtain

$$\begin{bmatrix} R_0 & R_1 \\ R_1 & R_0 \end{bmatrix} \begin{bmatrix} 1 \\ a_{11} \end{bmatrix} = \begin{bmatrix} R_0 & R_1 \\ R_1 & R_0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} - \gamma_1 \begin{bmatrix} R_0 & R_1 \\ R_1 & R_0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \text{or,}$$

$$\begin{bmatrix} E_1 \\ 0 \end{bmatrix} = \begin{bmatrix} E_0 \\ \Delta_0 \end{bmatrix} - \gamma_1 \begin{bmatrix} \Delta_0 \\ E_0 \end{bmatrix}, \quad \text{or,}$$

$$E_1 = E_0 - \gamma_1 \Delta_0 \,, \quad 0 = \Delta_0 - \gamma_1 E_0 \,, \quad \text{or}$$

$$\gamma_1 = \frac{\Delta_0}{E_0}, \quad E_1 = E_0 - \gamma_1 \Delta_0 = (1 - \gamma_1^2) E_0 \,, \quad \text{where } \; \Delta_0 = R_1$$

These define γ_1 and E_1. As a preliminary to Step 2, enlarge Eq. (12.4.3) to the next size by padding a zero

$$\begin{bmatrix} R_0 & R_1 & R_2 \\ R_1 & R_0 & R_1 \\ R_2 & R_1 & R_0 \end{bmatrix} \begin{bmatrix} 1 \\ a_{11} \\ 0 \end{bmatrix} = \begin{bmatrix} E_1 \\ 0 \\ \Delta_1 \end{bmatrix}, \quad \text{this defines } \Delta_1\text{. Then, also}$$

$$\begin{bmatrix} R_0 & R_1 & R_2 \\ R_1 & R_0 & R_1 \\ R_2 & R_1 & R_0 \end{bmatrix} \begin{bmatrix} 0 \\ a_{11} \\ 1 \end{bmatrix} = \begin{bmatrix} \Delta_1 \\ 0 \\ E_1 \end{bmatrix}, \quad \text{by reversal invariance}$$

Step 2

We wish to solve

$$\begin{bmatrix} R_0 & R_1 & R_2 \\ R_1 & R_0 & R_1 \\ R_2 & R_1 & R_0 \end{bmatrix} \begin{bmatrix} 1 \\ a_{21} \\ a_{22} \end{bmatrix} = \begin{bmatrix} E_2 \\ 0 \\ 0 \end{bmatrix} \tag{12.4.4}$$

Try an expression of the form:

$$\begin{bmatrix} 1 \\ a_{21} \\ a_{22} \end{bmatrix} = \begin{bmatrix} 1 \\ a_{11} \\ 0 \end{bmatrix} - \gamma_2 \begin{bmatrix} 0 \\ a_{11} \\ 1 \end{bmatrix}, \quad \text{with } \gamma_2 \text{ to be determined}$$

Acting on both sides by the 3×3 autocorrelation matrix and using Step 1, we find

$$\begin{bmatrix} E_2 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} E_1 \\ 0 \\ \Delta_1 \end{bmatrix} - \gamma_2 \begin{bmatrix} \Delta_1 \\ 0 \\ E_1 \end{bmatrix}, \quad \text{or,}$$

$$E_2 = E_1 - \gamma_2 \Delta_1 \,, \quad 0 = \Delta_1 - \gamma_2 E_1 \,, \quad \text{or}$$

$$\gamma_2 = \frac{\Delta_1}{E_1}, \quad E_2 = (1 - \gamma_2^2) E_1 \,, \quad \text{where } \; \Delta_1 = [R_2, R_1] \begin{bmatrix} 1 \\ a_{11} \end{bmatrix}$$

These define γ_2 and E_2. As a preliminary to Step 3, enlarge Eq. (12.4.4) to the next size by padding a zero

$$\begin{bmatrix} R_0 & R_1 & R_2 & R_3 \\ R_1 & R_0 & R_1 & R_2 \\ R_2 & R_1 & R_0 & R_1 \\ R_3 & R_2 & R_1 & R_0 \end{bmatrix} \begin{bmatrix} 1 \\ a_{21} \\ a_{22} \\ 0 \end{bmatrix} = \begin{bmatrix} E_2 \\ 0 \\ 0 \\ \Delta_2 \end{bmatrix}, \quad \text{this defines } \Delta_2\text{. Then, also}$$

$$\begin{bmatrix} R_0 & R_1 & R_2 & R_3 \\ R_1 & R_0 & R_1 & R_2 \\ R_2 & R_1 & R_0 & R_1 \\ R_3 & R_2 & R_1 & R_0 \end{bmatrix} \begin{bmatrix} 0 \\ a_{22} \\ a_{21} \\ 1 \end{bmatrix} = \begin{bmatrix} \Delta_2 \\ 0 \\ 0 \\ E_2 \end{bmatrix}, \quad \text{by reversal invariance}$$

Step 3

We wish to solve

$$\begin{bmatrix} R_0 & R_1 & R_2 & R_3 \\ R_1 & R_0 & R_1 & R_2 \\ R_2 & R_1 & R_0 & R_1 \\ R_3 & R_2 & R_1 & R_0 \end{bmatrix} \begin{bmatrix} 1 \\ a_{31} \\ a_{32} \\ a_{33} \end{bmatrix} = \begin{bmatrix} E_3 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{12.4.5}$$

Try an expression of the form:

$$\begin{bmatrix} 1 \\ a_{31} \\ a_{32} \\ a_{33} \end{bmatrix} = \begin{bmatrix} 1 \\ a_{21} \\ a_{22} \\ 0 \end{bmatrix} - \gamma_3 \begin{bmatrix} 0 \\ a_{22} \\ a_{21} \\ 1 \end{bmatrix}, \quad \text{with } \gamma_3 \text{ to be determined}$$

Acting on both sides by the 4×4 autocorrelation matrix and using Step 2, we obtain

$$\begin{bmatrix} E_3\\ 0\\ 0\\ 0\end{bmatrix} = \begin{bmatrix} E_2\\ 0\\ 0\\ \Delta_2\end{bmatrix} - \gamma_3\begin{bmatrix} \Delta_2\\ 0\\ 0\\ E_2\end{bmatrix}, \quad \text{or,}$$

$$E_3 = E_2 - \gamma_3\Delta_2\,,\quad 0 = \Delta_2 - \gamma_3 E_2\,,\quad \text{or}$$

$$\gamma_3 = \frac{\Delta_2}{E_2}\,,\quad E_3 = (1-\gamma_3^2)E_2\,,\quad \text{where}\;\; \Delta_2 = [R_3, R_2, R_1]\begin{bmatrix} 1\\ a_{21}\\ a_{22}\end{bmatrix}$$

Clearly, the procedure can be continued to higher and higher orders, as required in each problem. Note that at each step, we used the order-updating Eqs. (1.8.40) in conjunction with Eq. (1.8.47).

12.5 Autocorrelation Sequence Extensions

In this section, we discuss the problem of extending an autocorrelation function and the related issues of singular autocorrelation matrices. The equivalence between an autocorrelation function and the set of reflection coefficients provides a convenient and systematic way to (a) test whether a given finite set of numbers are the autocorrelation lags of a stationary signal and (b) extend a given finite set of autocorrelation lags to arbitrary lengths while preserving the autocorrelation property.

For a finite set of numbers $\{R(0), R(1), \dots, R(p)\}$ to be the lags of an autocorrelation function, it is necessary and sufficient that all reflection coefficients, extracted from this set via the Levinson recursion, have magnitude less than one; that is, $|\gamma_i| < 1$, for $i = 1, 2, \dots, p$, and also that $R(0) > 0$. These conditions are equivalent to the positive definiteness of the autocorrelation matrix R_p. The proof follows from the fact that the positivity of R_p is equivalent to the conditions on the prediction errors $E_i > 0$, for $i = 1, 2, \dots, p$. In turn, these conditions are equivalent to $E_0 = R(0) > 0$ and and, through Eq. (12.3.13), to the reflection coefficients having magnitude less than one.

The problem of extending a finite set $\{R(0), R(1), \dots, R(p)\}$ of autocorrelation lags is to find a number $R(p+1)$ such that the extended set $\{R(0), R(1), \dots, R(p), R(p+1)\}$ is still an autocorrelation sequence. This can be done by parametrizing $R(p+1)$ in terms of the next reflection coefficient γ_{p+1}. Solving Eq. (12.3.12) for $R(p+1)$ and using Eq. (12.3.11), we obtain

$$R(p+1) = \gamma_{p+1}E_p - \big[a_{p1}R(p) + a_{p2}R(p-1) + \cdots + a_{pp}R(1)\big] \tag{12.5.1}$$

Any number γ_{p+1} in the range $-1 < \gamma_{p+1} < 1$ will give rise to an acceptable value for $R(p+1)$. The choice $\gamma_{p+1} = 0$ is special and corresponds to the so-called autoregressive or *maximum entropy extension* of the autocorrelation function (see Problem 12.16). If this choice is repeated to infinity, we will obtain the set of reflection coefficients

$$\{\gamma_1, \gamma_2, \dots, \gamma_p, 0, 0, \dots\}$$

It follows from the Levinson recursion that all prediction-error filters of order greater than p will remain equal to the pth filter, $A_p(z) = A_{p+1}(z) = A_{p+2}(z) = \cdots$. Therefore, the corresponding whitening filter will be $A(z) = A_p(z)$, that is, an autoregressive model of order p. With the exception of the above autoregressive extension that leads to an all-pole signal model, the extendibility conditions $|\gamma_{p+i}| < 1$, $i \geq 1$, do not necessarily guarantee that the resulting signal model will be a rational (pole-zero) model.

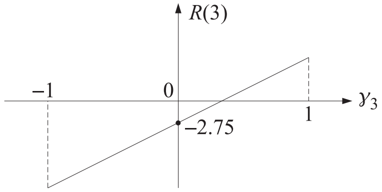
Example 12.5.1: Consider the three numbers $\{R(0), R(1), R(2)\} = \{8, 4, -1\}$. The Levinson recursion gives $\{\gamma_1, \gamma_2\} = \{0.5, -0.5\}$ and $\{E_1, E_2\} = \{6, 4.5\}$. Thus, the above numbers qualify to be autocorrelation lags. The corresponding prediction-error filters are

$$\mathbf{a}_1 = \begin{bmatrix} 1\\ a_{11}\end{bmatrix} = \begin{bmatrix} 1\\ -0.5\end{bmatrix},\quad \mathbf{a}_2 = \begin{bmatrix} 1\\ a_{21}\\ a_{22}\end{bmatrix} = \begin{bmatrix} 1\\ -0.75\\ 0.5\end{bmatrix}$$

The next lag in this sequence can be chosen according to Eq. (12.5.1)

$$R(3) = \gamma_3 E_2 - \big[a_{21}R(2) + a_{22}R(1)\big] = 4.5\gamma_3 - 2.75$$

where γ_3 is any number in the interval $-1 < \gamma_3 < 1$. The resulting possible values of $R(3)$ are plotted below versus γ_3 . In particular, the autoregressive extension corresponds to $\gamma_3 = 0$, which gives $R(3) = -2.75$. □



The end-points, $\gamma_{p+1} = \pm 1$, of the allowed interval $(-1, 1)$ correspond to the two possible extreme values of $R(p+1)$:

$$R(p+1) = \pm E_p - \big[a_{p1}R(p) + a_{p2}R(p-1) + \cdots + a_{pp}R(1)\big]$$

In this case, the corresponding prediction error vanishes $E_{p+1} = (1 - \gamma_{p+1}^2)E_p = 0$. This makes the resulting order-$(p+1)$ autocorrelation matrix R_{p+1} singular. The prediction filter becomes either the symmetric (if $\gamma_{p+1} = -1$) or antisymmetric (if $\gamma_{p+1} = 1$) combination

$$\mathbf{a}_{p+1} = \begin{bmatrix} \mathbf{a}_p\\ 0\end{bmatrix} + \begin{bmatrix} 0\\ \mathbf{a}_p^R\end{bmatrix},\quad A_{p+1}(z) = A_p(z) + z^{-1}A_p^R(z)\,,\quad \text{or,}$$

$$\mathbf{a}_{p+1} = \begin{bmatrix} \mathbf{a}_p\\ 0\end{bmatrix} - \begin{bmatrix} 0\\ \mathbf{a}_p^R\end{bmatrix},\quad A_{p+1}(z) = A_p(z) - z^{-1}A_p^R(z)$$

In either case, it can be shown that the zeros of the polynomial $A_{p+1}(z)$ lie on the unit circle, and that the prediction filter $\mathbf{a}_{p+1}$ becomes an eigenvector of R_{p+1} with zero eigenvalue; namely, $R_{p+1}\mathbf{a}_{p+1} = 0$. This follows from the normal Eqs. (12.3.28) $R_{p+1}\mathbf{a}_{p+1} = E_{p+1}\mathbf{u}_{p+1}$ and $E_{p+1} = 0$.

Example 12.5.2: Consider the extended autocorrelation sequence of Example 12.5.1 defined by the singular choice $\gamma_3 = -1$. Then, $R(3) = -4.5 - 2.75 = -7.25$. The corresponding order-3 prediction-error filter is computed using the order-2 predictor and the Levinson recursion

$$\mathbf{a}_3 = \begin{bmatrix} 1 \\ a_{31} \\ a_{32} \\ a_{33} \end{bmatrix} = \begin{bmatrix} 1 \\ -0.75 \\ 0.5 \\ 0 \end{bmatrix} - \gamma_3 \begin{bmatrix} 0 \\ 0.5 \\ -0.75 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ -0.25 \\ -0.25 \\ 1 \end{bmatrix}$$

It is symmetric about its middle. Its zeros, computed as the solutions of $(1 - 0.25z^{-1} - 0.25z^{-2} + z^{-3}) = (1 + z^{-1})(1 - 1.25z^{-1} + z^{-2}) = 0$ are

$$z = -1\,, \quad z = \frac{5 \pm j\sqrt{39}}{8}$$

and lie on the unit circle. Finally, we verify that $\mathbf{a}_3$ is an eigenvector of R_3 with zero eigenvalue:

$$R_3\mathbf{a}_3 = \begin{bmatrix} 8 & 4 & -1 & -7.25 \\ 4 & 8 & 4 & -1 \\ -1 & 4 & 8 & 4 \\ -7.25 & -1 & 4 & 8 \end{bmatrix} \begin{bmatrix} 1 \\ -0.25 \\ -0.25 \\ 1 \end{bmatrix} = 0$$

□

Singular autocorrelation matrices, and the associated symmetric or antisymmetric prediction filters with zeros on the unit circle, find application in the method of *line spectrum pairs* (LSP) of speech analysis [937]. They are also intimately related to the eigenvector methods of spectrum estimation, such as Pisarenko's method of harmonic retrieval, discussed in Sec. 14.2. This connection arises from the property that singular autocorrelation matrices (with nonsingular principal minors) admit a *representation* as a sum of sinusoidal components [938], the frequencies of which are given precisely by the zeros, on the unit circle, of the corresponding prediction filter. This sinusoidal representation is equivalent to the eigen-decomposition of the matrix. The prediction filter can, alternatively, be computed as the eigenvector belonging to zero eigenvalue. The proof of these results can be derived as a limiting case; namely, the noise-free case, of the more general eigenvector methods that deal with sinusoids in noise. A direct proof is suggested in Problem 14.10.

Example 12.5.3: Consider the autocorrelation matrix $R = \begin{bmatrix} 2 & 1 & -1 \\ 1 & 2 & 1 \\ -1 & 1 & 2 \end{bmatrix}$. It is easily verified that the corresponding autocorrelation lags $R(k)$ admit the sinusoidal representation

$$R(k) = 2\cos(\omega_1 k) = e^{j\omega_1 k} + e^{-j\omega_1 k}\,, \quad \text{for } k = 0, 1, 2$$

where $\omega_1 = \pi/3$. Sending these lags through the Levinson recursion, we find $\{\gamma_1, \gamma_2\} = \{0.5, -1\}$ and $\{E_1, E_2\} = \{1.5, 0\}$. Thus, R singular. Its eigenvalues are $\{0, 3, 3\}$. The corresponding prediction filters are $\mathbf{a}_1 = [1, -0.5]^T$ and $\mathbf{a}_2 = [1, -1, 1]^T$. It is easily verified that $\mathbf{a}_2$ is an eigenvector of R with zero eigenvalue, i.e., $R\mathbf{a}_2 = 0$. The corresponding eigenfilter $A_2(z) = 1 - z^{-1} + z^{-2}$, is symmetric about its middle and has zeros on the unit circle coinciding with the sinusoids present in R, namely, $z = e^{\pm j\omega_1}$. The other two eigenvectors of R are

$$\mathbf{c} = \begin{bmatrix} 1 \\ \cos\omega_1 \\ \cos 2\omega_1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0.5 \\ -0.5 \end{bmatrix}, \quad \mathbf{d} = \begin{bmatrix} 0 \\ \sin\omega_1 \\ \sin 2\omega_1 \end{bmatrix} = \begin{bmatrix} 0 \\ \sqrt{3}/2 \\ \sqrt{3}/2 \end{bmatrix}$$

both belonging to eigenvalue $\lambda = 3$. Their norm is $\|\mathbf{c}\| = \|\mathbf{d}\| = \sqrt{3}/2$. The three eigenvectors $\mathbf{a}_2, \mathbf{c}, \mathbf{d}$ are mutually orthogonal. It is easily verified that the matrix R may be represented in the form $R = 2\mathbf{c}\mathbf{c}^T + 2\mathbf{d}\mathbf{d}^T$, which, after normalizing $\mathbf{c}$ and $\mathbf{d}$ to unit norm, is recognized as the eigendecomposition of R, We can also express R in terms of its complex sinusoidal components in the form $R = \mathbf{s}\mathbf{s}^\dagger + \mathbf{s}^*\mathbf{s}^T$, where

$$\mathbf{s} = \mathbf{c} + j\mathbf{d} = \begin{bmatrix} 1 \\ e^{j\omega_1} \\ e^{2j\omega_1} \end{bmatrix}, \quad \mathbf{s}^\dagger = \mathbf{s}^{*T} = [1, e^{-j\omega_1}, e^{-2j\omega_1}]$$

Example 12.5.4: Similarly, one can verify that the four autocorrelation lags $\{8, 4, -1, -7.25\}$ of the singular matrix of Example 12.5.2 can be represented in the sinusoidal form

$$R(k) = P_1 e^{j\omega_1 k} + P_2 e^{j\omega_2 k} + P_3 e^{j\omega_3 k}\,, \quad \text{for } k = 0, 1, 2, 3$$

where $P_1 = 8/13$, $P_2 = P_3 = 96/13$, and ω_i correspond to the zeros of the prediction filter $\mathbf{a}_3$, namely,

$$e^{j\omega_1} = -1\,, \quad e^{j\omega_2} = \frac{5 + j\sqrt{39}}{8}\,, \quad e^{j\omega_3} = \frac{5 - j\sqrt{39}}{8}\,, \quad \text{so that, } \omega_3 = -\omega_2$$

The matrix itself has the sinusoidal representation

$$R = P_1\mathbf{s}_1\mathbf{s}_1^\dagger + P_2\mathbf{s}_2\mathbf{s}_2^\dagger + P_3\mathbf{s}_3\mathbf{s}_3^\dagger\,, \quad \text{where } \mathbf{s}_i = \begin{bmatrix} 1 \\ e^{j\omega_i} \\ e^{2j\omega_i} \\ e^{3j\omega_i} \end{bmatrix}$$

Had we chosen the value $\gamma_3 = 1$ in Example 12.5.2, we would have found the extended lag $R(3) = 1.75$ and the antisymmetric order-3 prediction-error filter $\mathbf{a}_3 = [1, -1.25, 1.25, -1]^T$, whose zeros are on the unit circle:

$$e^{j\omega_1} = 1\,, \quad e^{j\omega_2} = \frac{1 + j\sqrt{63}}{8}\,, \quad e^{j\omega_3} = \frac{1 - j\sqrt{63}}{8}$$

with $R(k)$ admitting the sinusoidal representation

$$R(k) = P_1 + 2P_2\cos(\omega_2 k) = [8, 4, -1, 1.75]\,, \quad \text{for } k = 0, 1, 2, 3$$

where $P_1 = 24/7$ and $P_2 = 16/7$. □

12.6 Split Levinson Algorithm

The main computational burden of Levinson's algorithm is $2p$ multiplications per stage, arising from the p multiplications in Eq. (12.3.15) and in the computation of the inner product (12.3.12). Thus, for M stages, the algorithm requires

$$2\sum_{p=1}^{M} p = M(M+1)$$

or, $O(M^2)$ multiplications. This represents a factor of M savings over solving the normal equations (12.3.7) by direct matrix inversion, requiring $O(M^3)$ operations. The savings can be substantial considering that in speech processing M = 10–15, and in seismic processing M = 100–200. Progress in VLSI hardware has motivated the development of efficient parallel implementations of Levinson's algorithm and its variants [939–958]. With M parallel processors, the complexity of the algorithm is typically reduced by another factor of M to $O(M)$ or $O(M\log M)$ operations.

An interesting recent development is the realization that Levinson's algorithm has some inherent redundancy, which can be exploited to derive more efficient versions of the algorithm allowing an additional 50% reduction in computational complexity. These versions were motivated by a new stability test for linear prediction polynomials by Bistritz [959], and have been termed *Split Levinson* or *Immitance-Domain Levinson algorithms* [960–967]. They are based on efficient *three-term recurrence* relations for the symmetrized or antisymmetrized prediction polynomials. Following [960], we define the order-p symmetric polynomial

$$F_p(z)= A_{p-1}(z)+z^{-1}A^R_{p-1}(z)\,,\quad \mathbf{f}_p = \begin{bmatrix}\mathbf{a}_{p-1}\\ 0\end{bmatrix}+\begin{bmatrix}0\\ \mathbf{a}^R_{p-1}\end{bmatrix} \tag{12.6.1}$$

The coefficient vector $\mathbf{f}_p$ is symmetric about its middle; that is, $f_{p0} = f_{pp} = 1$ and $f_{pi} = a_{p-1,i} + a_{p-1,p-i} = f_{p,p-i}$, for $i = 1,2,\dots,p-1$. Thus, only half of the vector $\mathbf{f}_p$, is needed to specify it completely. Using the backward recursion (12.3.22) to write $A_{p-1}(z)$ in terms of $A_p(z)$, we obtain the alternative expression

$$F_p = \frac{1}{1-\gamma_p^2}\big[(A_p+\gamma_p A_p^R)+z^{-1}(\gamma_p zA_p + zA_p^R)\big] = \frac{1}{1-\gamma_p}[A_p + A_p^R]\,,\quad \text{or,}$$

$$(1-\gamma_p)F_p(z)= A_p(z)+A_p^R(z)\,,\quad (1-\gamma_p)\mathbf{f}_p = \mathbf{a}_p + \mathbf{a}_p^R \tag{12.6.2}$$

The polynomial $A_p(z)$ and its reverse may be recovered from the knowledge of the symmetric polynomials $F_p(z)$. Writing Eq. (12.6.1) for order $p+1$, we obtain $F_{p+1}(z)= A_p(z)+z^{-1}A_p^R(z)$. This equation, together with Eq. (12.6.2), may be solved for $A_p(z)$ and $A_p^R(z)$, yielding

$$A_p(z)=\frac{F_{p+1}(z)-(1-\gamma_p)z^{-1}F_p(z)}{1-z^{-1}}\,,\quad A_p^R(z)=\frac{(1-\gamma_p)F_p(z)-F_{p+1}(z)}{1-z^{-1}} \tag{12.6.3}$$

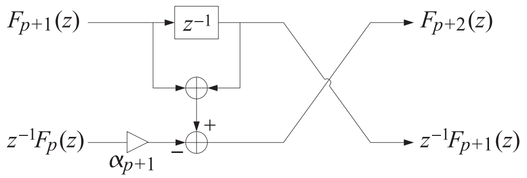
Inserting these expressions into the forward Levinson recursion (12.3.16) and canceling the common factor $1/(1-z^{-1})$, we obtain a three-term recurrence relation for $F_p(z)$:

$$F_{p+2}-(1-\gamma_{p+1})z^{-1}F_{p+1} = \big[F_{p+1}-(1-\gamma_p)z^{-1}F_p\big]-\gamma_{p+1}z^{-1}\big[(1-\gamma_p)F_p - F_{p+1}\big]$$

or,

$$F_{p+2}(z)= (1+z^{-1})F_{p+1}(z)-\alpha_{p+1}z^{-1}F_p(z) \tag{12.6.4}$$

where $\alpha_{p+1} = (1+\gamma_{p+1})(1-\gamma_p)$. In block diagram form



Because $F_p(z)$ has order p and is delayed by z^{-1}, the coefficient form of (12.6.4) is

$$\mathbf{f}_{p+2} = \begin{bmatrix}\mathbf{f}_{p+1}\\ 0\end{bmatrix}+\begin{bmatrix}0\\ \mathbf{f}_{p+1}\end{bmatrix}-\alpha_{p+1}\begin{bmatrix}0\\ \mathbf{f}_p\\ 0\end{bmatrix} \tag{12.6.5}$$

The recursion is initialized by $F_0(z)= 2$ and $F_1(z)= 1+z^{-1}$. Because of the symmetric nature of the polynomial $F_p(z)$ only half of its coefficients need be updated by Eqs. (12.6.4) or (12.6.5). To complete the recursion, we need an efficient way to update the coefficients α_{p+1}. Taking the dot product of both sides of Eq. (12.6.2) with the row vector $[R(0),R(1),\dots,R(p)]$, we obtain

$$[R(0),\dots,R(p)]\mathbf{a}_p + [R(0),\dots,R(p)]\mathbf{a}_p^R = (1-\gamma_p)[R(0),\dots,R(p)]\mathbf{f}_p$$

The first term is recognized as the gapped function $g_p(0)= E_p$, and the second term as $g_p(p)= 0$. Dividing by $1-\gamma_p$ and denoting $\tau_p = E_p/(1-\gamma_p)$, we obtain

$$\tau_p = [R(0),R(1),\dots,R(p)]\mathbf{f}_p = \sum_{i=0}^{p} R(i)f_{pi} \tag{12.6.6}$$

Because of the symmetric nature of $\mathbf{f}_p$ the quantity τ_p can be computed using only half of the terms in the above inner product. For example, if p is odd, the above sum may be folded to half its terms

$$\tau_p = \sum_{i=0}^{(p-1)/2}[R(i)+R(p-i)]f_{pi}$$

Because Eqs. (12.6.5) and (12.6.6) can be folded in half, the total number of multiplications per stage will be $2(p/2)= p$, as compared with $2p$ for the classical Levinson algorithm. This is how the 50% reduction in computational complexity arises. The recursion is completed by noting that α_{p+1} can be computed in terms of τ_p by

$$\alpha_{p+1} = \frac{\tau_{p+1}}{\tau_p} \tag{12.6.7}$$

This follows from Eq. (12.3.13),

$$\frac{\tau_{p+1}}{\tau_p} = \frac{E_{p+1}}{1-\gamma_{p+1}}\,\frac{1-\gamma_p}{E_p} = \frac{1-\gamma_{p+1}^2}{1-\gamma_{p+1}}(1-\gamma_p)= (1+\gamma_{p+1})(1-\gamma_p)= \alpha_{p+1}$$

A summary of the algorithm, which also includes a recursive computation of the reflection coefficients, is as follows:

1. Initialize with $\tau_0 = E_0 = R(0)$, $\gamma_0 = 0$, $\mathbf{f}_0 = [2]$, $\mathbf{f}_1 = [1,1]^T$.
2. At stage p, the quantities $\tau_p, \gamma_p, \mathbf{f}_p, \mathbf{f}_{p+1}$ are available.
3. Compute τ_{p+1} from Eq. (12.6.6), using only half the terms in the sum.
4. Compute α_{p+1} from Eq. (12.6.7), and solve for $\gamma_{p+1} = -1 + \alpha_{p+1}/(1-\gamma_p)$.
5. Compute $\mathbf{f}_{p+2}$ from Eq. (12.6.5), using half of the coefficients.
6. Go to stage $p+1$.

After the final desired order is reached, the linear prediction polynomial can be recovered from Eq. (12.6.3), which can be written recursively as

$$a_{pi} = a_{p,i-1} + f_{p+1.i} - (1-\gamma_p)f_{p,i-1}\,, \quad i = 1,2,\dots,p \tag{12.6.8}$$

with $a_{p0} = 1$, or vectorially,

$$\begin{bmatrix} \mathbf{a}_p \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{a}_p \end{bmatrix} + \mathbf{f}_{p+1} - (1-\gamma_p)\begin{bmatrix} 0 \\ \mathbf{f}_p \end{bmatrix} \tag{12.6.9}$$

Using the three-term recurrence (12.6.5), we may replace $\mathbf{f}_{p+1}$ in terms of $\mathbf{f}_p$ and $\mathbf{f}_{p-1}$, and rewrite Eq. (12.6.9) as

$$\begin{bmatrix} \mathbf{a}_p \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{a}_p \end{bmatrix} + \begin{bmatrix} \mathbf{f}_p \\ 0 \end{bmatrix} + \gamma_p \begin{bmatrix} 0 \\ \mathbf{f}_p \end{bmatrix} - \alpha_p \begin{bmatrix} 0 \\ \mathbf{f}_{p-1} \\ 0 \end{bmatrix} \tag{12.6.10}$$

and in the z-domain

$$A_p(z) = z^{-1}A_p(z) + (1+\gamma_p z^{-1})F_p(z) - \alpha_p z^{-1}F_{p-1}(z) \tag{12.6.11}$$

Example 12.6.1: We rederive the results of Example 12.3.1 using this algorithm, showing explicitly the computational savings. Initialize with $\tau_0 = R(0) = 128$, $\mathbf{f}_0 = [2]$, $\mathbf{f}_1 = [1,1]^T$. Using Eq. (12.6.6), we compute

$$\tau_1 = [R(0), R(1)]\mathbf{f}_1 = [R(0)+R(1)]f_{10} = 128 - 64 = 64$$

Thus, $\alpha_1 = \tau_1/\tau_0 = 64/128 = 0.5$ and $\gamma_1 = -1 + \alpha_1 = -0.5$. Using Eq. (12.6.5) we find

$$\mathbf{f}_2 = \begin{bmatrix} \mathbf{f}_1 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ \mathbf{f}_1 \end{bmatrix} - \alpha_1 \begin{bmatrix} 0 \\ \mathbf{f}_0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} - 0.5 \begin{bmatrix} 0 \\ 2 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

and compute τ_2

$$\tau_2 = [R(0), R(1), R(2)]\mathbf{f}_2 = [R(0)+R(2)]f_{20} + R(1)f_{21} = 144$$

Thus, $\alpha_2 = \tau_2/\tau_1 = 144/64 = 2.25$ and $\gamma_2 = -1 + \alpha_2/(1-\gamma_1) = -1 + 2.25/1.5 = 0.5$. Next, compute $\mathbf{f}_3$ and τ_3

$$\mathbf{f}_3 = \begin{bmatrix} \mathbf{f}_2 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ \mathbf{f}_2 \end{bmatrix} - \alpha_2 \begin{bmatrix} 0 \\ \mathbf{f}_1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix} - 2.25 \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ -0.25 \\ -0.25 \\ 1 \end{bmatrix}$$

$$\tau_3 = [R(0), R(1), R(2), R(3)]\mathbf{f}_3 = [R(0)+R(3)]f_{30} + [R(1)+R(2)]f_{31} = 36$$

which gives $\alpha_3 = \tau_3/\tau_2 = 36/144 = 0.25$ and $\gamma_3 = -1 + \alpha_3/(1-\gamma_2) = -0.5$. Next, we compute $\mathbf{f}_4$ and τ_4

$$\mathbf{f}_4 = \begin{bmatrix} \mathbf{f}_3 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ \mathbf{f}_3 \end{bmatrix} - \alpha_3 \begin{bmatrix} 0 \\ \mathbf{f}_2 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ -0.25 \\ -0.25 \\ 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ -0.25 \\ -0.25 \\ 1 \end{bmatrix} - 0.25 \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0.5 \\ -0.75 \\ 0.5 \\ 1 \end{bmatrix}$$

$$\begin{aligned} \tau_4 &= [R(0), R(1), R(2), R(3), R(4)]\mathbf{f}_4 \\ &= [R(0)+R(4)]f_{40} + [R(1)+R(3)]f_{41} + R(2)f_{42} = 81 \end{aligned}$$

which gives $\alpha_4 = \tau_4/\tau_3 = 81/36 = 2.25$ and $\gamma_4 = -1 + \alpha_4/(1-\gamma_3) = 0.5$. The final prediction filter $\mathbf{a}_4$ can be computed using Eq. (12.6.9) or (12.6.10). To avoid computing $\mathbf{f}_5$ we use Eq. (12.6.10), which gives

$$\begin{bmatrix} 1 \\ a_{41} \\ a_{42} \\ a_{43} \\ a_{44} \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ a_{41} \\ a_{42} \\ a_{43} \\ a_{44} \end{bmatrix} + \begin{bmatrix} 1 \\ 0.5 \\ -0.75 \\ 0.5 \\ 1 \\ 0 \end{bmatrix} + 0.5 \begin{bmatrix} 0 \\ 1 \\ 0.5 \\ -0.75 \\ 0.5 \\ 1 \end{bmatrix} - 2.25 \begin{bmatrix} 0 \\ 1 \\ -0.25 \\ -0.25 \\ 1 \\ 0 \end{bmatrix}$$

with solution $\mathbf{a}_4 = [1, -0.25, -0.1875, 0.5, -0.5]^T$. □

12.7 Analysis and Synthesis Lattice Filters

The Levinson recursion, expressed in the 2×2 matrix form of Eq. (12.3.18) forms the basis of the so-called lattice, or ladder, realizations of the prediction-error filters and their inverses [917]. Remembering that the prediction-error sequence $e_p(n)$ is the convolution of the prediction-error filter $[1, a_{p1}, a_{p2}, \dots, a_{pp}]$ with the original data sequence y_n, that is,

$$e_p^+(n) = y_n + a_{p1}y_{n-1} + a_{p2}y_{n-2} + \cdots + a_{pp}y_{n-p} \tag{12.7.1}$$

we find in the z-domain

$$E_p^+(z) = A_p(z)Y(z) \tag{12.7.2}$$

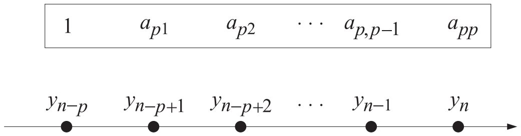
where we changed the notation slightly and denoted $e_p(n)$ by $e_p^+(n)$. At this point, it proves convenient to introduce the *backward* prediction-error sequence, defined in terms of the *reverse* of the prediction-error filter, as follows:

$$\begin{aligned} E_p^-(z) &= A_p^R(z)Y(z) \\ e_p^-(n) &= y_{n-p} + a_{p1}y_{n-p+1} + a_{p2}y_{n-p+2} + \cdots + a_{pp}y_n \end{aligned} \tag{12.7.3}$$

where $A_p^R(z)$ is the reverse of $A_p(z)$, namely,

$$A_p^R(z) = z^{-p}A_p(z^{-1}) = a_{pp} + a_{p,p-1}z^{-1} + a_{p,p-2}z^{-2} + \cdots + a_{p1}z^{-(p-1)} + z^{-p}$$

The signal sequence $e_p^-(n)$ may be interpreted as the *postdiction* error in postdicting the value of y_{n-p} on the basis of the *future* p samples $\{y_{n-p+1}, y_{n-p+2}, \dots, y_{n-1}, y_n\}$, as shown below



Actually, the above choice of postdiction coefficients is the optimal one that minimizes the mean-square postdiction error

$$E[e_p^-(n)^2] = \min \tag{12.7.4}$$

This is easily shown by inserting Eq. (12.7.3) into (12.7.4) and using stationarity

$$\begin{aligned} E[e_p^-(n)^2] &= E\left[\left(\sum_{m=0}^{p} a_{pm} y_{n-p+m}\right)^2\right] = \sum_{m,k=0}^{p} a_{pm} E[y_{n-p+m} y_{n-p+k}] a_{pk} \\ &= \sum_{m,k=0}^{p} a_{pm} R(m-k) a_{pk} = E[e_p^+(n)^2] \end{aligned}$$

which shows that the forward and the backward prediction error criteria are the same, thus, having the *same* solution for the optimal coefficients. We can write Eqs. (12.7.1) and (12.7.3) vectorially

$$e_p^+(n) = [1, a_{p1}, \dots, a_{pp}] \begin{bmatrix} y_n \\ y_{n-1} \\ \vdots \\ y_{n-p} \end{bmatrix} = \mathbf{a}_p^T \mathbf{y}_p(n) \tag{12.7.5a}$$

$$e_p^-(n) = [a_{pp}, a_{p,p-1}, \dots, 1] \begin{bmatrix} y_n \\ y_{n-1} \\ \vdots \\ y_{n-p} \end{bmatrix} = \mathbf{a}_p^{RT} \mathbf{y}_p(n) = \mathbf{b}_p^T \mathbf{y}_p(n) \tag{12.7.5b}$$

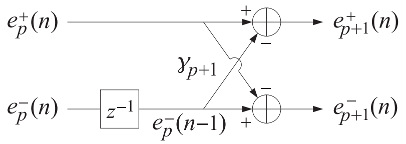
They are recognized as the forward and backward prediction errors e_a and e_b of Eq. (1.8.9). Multiplying both sides of the Levinson recursion (12.3.18) by $Y(z)$, we cast it in the equivalent form in terms of the forward and backward prediction-error sequences:

$$\begin{bmatrix} E_{p+1}^+(z) \\ E_{p+1}^-(z) \end{bmatrix} = \begin{bmatrix} 1 & -\gamma_{p+1} z^{-1} \\ -\gamma_{p+1} & z^{-1} \end{bmatrix} \begin{bmatrix} E_p^+(z) \\ E_p^-(z) \end{bmatrix} \tag{12.7.6}$$

and in the time domain

$$\begin{aligned} e_{p+1}^+(n) &= e_p^+(n) - \gamma_{p+1} e_p^-(n-1) \\ e_{p+1}^-(n) &= e_p^-(n-1) - \gamma_{p+1} e_p^+(n) \end{aligned} \tag{12.7.7}$$

and in block diagram form



These recursions are initialized at $p = 0$ by

$$E_0^\pm(z) = A_0(z) Y(z) = Y(z) \quad \text{and} \quad e_0^\pm(n) = y_n \tag{12.7.8}$$

Eqs. (12.7.7) are identical to (1.8.50), with the identifications $e_a \to e_{p+1}^+(n)$, $\bar{e}_a \to e_p^+(n)$, $e_b \to e_{p+1}^-(n)$, $\bar{e}_b \to e_p^-(n-1)$ the last following from Eq. (12.3.26).

The lattice realization of the prediction-error filter is based on the recursion (12.7.7). Starting at $p = 0$, the output of the pth stage of (12.7.7) becomes the input of the $(p+1)$th stage, up to the final desired order $p = M$. This is depicted in Fig. 12.7.1.

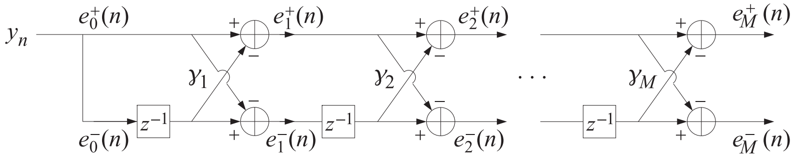
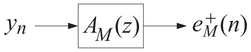


Fig. 12.7.1 Analysis lattice filter.

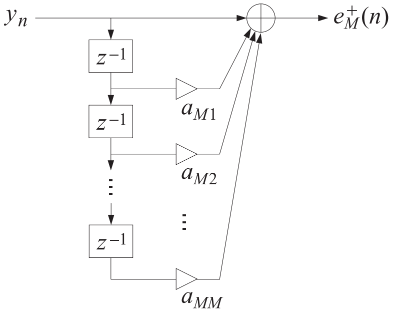
At each time instant n the numbers held in the M delay registers of the lattice can be taken as the internal state of the lattice. The function **lattice** is an implementation of Fig. 12.7.1. At each instant n, the function takes two overall inputs $e_0^\pm(n)$, makes M calls to the function **section** that implements the single lattice section (12.7.7), produces the two overall outputs $e_M^\pm(n)$, and updates the internal state of the lattice in preparation for the next call. By allowing the reflection coefficients to change between calls, the function can also be used in adaptive lattice filters.

Eqs. (12.7.3) imply that the transfer function from the input y_n to the output $e_M^+(n)$ is the desired prediction-error filter $A_M(z)$, whereas the transfer function from y_n to $e_M^-(n)$ is the reversed filter $A_M^R(z)$. The lattice realization is therefore equivalent to the direct-form realization

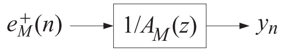
$$e_M^+(n) = y_n + a_{M1} y_{n-1} + a_{M2} y_{n-2} + \cdots a_{MM} y_{n-M}$$



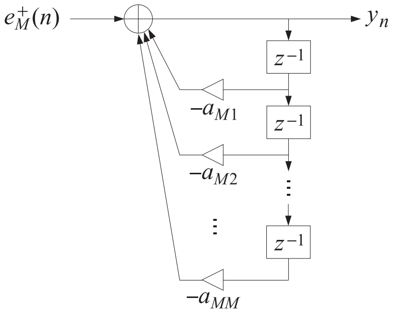
realized directly in terms of the prediction coefficients. It is depicted below



The synthesis filter $1/A_M(z)$ can also be realized in a lattice form. The input to the synthesis filter is the prediction error sequence $e_M^+(n)$ and its output is the original sequence y_n:



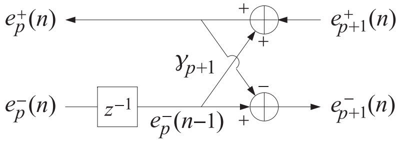
Its direct-form realization is:



For the lattice realization, since y_n corresponds to $e_0^+(n)$, we must write Eq. (12.7.7) in an order-decreasing form, starting at $e_M^+(n)$ and ending with $e_0^+(n)=y_n$. Rearranging the terms of the first of Eq. (12.7.7), we have

$$\begin{aligned} e_p^+(n) &= e_{p+1}^+(n)+\gamma_{p+1}e_p^-(n-1) \\ e_{p+1}^-(n) &= e_p^-(n-1)-\gamma_{p+1}e_p^+(n) \end{aligned} \tag{12.7.9}$$

which can be realized as shown below:



Note the difference in signs in the upper and lower adders. Putting together the stages from $p=M$ to $p=0$, we obtain the synthesis lattice filter shown in Fig. 12.7.2.

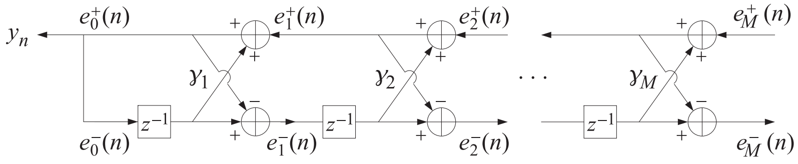


Fig. 12.7.2 Synthesis lattice filter.

Lattice structures based on the split Levinson algorithm can also be developed [960,961]. They are obtained by cascading the block diagram realizations of Eq. (12.6.4) for different values of α_p. The output signals from each section are defined by

$$e_p(n)=\sum_{i=0}^{p}f_{pi}y_{n-i}\,,\quad E_p(z)=F_p(z)Y(z)$$

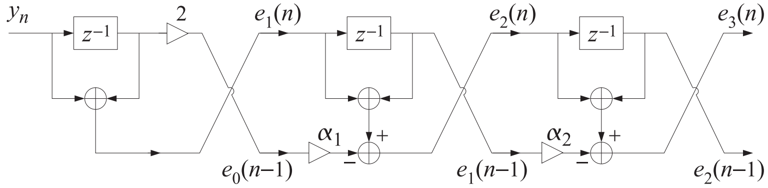
Multiplying both sides of Eq. (12.6.1) by $Y(z)$ we obtain the time-domain expression

$$e_p(n)=e_{p-1}^+(n)+e_{p-1}^-(n-1)$$

Similarly, multiplying both sides of Eq. (12.6.4) by $Y(z)$ we obtain the recursions

$$e_{p+2}(n)=e_{p+1}(n)+e_{p+1}(n-1)-\alpha_p e_p(n-1)$$

They are initialized by $e_0(n)=2y_n$ and $e_1(n)=y_n+y_{n-1}$. Putting together the various sections we obtain the lattice-type realization



The forward prediction error may be recovered from Eq. (12.6.3) or Eq. (12.6.11) by multiplying both sides with $Y(z)$; for example, using Eq. (12.6.11) we find

$$e_p^+(n)=e_{p-1}^+(n)+e_p(n)+\gamma_p e_p(n)-\alpha_p e_{p-1}(n-1)$$

12.8 Alternative Proof of the Minimum-Phase Property

The synthesis filter $1/A_M(z)$ must be stable and causal, which requires all the M zeros of the prediction-error filter $A_M(z)$ to lie inside the unit circle on the complex z-plane. We have already presented a proof of this fact which was based on the property that the coefficients of $A_M(z)$ minimized the mean-squared prediction error $E[e_M^+(n)^2]$. Here, we present an alternative proof based on the Levinson recursion and the fact that all reflection coefficients γ_p have magnitude less than one [920]. From the definition (12.7.3), it follows that

$$e_p^-(n-1) = y_{n-p-1} + a_{p1}y_{n-p} + a_{p2}y_{n-p+1} + \cdots + a_{pp}y_{n-1} \tag{12.8.1}$$

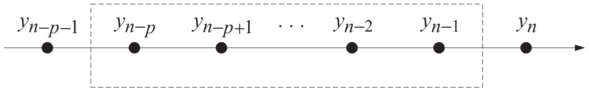
This quantity represents the estimation error of postdicting y_{n-p-1} on the basis of the p future samples $\{y_{n-p}, y_{n-p+1}, \dots, y_{n-1}\}$. Another way to say this is that the linear combination of these p samples is the projection of y_{n-p-1} onto the subspace of random variables spanned by $\{y_{n-p}, y_{n-p+1}, \dots, y_{n-1}\}$; that is,

$$e_p^-(n-1) = y_{n-p-1} - (\text{projection of } y_{n-p-1} \text{ onto } \{y_{n-p}, y_{n-p+1}, \dots, y_{n-1}\}) \tag{12.8.2}$$

On the other hand, $e_p^+(n)$ given in Eq. (12.7.1) is the estimation error of y_n based on the same set of samples. Therefore,

$$e_p^+(n) = y_n - (\text{projection of } y_n \text{ onto } \{y_{n-p}, y_{n-p+1}, \dots, y_{n-1}\}) \tag{12.8.3}$$

The samples $\{y_{n-p}, y_{n-p+1}, \dots, y_{n-1}\}$ are the *intermediate* set of samples between y_{n-p-1} and y_n as shown below:



Therefore, according to the discussion in Sec. 1.7, the PARCOR coefficient between y_{n-p-1} and y_n with the effect of intermediate samples removed is given by

$$\text{PARCOR} = \frac{E[e_p^+(n)e_p^-(n-1)]}{E[e_p^-(n-1)^2]}$$

This is precisely the reflection coefficient γ_{p+1} of Eq. (12.3.11). Indeed, using Eq. (12.8.1) and the gap conditions, $g_p(k) = 0$, $k = 1, 2, \dots, p$, we find

$$\begin{aligned} E[e_p^+(n)e_p^-(n-1)] &= E[e_p^+(n)(y_{n-p-1} + a_{p1}y_{n-p} + a_{p2}y_{n-p+1} + \cdots + a_{pp}y_{n-1})] \\ &= g_p(p+1) + a_{p1}g_p(p) + a_{p2}g_p(p-1) + \cdots a_{pp}g_p(1) \\ &= g_p(p+1) \end{aligned}$$

Similarly, invoking stationarity and Eq. (12.7.4),

$$E[e_p^-(n-1)^2] = E[e_p^-(n)^2] = E[e_p^+(n)^2] = g_p(0)$$

Thus, the reflection coefficient γ_{p+1} is really a PARCOR coefficient:

$$\gamma_{p+1} = \frac{E[e_p^+(n)e_p^-(n-1)]}{E[e_p^-(n-1)^2]} = \frac{E[e_p^+(n)e_p^-(n-1)]}{\sqrt{E[e_p^-(n-1)^2]\,E[e_p^+(n)^2]}} \tag{12.8.4}$$

Using the Schwarz inequality with respect to the inner product $E[uv]$, that is,

$$\big|E[uv]\big|^2 \le E[u^2]\,E[v^2]$$

then Eq. (12.8.4) implies that γ_{p+1} will have magnitude less than one:

$$|\gamma_{p+1}| \le 1\,, \quad \text{for each } p = 0, 1, \dots \tag{12.8.5}$$

To prove the minimum-phase property of $A_M(z)$ we must show that all of its M zeros are inside the unit circle. We will do this by induction. Let Z_p and N_p denote the number of zeros and poles of $A_p(z)$ that lie inside the unit circle. Levinson's recursion, Eq. (12.3.13), expresses $A_{p+1}(z)$ as the sum of $A_p(z)$ and a correction term $F(z) = -\gamma_{p+1}z^{-1}A_p^R(z)$, that is,

$$A_{p+1}(z) = A_p(z) + F(z)$$

Using the inequality (12.8.5) and the fact that $A_p(z)$ has the same magnitude spectrum as $A_p^R(z)$, we find the inequality

$$\big|F(z)\big| = \big|-\gamma_{p+1}z^{-1}A_p^R(z)\big| = \big|\gamma_{p+1}A_p(z)\big| \le \big|A_p(z)\big|$$

for $z = e^{j\omega}$ on the unit circle. Then, the argument principle and Rouche's theorem imply that the addition of the function $F(z)$ will not affect the difference $N_p - Z_p$ of poles and zeros contained inside the unit circle. Thus,

$$N_{p+1} - Z_{p+1} = N_p - Z_p$$

Since the only pole of $A_p(z)$ is the multiple pole of order p at the origin arising from the term z^{-p}, it follows that $N_p = p$. Therefore,

$$(p+1) - Z_{p+1} = p - Z_p\,, \quad \text{or,}$$

$$Z_{p+1} = Z_p + 1$$

Starting at $p = 0$ with $A_0(z) = 1$, we have $Z_0 = 0$. It follows that

$$Z_p = p$$

which states that all the p zeros of the polynomial $A_p(z)$ lie inside the unit circle.

Another way to state this result is: "A necessary and sufficient condition for a polynomial $A_M(z)$ to have all of its M zeros strictly inside the unit circle is that all reflection coefficients $\{\gamma_1, \gamma_2, \dots, \gamma_M\}$ resulting from $A_M(z)$ via the backward recursion Eq. (12.3.21) have magnitude strictly less than one." This is essentially equivalent to the well-known *Schur-Cohn test* of stability [968–971]. The function **bkwlev** can be used in this regard to obtain the sequence of reflection coefficients. The Bistritz test [959], mentioned in Sec. 12.6, is an alternative stability test.

Example 12.8.1: Test the minimum phase property of the polynomials

(a) $A(z) = 1 - 2.60z^{-1} + 2.55z^{-2} - 2.80z^{-3} + 0.50z^{-4}$
(b) $A(z) = 1 - 1.40z^{-1} + 1.47z^{-2} - 1.30z^{-3} + 0.50z^{-4}$

Sending the coefficients of each through the function **bkwlev**, we find the set of reflection coefficients

(a) $\{0.4, -0.5, 2.0, -0.5\}$
(b) $\{0.4, -0.5, 0.8, -0.5\}$

Since among (a) there is one reflection coefficient of magnitude greater than one, case (a) will not be minimum phase, whereas case (b) is. □

12.9 Orthogonality of Backward Prediction Errors—Cholesky Factorization

Another interesting structural property of the lattice realizations is that, in a certain sense, the backward prediction errors $e_p^-(n)$ are orthogonal to each other. To see this, consider the case $M = 3$, and form the matrix product

$$\underbrace{\begin{bmatrix} R_0 & R_1 & R_2 & R_3 \\ R_1 & R_0 & R_1 & R_2 \\ R_2 & R_1 & R_0 & R_1 \\ R_3 & R_2 & R_1 & R_0 \end{bmatrix}}_{R}\underbrace{\begin{bmatrix} 1 & a_{11} & a_{22} & a_{33} \\ 0 & 1 & a_{21} & a_{32} \\ 0 & 0 & 1 & a_{31} \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{L^T} = \underbrace{\begin{bmatrix} E_0 & 0 & 0 & 0 \\ * & E_1 & 0 & 0 \\ * & * & E_2 & 0 \\ * & * & * & E_3 \end{bmatrix}}_{L_1}$$

Because the normal equations (written upside down) are satisfied by each prediction-error filter, the right-hand side will be a lower-triangular matrix. The “don’t care” entries have been denoted by $*$s. Multiply from the left by L to get

$$LRL^T = LL_1 = \begin{bmatrix} E_0 & 0 & 0 & 0 \\ * & E_1 & 0 & 0 \\ * & * & E_2 & 0 \\ * & * & * & E_3 \end{bmatrix}$$

Since L is by definition lower-triangular, the right-hand side will still be lower triangular. But the left-hand side is symmetric. Thus, so is the right-hand side and as a result it must be diagonal. We have shown that

$$LRL^T = D = \text{diag}\{E_0, E_1, E_2, E_3\} \tag{12.9.1}$$

or, written explicitly

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ a_{11} & 1 & 0 & 0 \\ a_{22} & a_{21} & 1 & 0 \\ a_{33} & a_{32} & a_{31} & 1 \end{bmatrix}\begin{bmatrix} R_0 & R_1 & R_2 & R_3 \\ R_1 & R_0 & R_1 & R_2 \\ R_2 & R_1 & R_0 & R_1 \\ R_3 & R_2 & R_1 & R_0 \end{bmatrix}\begin{bmatrix} 1 & a_{11} & a_{22} & a_{33} \\ 0 & 1 & a_{21} & a_{32} \\ 0 & 0 & 1 & a_{31} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} E_0 & 0 & 0 & 0 \\ 0 & E_1 & 0 & 0 \\ 0 & 0 & E_2 & 0 \\ 0 & 0 & 0 & E_3 \end{bmatrix}$$

This is identical to Eq. (1.8.17). The pqth element of this matrix equation is then

$$\mathbf{b}_p^T R \mathbf{b}_q = \delta_{pq} E_p \tag{12.9.2}$$

where $\mathbf{b}_p$ and $\mathbf{b}_q$ denote the pth and qth columns of L^T. These are recognized as the *backward prediction-error filters* of orders p and q. Eq. (12.9.2) implies then the orthogonality of the backward prediction-error filters with respect to an inner product $\mathbf{x}^T R\mathbf{y}$.

The backward prediction errors $e_p^-(n)$ can be expressed in terms of the $\mathbf{b}_p$s and the vector of samples $\mathbf{y}(n) = [y_n, y_{n-1}, y_{n-2}, y_{n-3}]^T$, as follows:

$$\begin{aligned} e_0^-(n) &= [1,\quad 0,\ \ 0,\ \ 0]\mathbf{y}(n) = \mathbf{b}_0^T\mathbf{y}(n) = y_n \\ e_1^-(n) &= [a_{11},\ \ 1,\ \ 0,\ \ 0]\mathbf{y}(n) = \mathbf{b}_1^T\mathbf{y}(n) = a_{11}y_n + y_{n-1} \\ e_2^-(n) &= [a_{22}, a_{21},\ \ 1,\ \ 0]\mathbf{y}(n) = \mathbf{b}_2^T\mathbf{y}(n) = a_{22}y_n + a_{21}y_{n-1} + y_{n-2} \\ e_3^-(n) &= [a_{33}, a_{32}, a_{31}, 1]\mathbf{y}(n) = \mathbf{b}_3^T\mathbf{y}(n) = a_{33}y_n + a_{32}y_{n-1} + a_{31}y_{n-2} + y_{n-3} \end{aligned} \tag{12.9.3}$$

which can be rearranged into the vector form

$$\mathbf{e}^-(n) = \begin{bmatrix} e_0^-(n) \\ e_1^-(n) \\ e_2^-(n) \\ e_3^-(n) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ a_{11} & 1 & 0 & 0 \\ a_{22} & a_{21} & 1 & 0 \\ a_{33} & a_{32} & a_{31} & 1 \end{bmatrix}\begin{bmatrix} y_n \\ y_{n-1} \\ y_{n-2} \\ y_{n-3} \end{bmatrix} = L\mathbf{y}(n) \tag{12.9.4}$$

It is identical to Eq. (1.8.15). Using Eq. (12.9.1), it follows now that the covariance matrix of $\mathbf{e}^-(n)$ is diagonal; indeed, since $R = E[\mathbf{y}(n)\mathbf{y}(n)^T]$,

$$R_{e^-e^-} = E[\mathbf{e}^-(n)\mathbf{e}^-(n)^T] = LRL^T = D \tag{12.9.5}$$

which can also be expressed component-wise as the zero-lag cross-correlation

$$R_{e_p^-e_q^-}(0) = E\big[e_p^-(n)e_q^-(n)\big] = \delta_{pq}E_p \tag{12.9.6}$$

Thus, at each time instant n, the backward prediction errors $e_p^-(n)$ are mutually uncorrelated (orthogonal) with each other. The orthogonality conditions (12.9.6) and the lower-triangular nature of L render the transformation (12.9.4) equivalent to the *Gram-Schmidt orthogonalization* of the data vector $\mathbf{y}(n) = [y_n, y_{n-1}, y_{n-2}, y_{n-3}]^T$. Equation (12.9.1), written as

$$R = L^{-1}DL^{-T}$$

corresponds to an *LU Cholesky factorization* of the covariance matrix R.

Since the backward errors $e_p^-(n)$, $p = 0, 1, 2, \dots, M$, for an Mth order predictor are generated at the output of each successive lattice segment of Fig. 12.7.1, we may view the analysis lattice filter as an implementation of the Gram-Schmidt orthogonalization of the vector $\mathbf{y}(n) = [y_n, y_{n-1}, y_{n-2}, \dots, y_{n-M}]^T$.

It is interesting to note, in this respect, that this implementation requires only knowledge of the reflection coefficients $\{\gamma_1, \gamma_2, \dots, \gamma_M\}$.

The data vector $\mathbf{y}(n)$ can also be orthogonalized by means of the forward predictors, using the matrix U. This representation, however, is not as conveniently realized by the lattice structure because the resulting orthogonalized vector consists of forward prediction errors that are orthogonal, but *not* at the same time instant. This can be seen from the definition of the forward errors

$$U\mathbf{y}(n) = \begin{bmatrix} 1 & a_{31} & a_{32} & a_{33} \\ 0 & 1 & a_{21} & a_{22} \\ 0 & 0 & 1 & a_{11} \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} y_n \\ y_{n-1} \\ y_{n-2} \\ y_{n-3} \end{bmatrix} = \begin{bmatrix} e_3^+(n) \\ e_2^+(n-1) \\ e_1^+(n-2) \\ e_0^+(n-3) \end{bmatrix}$$

Thus, additional delays must be inserted at the forward outputs of the lattice structure to achieve orthogonalization. For this reason, the backward outputs, being mutually orthogonal at the *same* time instant n, are preferred. The corresponding UL factorization of R is in this basis

$$URU^T = \text{diag}\{E_3, E_2, E_1, E_0\}$$

This is the reverse of Eq. (12.9.1)) obtained by acting on both sides by the reversing matrix J and using the fact that $U = JLJ$, the invariance of $R = JRJ$, and $J^2 = I$.

The above orthogonalization may also be understood in the z-domain: since the backward prediction error $e_p^-(n)$ is the output of the reversed prediction-error filter $A_p^R(z)$ driven by the data sequence y_n, we have for the cross-density

$$S_{e_p^- e_q^-}(z)=A_p^R(z)S_{yy}(z)A_q^R(z^{-1})$$

Integrating this expression over the unit circle and using Eq. (12.9.6), we find

$$\begin{aligned}\oint_{\text{u.c.}} A_p^R(z)S_{yy}(z)A_q^R(z^{-1})\,\frac{dz}{2\pi jz} &= \oint_{\text{u.c.}} S_{e_p^- e_q^-}(z)\,\frac{dz}{2\pi jz}\\ &= R_{e_p^- e_q^-}(0)=E\big[e_p^-(n)e_q^-(n)\big]=\delta_{pq}E_p\end{aligned}\qquad(12.9.7)$$

that is, the reverse polynomials $A_p^R(z)$ are *mutually orthogonal* with respect to the above inner product defined by the (positive-definite) weighting function $S_{yy}(z)$. Equation (12.9.7) is the z-domain expression of Eq. (12.9.2). This result establishes an intimate connection between the linear prediction problem and the theory of *orthogonal polynomials* on the unit circle developed by Szegö [972,973].

The LU factorization of R implies a UL factorization of the inverse of R; that is, solving Eq. (12.9.1) for R^{-1} we have:

$$R^{-1}=L^TD^{-1}L \qquad (12.9.8)$$

Since the Levinson recursion generates all the lower order prediction-error filters, it essentially generates the inverse of R.

The computation of this inverse may also be done *recursively* in the order, as follows. To keep track of the order let us use an extra index

$$R_3^{-1}=L_3^TD_3^{-1}L_3 \qquad (12.9.9)$$

The matrix L_3 contains as a submatrix the matrix L_2; in fact,

$$L_3=\left[\begin{array}{ccc|c} 1 & 0 & 0 & 0\\ a_{11} & 1 & 0 & 0\\ a_{22} & a_{21} & 1 & 0\\ \hline a_{33} & a_{32} & a_{31} & 1\end{array}\right]=\left[\begin{array}{c|c} L_2 & \mathbf{0}\\ \hline \boldsymbol{\alpha}_3^{RT} & 1\end{array}\right] \qquad (12.9.10)$$

where $\boldsymbol{\alpha}_3^{RT}$ denotes the transpose of the reverse of the vector of prediction coefficients; namely, $\boldsymbol{\alpha}_3^{RT}=[a_{33},a_{32},a_{21}]$. The diagonal matrix D_3^{-1} may also be block divided in the same manner:

$$D_3^{-1}=\left[\begin{array}{c|c} D_2^{-1} & \mathbf{0}\\ \hline \mathbf{0}^T & 1\end{array}\right]$$

Inserting these block decompositions into Eq. (12.9.9) and using the lower order result $R_2^{-1}=L_2^TD_2^{-1}L_2$, we find

$$R_3^{-1}=\left[\begin{array}{c|c} R_2^{-1}+\dfrac{1}{E_3}\boldsymbol{\alpha}_3^R\boldsymbol{\alpha}_3^{RT} & \dfrac{1}{E_3}\boldsymbol{\alpha}_3^R\\ \hline \dfrac{1}{E_3}\boldsymbol{\alpha}_3^{RT} & \dfrac{1}{E_3}\end{array}\right]=\left[\begin{array}{c|c} R_2^{-1} & \mathbf{0}\\ \hline \mathbf{0}^T & 0\end{array}\right]+\frac{1}{E_3}\mathbf{b}_3\mathbf{b}_3^T \qquad (12.9.11)$$

where $\mathbf{b}_3=\mathbf{a}_3^R=[\boldsymbol{\alpha}_3^{RT},1]^T=[a_{33},a_{32},a_{31},1]^T$. This is identical to Eq. (1.8.28).

Thus, through Levinson's algorithm, as the prediction coefficients $\boldsymbol{\alpha}_3$ and error E_3 are obtained, the inverse of R may be *updated* to the next higher order. Eq. (12.9.11) also suggests an efficient way of solving more general normal equations of the type

$$R_3\mathbf{h}_3=\begin{bmatrix} R_0 & R_1 & R_2 & R_3\\ R_1 & R_0 & R_1 & R_2\\ R_2 & R_1 & R_0 & R_1\\ R_3 & R_2 & R_1 & R_0\end{bmatrix}\begin{bmatrix} h_{30}\\ h_{31}\\ h_{32}\\ h_{33}\end{bmatrix}=\begin{bmatrix} r_0\\ r_1\\ r_2\\ r_3\end{bmatrix}=\mathbf{r}_3 \qquad (12.9.12)$$

for a given right-hand vector $\mathbf{r}_3$. Such normal equations arise in the design of FIR Wiener filters; for example, Eq. (11.3.9). The solution for $\mathbf{h}_3$ is obtained recursively from the solution of similar linear equations of lower order. For example, let $\mathbf{h}_2$ be the solution of the previous order

$$R_2\mathbf{h}_2=\begin{bmatrix} R_0 & R_1 & R_2\\ R_1 & R_0 & R_1\\ R_2 & R_1 & R_0\end{bmatrix}\begin{bmatrix} h_{20}\\ h_{21}\\ h_{22}\end{bmatrix}=\begin{bmatrix} r_0\\ r_1\\ r_2\end{bmatrix}=\mathbf{r}_2$$

where the right-hand side vector $\mathbf{r}_2$ is part of $\mathbf{r}_3$. Then, Eq. (12.9.11) implies a recursive relationship between $\mathbf{h}_3$ and $\mathbf{h}_2$:

$$\mathbf{h}_3=R_3^{-1}\mathbf{r}_3=\left[\begin{array}{c|c} R_2^{-1}+\dfrac{1}{E_3}\boldsymbol{\alpha}_3^R\boldsymbol{\alpha}_3^{RT} & \dfrac{1}{E_3}\boldsymbol{\alpha}_3^R\\ \hline \dfrac{1}{E_3}\boldsymbol{\alpha}_3^{RT} & \dfrac{1}{E_3}\end{array}\right]\left[\begin{array}{c}\mathbf{r}_2\\ \hline r_3\end{array}\right]=\left[\begin{array}{c}\mathbf{h}_2+\dfrac{1}{E_3}\boldsymbol{\alpha}_3^R(r_3+\boldsymbol{\alpha}_3^{RT}\mathbf{r}_2)\\ \hline \dfrac{1}{E_3}(r_3+\boldsymbol{\alpha}_3^{RT}\mathbf{r}_2)\end{array}\right]$$

In terms of the reverse prediction-error filter $\mathbf{b}_3=\mathbf{a}_3^R=[a_{33},a_{32},a_{31},1]^T=[\boldsymbol{\alpha}_3^{RT},1]^T$, we may write

$$\mathbf{h}_3=\left[\begin{array}{c}\mathbf{h}_2\\ \hline 0\end{array}\right]+c\,\mathbf{b}_3\,,\quad\text{where}\ \ c=\frac{1}{E_3}(r_3+\boldsymbol{\alpha}_3^{RT}\mathbf{r}_2)=\frac{1}{E_3}\mathbf{b}_3^T\mathbf{r}_3 \qquad (12.9.13)$$

Thus, the recursive updating of the solution $\mathbf{h}$ must be done by carrying out the auxiliary updating of the prediction-error filters. The method requires $O(M^2)$ operations, compared to $O(M^3)$ if the inverse of R were to be computed directly.

This recursive method of solving general normal equations, developed by Robinson and Treitel, has been reviewed elsewhere [921,922,974–976] Some additional insight into the properties of these recursions can be gained by using the Toeplitz property of R. This property together with the symmetric nature of R imply that R commutes with the reversing matrix:

$$J_3=\begin{bmatrix} 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\\ 0 & 1 & 0 & 0\\ 1 & 0 & 0 & 0\end{bmatrix}=J_3^{-1}\,,\quad J_3R_3J_3=R_3 \qquad (12.9.14)$$

Therefore, even though the inverse R_3^{-1} is not Toeplitz, it still commutes with this reversing matrix; that is,

$$J_3R_3^{-1}J_3=R_3^{-1} \qquad (12.9.15)$$

The effect of this symmetry property on the block decomposition Eq. (12.9.11) may be seen by decomposing J_3 also as

$$J_3 = \left[\begin{array}{c|c} \mathbf{0} & J_2 \\ \hline 1 & \mathbf{0}^T \end{array}\right] = \left[\begin{array}{c|c} \mathbf{0}^T & 1 \\ \hline J_2 & \mathbf{0} \end{array}\right]$$

where J_2 is the lower order reversing matrix. Combining Eq. (12.9.15) with Eq. (12.9.11), we find

$$R_3^{-1} = J_3 R_3^{-1} J_3 = \left[\begin{array}{c|c} \mathbf{0}^T & 1 \\ \hline J_2 & \mathbf{0} \end{array}\right] \left[\begin{array}{c|c} R_2^{-1} + \dfrac{1}{E_3}\boldsymbol{\alpha}_3^R \boldsymbol{\alpha}_3^{RT} & \dfrac{1}{E_3}\boldsymbol{\alpha}_3^R \\ \hline \dfrac{1}{E_3}\boldsymbol{\alpha}_3^{RT} & \dfrac{1}{E_3} \end{array}\right] \left[\begin{array}{c|c} \mathbf{0} & J_2 \\ \hline 1 & \mathbf{0}^T \end{array}\right]$$

or, since R_2 commutes with J_2, and $J_2\boldsymbol{\alpha}_3^R = \boldsymbol{\alpha}_3$, we have

$$R_3^{-1} = \left[\begin{array}{c|c} \dfrac{1}{E_3} & \dfrac{1}{E_3}\boldsymbol{\alpha}_3^T \\ \hline \dfrac{1}{E_3}\boldsymbol{\alpha}_3 & R_2^{-1} + \dfrac{1}{E_3}\boldsymbol{\alpha}_3 \boldsymbol{\alpha}_3^T \end{array}\right] = \begin{bmatrix} 0 & \mathbf{0}^T \\ \mathbf{0} & R_2^{-1} \end{bmatrix} + \frac{1}{E_3}\mathbf{a}_3 \mathbf{a}_3^T \tag{12.9.16}$$

which is the same as Eq. (1.8.35). Both ways of expressing R_3^{-1} given by Eqs. (12.9.16) and (12.9.11), are useful. They may be combined as follows: Eq. (12.9.16) gives for the ijth matrix element:

$$(R_3^{-1})_{ij} = (R_2^{-1} + \boldsymbol{\alpha}_3 \boldsymbol{\alpha}_3^T E_3^1)_{i-1,j-1} = (R_2^{-1})_{i-1,j-1} + \alpha_{3i}\alpha_{3j}E_3^{-1}$$

which valid for $1 \le i,j \le 3$. On the other hand, from Eq. (12.9.11) we have

$$(R_3^{-1})_{i-1,j-1} = (R_2^{-1})_{i-1,j-1} + \alpha_{3i}^R \alpha_{3j}^R E_3^{-1}$$

which is valid also for $1 \le i,j \le 3$. Subtracting the two to cancel the common term $(R_2^{-1})_{i-1,j-1}$, we obtain the Goberg-Semencul-Trench-Zohar recursion [977–981]:

$$(R_3^{-1})_{ij} = (R_3^{-1})_{i-1,j-1} + (\boldsymbol{\alpha}_3\boldsymbol{\alpha}_3^T - \boldsymbol{\alpha}_3^R \boldsymbol{\alpha}_3^{RT})_{ij}E_3^{-1}\,, \quad 1 \le i,j \le 3 \tag{12.9.17}$$

which allows the building-up of R_3^{-1} along each diagonal, provided one knows the “boundary” values to get these recursions started. But these are:

$$(R_3^{-1})_{00} = E_3^{-1}\,, \quad (R_3^{-1})_{i0} = (R_3^{-1})_{0i} = a_{3i}E_3^{-1}\,, \quad 1 \le i,j \le 3 \tag{12.9.18}$$

Thus, from the prediction-error filter $\mathbf{a}_3$ and its reverse, the entire inverse of the autocorrelation matrix may be built up. Computationally, of course, the best procedure is to use Eq. (12.9.8), where L and D are obtained as byproducts of the Levinson recursion. The function **lev** of the appendix starts with the $M+1$ autocorrelation lags $\{R(0), R(1), \dots, R(M)\}$ and generates the required matrices L and D. The main reason for the existence of fast algorithms for Toeplitz matrices can be traced to the nesting property that the principal submatrices of a Toeplitz matrix are simply the lower order Toeplitz submatrices. Similar fast algorithms have been developed for other types of structured matrices, such as Hankel and Vandermonde matrices [982–984].

12.10 Schur Algorithm

The Schur algorithm has its roots in the original work of Schur on the theory of functions bounded in the unit disk [985,986]. It is an important signal processing tool in a variety of contexts, such as linear prediction and signal modeling, fast matrix factorizations, filter synthesis, inverse scattering, and other applications [987–1007].

In linear prediction, Schur's algorithm is an efficient alternative to Levinson's algorithm and can be used to compute the set of reflection coefficients from the autocorrelation lags and also to compute the conventional LU Cholesky factorization of the autocorrelation matrix. The Schur algorithm is essentially the gapped function recursion (12.3.9). It proves convenient to work simultaneously with Eq. (12.3.9) and its reverse. We define the forward and backward gapped functions of order p

$$g_p^+(k) = E[e_p^+(n)y_{n-k}]\,, \quad g_p^-(k) = E[e_p^-(n)y_{n-k}] \tag{12.10.1}$$

The forward one is identical to that of Eq. (12.3.8). The backward one is the convolution of the backward filter $\mathbf{b}_p = \mathbf{a}_p^R$ with the autocorrelation function; that is,

$$g_p^+(k) = \sum_{i=0}^{p} a_{pi}R(k-i)\,, \quad g_p^-(k) = \sum_{i=0}^{p} b_{pi}R(k-i) \tag{12.10.2}$$

where $b_{pi} = a_{p,p-i}$. In the z-domain, we have

$$G_p^+(z) = A_p(z)S_{yy}(z)\,, \quad G_p^-(z) = A_p^R(z)S_{yy}(z) \tag{12.10.3}$$

Using $S_{yy}(z) = S_{yy}(z^{-1})$, it follows that

$$G_p^-(z) = A_p^R(z)S_{yy}(z) = z^{-p}A_p(z^{-1})S_{yy}(z^{-1}) = z^{-p}G_p^+(z^{-1})$$

and in the time domain:

$$g_p^-(k) = g_p^+(p-k) \tag{12.10.4}$$

Thus, the backward gapped function is the reflected and delayed version of the forward one. However, the delay is only p units—one less than required to completely align the gaps. Therefore, the forward and backward gapped functions have slightly different gaps of length p; namely,

$$\begin{aligned} g_p^+(k) &= 0\,, \quad \text{for } k = 1, 2, \dots, p \\ g_p^-(k) &= 0\,, \quad \text{for } k = 0, 1, \dots, p-1 \end{aligned} \tag{12.10.5}$$

By the definition (12.10.1), the gap conditions of the backward function are equivalent to the orthogonality conditions for the backward predictor; namely, that the estimation error $e_p^-(n)$ be orthogonal to the observations $\{y_{n-k},\ k = 0, 1, \dots, p-1\}$ that make up the estimate of y_{n-p}. Inserting the lattice recursions (12.7.7) into (12.10.1), or using the polynomial recursions (12.3.18) into (12.10.3), we obtain the lattice recursions for the gapped functions, known as the *Schur recursions*

$$\begin{aligned} g_{p+1}^+(k) &= g_p^+(k) - \gamma_{p+1}g_p^-(k-1) \\ g_{p+1}^-(k) &= g_p^-(k-1) - \gamma_{p+1}g_p^+(k) \end{aligned} \tag{12.10.6}$$

or, in matrix form

$$\begin{bmatrix} g_{p+1}^{+}(k) \\ g_{p+1}^{-}(k) \end{bmatrix} = \begin{bmatrix} 1 & -\gamma_{p+1} \\ -\gamma_{p+1} & 1 \end{bmatrix} \begin{bmatrix} g_{p}^{+}(k) \\ g_{p}^{-}(k-1) \end{bmatrix}$$

They are initialized by $g_0^{\pm}(k)= R(k)$. The first term of Eq. (12.10.6) is identical to Eq. (12.3.9) and the second term is the reverse of Eq. (12.3.9) obtained by the substitution $k \to p+1-k$. The forward gap condition $g_{p+1}^{+}(p+1)= 0$ can be solved for the reflection coefficient

$$\gamma_{p+1} = \frac{g_p^{+}(p+1)}{g_p^{-}(p)} \tag{12.10.7}$$

Note that Eq. (12.10.4) implies $g_p^{-}(p)= g_p^{+}(0)= E_p$, and therefore, Eq. (12.10.7) is the same as Eq. (12.3.11). For an Mth order predictor, we only need to consider the values $g_p^{\pm}(k)$, for $k = 0, 1, \dots, M$. We arrange these values (for the backward function) into the column vector

$$\mathbf{g}_p^{-} = \begin{bmatrix} g_p^{-}(0) \\ g_p^{-}(1) \\ \vdots \\ g_p^{-}(M) \end{bmatrix} \tag{12.10.8}$$

By virtue of the gap conditions (12.10.5), the first p entries, $k = 0, 1, \dots, p-1$, of this vector are zero. Therefore, we may construct the lower-triangular matrix having the $\mathbf{g}_p^{-}$s as columns

$$G = [\mathbf{g}_0^{-}, \mathbf{g}_1^{-}, \cdots, \mathbf{g}_M^{-}] \tag{12.10.9}$$

For example, if $M = 3$,

$$G = \begin{bmatrix} g_0^{-}(0) & 0 & 0 & 0 \\ g_0^{-}(1) & g_1^{-}(1) & 0 & 0 \\ g_0^{-}(2) & g_1^{-}(2) & g_2^{-}(2) & 0 \\ g_0^{-}(3) & g_1^{-}(3) & g_2^{-}(3) & g_3^{-}(3) \end{bmatrix}$$

The first column of G consists simply of the $M+1$ autocorrelation lags:

$$\mathbf{g}_0^{-} = \begin{bmatrix} R(0) \\ R(1) \\ \vdots \\ R(M) \end{bmatrix} \tag{12.10.10}$$

The main diagonal consists of the prediction errors of successive orders, namely, $g_p^{-}(p)= E_p$, for $p = 0, 1, \dots, M$. Stacking the values of definition (12.10.1) into a vector, we can write compactly,

$$\mathbf{g}_p^{-} = E[e_p(n)\mathbf{y}(n)] \tag{12.10.11}$$

where $\mathbf{y}(n)= [y_n, y_{n-1}, \dots, y_{n-M}]^T$ is the data vector for an Mth order predictor. Thus, the matrix G can be written as in Eq. (1.8.56)

$$G = E\Big[\mathbf{y}(n)\,[e_0^{-}(n), e_1^{-}(n), \dots, e_M^{-}(n)]\Big] = E[\mathbf{y}(n)\mathbf{e}^{-}(n)^T] \tag{12.10.12}$$

where $\mathbf{e}^{-}(n)= [e_0^{-}(n), e_1^{-}(n), \dots, e_M^{-}(n)]^T$ is the decorrelated vector of backward prediction errors. Following Eq. (1.8.57), we multiply (12.10.12) from the left by the lower triangular matrix L, and using the transformation $\mathbf{e}^{-}(n)= L\mathbf{y}(n)$ and Eq. (12.9.5), we obtain

$$LG = LE[\mathbf{y}(n)\mathbf{e}^{-}(n)^T]= E[\mathbf{e}^{-}(n)\mathbf{e}^{-}(n)^T]= D$$

Therefore, G is essentially the inverse of L

$$G = L^{-1}D \tag{12.10.13}$$

Using Eq. (12.9.1), we obtain the conventional LU Cholesky factorization of the autocorrelation matrix R in the form

$$R = L^{-1}DL^{-T} = (GD^{-1})D(D^{-1}G^T)= GD^{-1}G^T \tag{12.10.14}$$

The backward gapped functions are computed by iterating the Schur recursions (12.10.6) for $0 \le k \le M$ and $0 \le p \le M$. One computational simplification is that, because of the presence of the gap, the functions $g_p^{\pm}(k)$ need only be computed for $p \le k \le M$ (actually, $g_p^{+}(p)= 0$ could also be skipped). This gives rise to the *Schur algorithm*:

0. Initialize in order by $g_0^{\pm}(k)= R(k)$, $k = 0, 1, \dots, M$.
1. At stage p, we have available $g_p^{\pm}(k)$ for $p \le k \le M$.
2. Compute $\gamma_{p+1} = \dfrac{g_p^{+}(p+1)}{g_p^{-}(p)}$.
3. For $p+1 \le k \le M$, compute

$$\begin{aligned} g_{p+1}^{+}(k) &= g_p^{+}(k) - \gamma_{p+1}g_p^{-}(k-1) \\ g_{p+1}^{-}(k) &= g_p^{-}(k-1) - \gamma_{p+1}g_p^{+}(k) \end{aligned}$$

4. Go to stage $p+1$.
5. At the final order M, set $E_M = g_M^{-}(M)$.

The function **schur** is an implementation of this algorithm. The inputs to the function are the order M and the lags $\{R(0), R(1), \dots, R(M)\}$. The outputs are the parameters $\{E_M, \gamma_1, \gamma_2, \dots, \gamma_M\}$. This function is a simple alternative to **lev**. It may be used in conjunction with **frwlev**, **bkwlev**, and **rlev**, to pass from one linear prediction parameter set to another. The function **schur1** is a small modification of **schur** that, in addition to the reflection coefficients, outputs the lower triangular Cholesky factor G. The prediction errors can be read off from the main diagonal of G, that is, $E_P = G(p, p)$, $p = 0, 1, \dots, M$.

Example 12.10.1: Sending the five autocorrelation lags, $\{128, -64, 80, -88, 89\}$, of Example 12.3.1 through **schur1** gives the set of reflection coefficients $\{\gamma_1, \gamma_2, \gamma_3, \gamma_4\} = \{-0.5, 0.5, -0.5, 0.5\}$, and the matrix G

$$G = \begin{bmatrix} 128 & 0 & 0 & 0 & 0 \\ -64 & 96 & 0 & 0 & 0 \\ 80 & -24 & 72 & 0 & 0 \\ -88 & 36 & 0 & 54 & 0 \\ 89 & -43.5 & 13.5 & 13.5 & 40.5 \end{bmatrix}$$

Recall that the first column should be the autocorrelation lags and the main diagonal should consist of the mean square prediction errors. It is easily verified that $GD^{-1}G^T = R$. □

The computational bottleneck of the classical Levinson recursion is the computation of the inner product (12.3.12). The Schur algorithm avoids this step by computing γ_{p+1} as the ratio of the two gapped function values (12.10.7). Moreover, at each stage p, the computations indicated in step 3 of the algorithm can be done in parallel. Thus, with M parallel processors, the overall computation can be reduced to $O(M)$ operations. As formulated above, the Schur algorithm is essentially equivalent to the Le Roux-Gueguen fixed-point algorithm [990]. The possibility of a fixed-point implementation arises from the fact that all gapped functions have a fixed dynamic range, bounded by

$$|g_p^{\pm}(k)| \leq R(0) \tag{12.10.15}$$

This is easily seen by applying the Schwarz inequality to definition (12.10.1) and using $E_p \leq R(0)$

$$|g_p^{\pm}(k)|^2 = |E[e_p^{\pm}(n)y_{n-k}]|^2 \leq E[e_p^{\pm}(n)^2]E[y_{n-k}^2] \leq E_p R(0) \leq R(0)^2$$

The Schur algorithm admits a nice filtering interpretation in terms of the lattice structure. By definition, the gapped functions are the convolution of the forward/backward pth order prediction filters with the autocorrelation sequence $R(k)$. Therefore, $g_p^{\pm}(k)$ will be the outputs from the pth section of the lattice filter, Fig. 12.7.1, driven by the input $R(k)$. Moreover, Eq. (12.10.6) states that the $(p+1)$st reflection coefficient is obtainable as the ratio of the two *inputs* to the $(p+1)$st lattice section, at time instant $p+1$ (note that $g_p^-(p) = g_p^-(p+1-1)$ is outputted at time p from the pth section and is delayed by one time unit before it is inputted to the $(p+1)$st section at time $p+1$.) The correct values of the gapped functions $g_p^{\pm}(k)$ are obtained when the input to the lattice filter is the infinite double-sided sequence $R(k)$. If we send in the finite *causal* sequence

$$x(k) = \{R(0), R(1), \dots, R(M), 0, 0, \dots\}$$

then, because of the initial and final transient behavior of the filter, the outputs of the pth section will agree with $g_p^{\pm}(k)$ only for $p \leq k \leq M$. To see this, let $y_p^{\pm}(k)$ denote the two outputs. Because of the causality of the input and filter and the finite length of the input, the convolutional filtering equation will be

$$y_p^+(k) = \sum_{i=\max\{0,k-M\}}^{\min\{p,k\}} a_{pi}\, x(k-i) = \sum_{i=\max\{0,k-M\}}^{\min\{p,k\}} a_{pi}\, R(k-i)$$

This agrees with Eq. (12.10.2) only after time p and before time M, that is,

$$y_p^{\pm}(k) = g_p^{\pm}(k)\,, \quad \text{only for } p \leq k \leq M$$

The column vector $\mathbf{y}_p^- = [y_p^-(0), y_p^-(1), \dots, y_p^-(M)]^T$, formed by the first M backward output samples of the pth section, will agree with $\mathbf{g}_p^-$ only for the entries $p \leq k \leq M$. Thus, the matrix of backward outputs $Y^- = [\mathbf{y}_0^-, \mathbf{y}_1^-, \dots, \mathbf{y}_M^-]$ formed by the columns $\mathbf{y}_p^-$ will agree with G only in its *lower-triangular* part. But this is enough to determine G because its upper part is zero.

Example 12.10.2: Send the autocorrelation lags of Example 12.10.1 into the lattice filter of Fig. 12.7.1 (with all its delay registers initialized to zero), arrange the forward/backward outputs from the pth section into the column vectors, $\mathbf{y}_p^{\pm}$, and put these columns together to form the output matrices $Y^{\pm}$. The result is,

$$Y^- = \begin{bmatrix} 128 & 64 & -64 & 64 & -64 \\ -64 & 96 & 64 & -80 & 96 \\ 80 & -24 & 72 & 64 & -96 \\ -88 & 36 & 0 & 54 & 64 \\ 89 & -43.5 & 13.5 & 13.5 & 40.5 \end{bmatrix}, \quad Y^+ = \begin{bmatrix} 128 & 128 & 128 & 128 & 128 \\ -64 & 0 & -32 & -64 & -96 \\ 80 & 48 & 0 & 32 & 72 \\ -88 & -48 & -36 & 0 & -32 \\ 89 & 45 & 27 & 27 & 0 \end{bmatrix}$$

The lower-triangular part of Y^- agrees with G. The forward/backward outputs $\mathbf{y}_p^{\pm}$ can be computed using, for example, the function **lattice**. They can also be computed directly by convolving the prediction filters with the input. For example, the backward filter of order 4 given in Example 12.3.1 is $\mathbf{a}_4^R = [-0.5, 0.5, -0.1875, -0.25, 1]^T$. Convolving it with the autocorrelation sequence gives the last column of Y^-

$$[128, -64, 80, -88, 89] * [-0.5, 0.5, -0.1875, -0.25, 1] = [-64, 96, -96, 64, 40.5, \dots]$$

Convolving the forward filter $\mathbf{a}_4$ with the autocorrelation sequence gives the last column of the matrix Y^+

$$[128, -64, 80, -88, 89] * [1, -0.25, -0.1875, 0.5, -0.5] = [128, -96, 72, -32, 0, \dots]$$

Note that we are interested only in the outputs for $0 \leq k \leq M = 4$. The last 4 outputs (in general, the last p outputs for a pth order filter) of these convolutions were not shown. They correspond to the transient behavior of the filter after the input is turned off. □

It is also possible to derive a *split* or *immitance-domain* version of the Schur algorithm that achieves a further 50% reduction in computational complexity [962,963]. Thus, with M parallel processors, the complexity of the Schur algorithm can be reduced to $O(M/2)$ operations. We define a symmetrized or split gapped function in terms of the symmetric polynomial $F_p(z)$ defined in Eq. (12.6.1)

$$g_p(k) = \sum_{i=0}^{p} f_{pi}\, R(k-i)\,, \quad G_p(z) = F_p(z) S_{yy}(z) \tag{12.10.16}$$

It can be thought of as the output of the filter $F_p(z)$ driven by the autocorrelation sequence. Multiplying both sides of Eq. (12.6.1) by $S_{yy}(z)$ and using the definition (12.10.3), we obtain $G_p(z) = G_{p-1}^+(z) + z^{-1}G_{p-1}^-(z)$, or, in the time domain

$$g_p(k) = g_{p-1}^+(k) + g_{p-1}^-(k-1) \tag{12.10.17}$$

Similarly, Eq. (12.6.2) gives

$$(1-\gamma_p)\, g_p(k) = g_p^+(k) + g_p^-(k) \tag{12.10.18}$$

It follows from Eqs. (12.10.4) and (12.10.18) or from the symmetry property of $F_p(z)$ that $g_p(k) = g_p(p-k)$, and in particular, $g_p(0) = g_p(p)$. The split Levinson algorithm of Sec. 12.6 requires the computation of the coefficients $\alpha_{p+1} = \tau_{p+1}/\tau_p$. Setting $k = 0$ in the definition (12.10.16) and using the reflection symmetry $R(i)=R(-i)$, we recognize that the inner product of Eq. (12.6.6) is $\tau_p = g_p(0)=g_p(p)$. Therefore, the coefficient α_{p+1} can be written as the ratio of the two gapped function values

$$\alpha_{p+1} = \frac{g_{p+1}(p+1)}{g_p(p)} \tag{12.10.19}$$

Because the forward and backward gapped functions have overlapping gaps, it follows that $g_p(k)$ will have gap $g_p(k)=0$, for $k=1,2,\dots,p-1$. Therefore, for an Mth order predictor, we only need to know the values of $g_p(k)$ for $p \le k \le M$. These can be computed by the following three-term recurrence, obtained by multiplying the recurrence (12.6.4) by $S_{yy}(z)$

$$g_{p+2}(k)=g_{p+1}(k)+g_{p+1}(k-1)-\alpha_{p+1}g_p(k-1) \tag{12.10.20}$$

Using $F_0(z)= 2$ and $F_1(z)= 1+z^{-1}$, it follows from the definition that $g_0(k)= 2R(k)$ and $g_1(k)=R(k)+R(k-1)$. To initialize τ_0 correctly, however, we must choose $g_0(0)=R(0)$, so that $\tau_0 = g_0(0)=R(0)$. Thus, we are led to the following *split Schur algorithm*:

0. Initialize by $g_0(k)= 2R(k)$, $g_1(k)= R(k)+R(k-1)$, for $k=1,2,\dots,M$, and $g_0(0)=R(0)$, $\gamma_0=0$.
1. At stage p, we have available γ_p, $g_p(k)$ for $p\le k\le M$, and $g_{p+1}(k)$ for $p+1\le k\le M$.
2. Compute α_{p+1} from Eq. (12.10.19) and solve for $\gamma_{p+1}=-1+\alpha_{p+1}/(1-\gamma_p)$.
3. For $p+2\le k\le M$, compute $g_{p+2}(k)$ using Eq. (12.10.20)
4. Go to stage $p+1$.

Recalling that $E_p = \tau_p(1-\gamma_p)$, we may set at the final order $E_M=\tau_M(1-\gamma_M)= g_M(M)(1-\gamma_M)$. Step 3 of the algorithm requires only one multiplication for each k, whereas step 3 of the ordinary Schur algorithm requires *two*. This reduces the computational complexity by 50%. The function **schur2** (see Appendix B) is an implementation of this algorithm. The inputs to the function are the order M and the lags $\{R(0),R(1),\dots,R(M)\}$. The outputs are the parameters $\{E_M,\gamma_1,\gamma_2,\dots,\gamma_M\}$. The function can be modified easily to include the computation of the backward gapped functions $g_p^-(k)$, which are the columns of the Cholesky matrix G. This can be done by the recursion

$$g_p^-(k)=g_p^-(k-1)+(1-\gamma_p)g_p(k)-g_{p+1}(k) \tag{12.10.21}$$

where $p+1\le k\le M$, with starting value $g_p^-(p)=E_p=g_p(p)(1-\gamma_p)$. This recursion will generate the lower-triangular part of G. Equation (12.10.21) follows by writing Eq. (12.10.17) for order $(p+1)$ and subtracting it from Eq. (12.10.18). Note, also, that Eq. (12.10.17) and the bound (12.10.15) imply the bound $|g_p(k)|\le 2R(0)$, which allows a fixed-point implementation.

We finish this section by discussing the connection of the Schur algorithm to Schur's original work. It follows from Eq. (12.10.3) that the ratio of the two gapped functions

$G_p^\pm(z)$ is an *all-pass stable* transfer function, otherwise known as a *lossless bounded real* function [971]:

$$S_p(z)=\frac{G_p^-(z)}{G_p^+(z)}=\frac{A_p^R(z)}{A_p(z)}=\frac{a_{pp}+a_{p,p-1}z^{-1}+\cdots+z^{-p}}{1+a_{p1}z^{-1}+\cdots+a_{pp}z^{-p}} \tag{12.10.22}$$

The all-pass property follows from the fact that the reverse polynomial $A^R(z)$ has the same magnitude response as $A_p(z)$. The stability property follows from the minimum-phase property of the polynomials $A_p(z)$, which in turn is equivalent to all reflection coefficients having magnitude less than one. Such functions satisfy the boundedness property

$$\big|S_p(z)\big|\le 1\,,\quad \text{for } |z|\ge 1 \tag{12.10.23}$$

with equality attained on the unit circle. Taking the limit $z\to\infty$, it follows from Eq. (12.10.22) that the reflection coefficient γ_p is obtainable from $S_p(z)$ by

$$S_p(\infty)=a_{pp}=-\gamma_p \tag{12.10.24}$$

Using the backward Levinson recursion reqs5.3.23, we obtain a new all-pass function

$$S_{p-1}(z)=\frac{G_{p-1}^-(z)}{G_{p-1}^+(z)}=\frac{A_{p-1}^R(z)}{A_{p-1}(z)}=\frac{z(\gamma_pA_p+A_p^R)}{A_p+\gamma_pA_p^R}$$

or, dividing numerator and denominator by $A_p(z)$

$$S_{p-1}(z)=z\,\frac{S_p(z)+\gamma_p}{1+\gamma_pS_p(z)} \tag{12.10.25}$$

This is Schur's original recursion [985]. Applying this recursion repeatedly from some initial value $p=M$ down to $p=0$, with $S_0(z)= 1$, will give rise to the set of reflection or Schur coefficients $\{\gamma_1,\gamma_2,\dots,\gamma_M\}$. The starting all-pass function $S_M(z)$ will be stable if and only if all reflection coefficients have magnitude less than one. We note finally that there is an intimate connection between the Schur algorithm and inverse scattering problems [991,995,1001,1002,1005–1007,1053].

In Sec. 12.13, we will see that the lattice recursions (12.10.6) describe the forward and backward moving waves incident on a layered structure. The Schur function $S_p(z)$ will correspond to the overall reflection response of the structure, and the recursion (12.10.25) will describe the successive removal of the layers. The coefficients γ_p will represent the elementary reflection coefficients at the layer interfaces. This justifies the term *reflection coefficients* for the γs.

12.11 Lattice Realizations of FIR Wiener Filters

In this section, we combine the results of Sections 11.3 and 12.9 to derive alternative realizations of Wiener filters that are based on the Gram-Schmidt lattice structures. Consider the FIR Wiener filtering problem of estimating a desired signal x_n, on the basis of the related signal y_n, using an Mth order filter. The I/O equation of the optimal filter is given by Eq. (11.3.8). The vector of optimal weights is determined by solving the set of normal equations, given by Eq. (11.3.9). The discussion of the previous section suggests that Eq. (11.3.9) can be solved efficiently using the Levinson recursion. Defining the data vector

$$\mathbf{y}(n)=\begin{bmatrix} y_n \\ y_{n-1} \\ \vdots \\ y_{n-M}\end{bmatrix} \qquad (12.11.1)$$

we rewrite Eq. (11.3.9) in the compact matrix form

$$R_{yy}\mathbf{h}=\mathbf{r}_{xy} \qquad (12.11.2)$$

where R_{yy} is the $(M+1)\times(M+1)$ autocorrelation matrix of $\mathbf{y}(n)$, and $\mathbf{r}_{xy}$, the $(M+1)$-vector of cross-correlations between x_n, and $\mathbf{y}(n)$, namely,

$$R_{yy}=E\big[\mathbf{y}(n)\mathbf{y}(n)^T\big]\,,\quad \mathbf{r}_{xy}=E[x_n\mathbf{y}(n)]=\begin{bmatrix} R_{xy}(0) \\ R_{xy}(1) \\ \vdots \\ R_{xy}(M)\end{bmatrix} \qquad (12.11.3)$$

and $\mathbf{h}$ is the $(M+1)$-vector of optimal weights

$$\mathbf{h}=\begin{bmatrix} h_0 \\ h_1 \\ \vdots \\ h_M\end{bmatrix} \qquad (12.11.4)$$

The I/O equation of the filter, Eq. (12.9.4), is

$$\hat{x}_n=\mathbf{h}^T\mathbf{y}(n)=h_0y_n+h_1y_{n-1}+\cdots+h_My_{n-M} \qquad (12.11.5)$$

Next, consider the Gram-Schmidt transformation of Eq. (12.9.4) from the data vector $\mathbf{y}(n)$ to the decorrelated vector $\mathbf{e}^-(n)$:

$$\mathbf{e}^-(n)=L\mathbf{y}(n)\quad\text{or,}\quad \begin{bmatrix} e_0^-(n) \\ e_1^-(n) \\ \vdots \\ e_M^-(n)\end{bmatrix}=L\begin{bmatrix} y_n \\ y_{n-1} \\ \vdots \\ y_{n-M}\end{bmatrix} \qquad (12.11.6)$$

Inserting (12.11.6) into (12.11.5), we find

$$\hat{x}_n=\mathbf{h}^TL^{-1}\mathbf{e}^-(n)$$

Defining the $(M+1)$-vector

$$\mathbf{g}=L^{-T}\mathbf{h} \qquad (12.11.7)$$

we obtain the alternative I/O equation for the Wiener filter:

$$\hat{x}_n=\mathbf{g}^T\mathbf{e}^-(n)=\sum_{p=0}^{M}g_pe_p^-(n)=g_0e_0^-(n)+g_1e_1^-(n)+\cdots+g_Me_M^-(n) \qquad (12.11.8)$$

This is easily recognized as the projection of x_n onto the subspace spanned by $\{e_0^-(n),e_1^-(n),\dots,e_M^-(n)\}$, which is the same as that spanned by the data vector $\{y_n, y_{n-1},\dots,y_{n-M}\}$. Indeed, it follows from Eqs. (12.11.7) and (12.11.2) that

$$\begin{aligned}\mathbf{g}^T=\mathbf{h}^TL^{-1}&=E[x_n\mathbf{y}(n)^T]E[\mathbf{y}(n)\mathbf{y}(n)^T]^{-1}L^{-1}\\ &=E[x_n\mathbf{e}^-(n)^T]L^{-T}\big(L^{-1}E[\mathbf{e}^-(n)\mathbf{e}^-(n)^T]L^{-T}\big)^{-1}L^{-1}\\ &=E[x_n\mathbf{e}^-(n)^T]E[\mathbf{e}^-(n)\mathbf{e}^-(n)^T]^{-1}\\ &=\big[E[x_ne_0^-(n)]/E_0,\;E[x_ne_1^-(n)]/E_1,\dots,\;E[x_ne_M^-(n)]/E_M\big]\end{aligned}$$

so that the estimate of x_n can be expressed as

$$\hat{x}_n=E[x_n\mathbf{e}^-(n)^T]E[\mathbf{e}^-(n)\mathbf{e}^-(n)^T]^{-1}\mathbf{e}^-(n)=E[x_n\mathbf{y}(n)^T]E[\mathbf{y}(n)\mathbf{y}(n)^T]^{-1}\mathbf{y}(n)$$

The key to the lattice realization of the optimal filtering equation (12.11.8) is the observation that the *analysis lattice filter* of Fig. 12.7.1 for the process y_n, provides, in its successive lattice stages, the signals $e_p^-(n)$ which are required in the sum (12.11.8). Thus, if the weight vector $\mathbf{g}$ is known, an alternative realization of the optimal filter will be as shown in Fig. 12.11.1. By comparison, the direct form realization using Eq. (12.11.5) operates directly on the vector $\mathbf{y}(n)$, which, at each time instant n, is available at the tap registers of the filter. This is depicted in Fig. 12.11.2.

Both types of realizations can be formulated *adaptively*, without requiring prior knowledge of the filter coefficients or the correlation matrices R_{yy} and $\mathbf{r}_{xy}$. We will discuss adaptive implementations in Chap. 16. If R_{yy} and $\mathbf{r}_{xy}$ are known, or can be estimated, then the design procedure for both the lattice and the direct form realizations is implemented by the following three steps:

1. Using Levinson's algorithm, implemented by the function **lev**, perform the LU Cholesky factorization of R_{yy}, to determine the matrices L and D.

2. The vector of weights $\mathbf{g}$ can be computed in terms of the known quantities $L,D,\mathbf{r}_{xy}$ as follows:

$$\mathbf{g}=L^{-T}\mathbf{h}=L^{-T}R_{yy}^{-1}\mathbf{r}_{xy}=L^{-T}\big(L^TD^{-1}L\big)\mathbf{r}_{xy}=D^{-1}L\mathbf{r}_{xy}$$

3. The vector $\mathbf{h}$ can be recovered from $\mathbf{g}$ by $\mathbf{h}=L^T\mathbf{g}$.

The function **firw** is an implementation of this design procedure. The inputs to the function are the order M and the correlation lags $\{R_{yy}(0),R_{yy}(1),\dots,R_{yy}(M)\}$ and $\{R_{xy}(0),R_{xy}(1),\dots,R_{xy}(M)\}$. The outputs are the quantities $L,D,\mathbf{g}$, and $\mathbf{h}$. The estimate (12.11.8) may also be written recursively in the order of the filter. If we denote,

$$\hat{x}_p(n)=\sum_{i=0}^{p}g_ie_i^-(n) \qquad (12.11.9)$$

we obtain the recursion

$$\hat{x}_p(n)=\hat{x}_{p-1}(n)+g_pe_p^-(n)\,,\quad p=0,1,\dots,M \qquad (12.11.10)$$

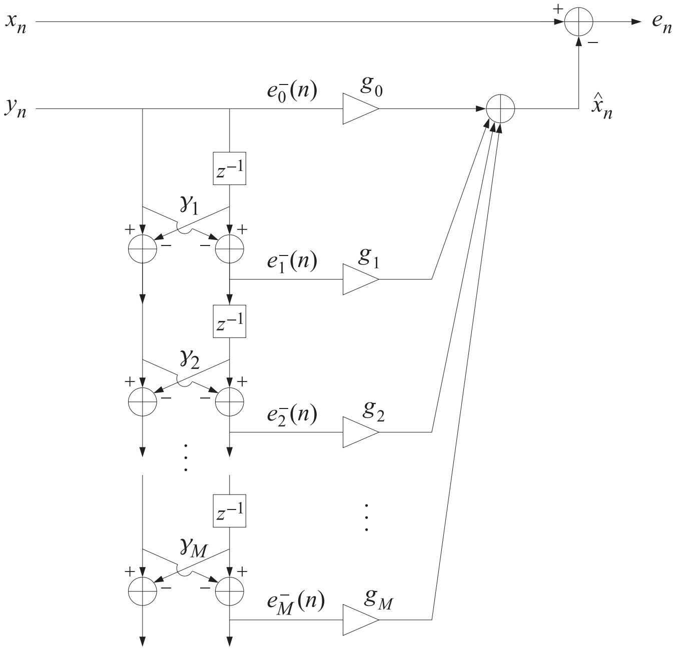


Fig. 12.11.1 Lattice realization of FIR Wiener filter.

initialized as $\hat{x}_{-1}(n)= 0$. The quantity $\hat{x}_p(n)$ is the *projection* of x_n on the subspace spanned by $\{e_0^-(n), e_1^-(n), \dots, e_p^-(n)\}$, which by virtue of the lower-triangular nature of the matrix L is the same space as that spanned by $\{y_n, y_{n-1}, \dots, y_{n-p}\}$. Thus, $\hat{x}_p(n)$ represents the optimal estimate of x_n based on a pth order filter. Similarly, $\hat{x}_{p-1}(n)$ represents the optimal estimate of x_n based on the $(p-1)$th order filter; that is, based on the past $p-1$ samples $\{y_n, y_{n-1}, \dots, y_{n-p+1}\}$. These two subspaces differ by y_{n-p}.

The term $e_p^-(n)$ is by construction the best postdiction error of estimating y_{n-p} from the samples $\{y_n, y_{n-1}, \dots, y_{n-p+1}\}$; that is, $e_p^-(n)$ is the *orthogonal complement* of y_{n-p} projected on that subspace. Therefore, the term $g_p e_p^-(n)$ in Eq. (12.11.10) represents the improvement in the estimate of x_n that results by taking into account the *additional* past value y_{n-p}; it represents that part of x_n that cannot be estimated in terms of the subspace $\{y_n, y_{n-1}, \dots, y_{n-p+1}\}$. The estimate $\hat{x}_p(n)$ of x_n is *better* than $\hat{x}_{p-1}(n)$ in the sense that it produces a smaller mean-squared estimation error. To see this, define the estimation errors in the two cases

$$e_p(n)= x_n - \hat{x}_p(n)\,, \quad e_{p-1}(n)= x_n - \hat{x}_{p-1}(n)$$

Using the recursion (12.11.10), we find

$$e_p(n)= e_{p-1}(n)-g_p e_p^-(n) \tag{12.11.11}$$

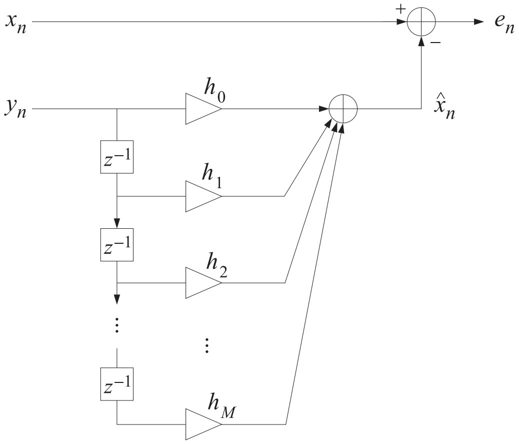


Fig. 12.11.2 Direct-form realization of FIR Wiener filter.

Using $g_p = E[x_n e_p^-(n)]/E_p$, we find for $\mathcal{E}_p = E[e_p(n)^2]$

$$\mathcal{E}_p = E[x_n^2]-\sum_{i=0}^{p} g_i E[x_n e_i^-(n)]= \mathcal{E}_{p-1} - g_p E[x_n e_p^-(n)]$$

$$= \mathcal{E}_{p-1} - \big(E[x_n e_p^-(n)]\big)^2/E_p = \mathcal{E}_{p-1} - g_p^2 E_p$$

Thus, $\mathcal{E}_p$ is smaller than $\mathcal{E}_{p-1}$. This result shows explicitly how the estimate is constantly improved as the length of the filter is increased. The nice feature of the lattice realization is that the filter length can be increased simply by adding more lattice sections without having to recompute the weights g_p of the previous sections. A realization equivalent to Fig. 12.11.1, but which shows explicitly the recursive construction (12.11.10) of the estimate of x_n and of the estimation error (12.11.11), is shown in Fig. 12.11.3.

The function **lwf** is an implementation of the lattice Wiener filter of Fig. 12.11.3. The function **dwf** implements the direct-form Wiener filter of Fig. 12.11.2. Each call to these functions transforms a pair of input samples $\{x, y\}$ into the pair of output samples $\{\hat{x}, e\}$ and updates the internal state of the filter. Successive calls over $n = 0, 1, 2, \dots,$ will transform the input sequences $\{x_n, y_n\}$ into the output sequences $\{\hat{x}_n, e_n\}$. In both realizations, the internal state of the filter is taken to be the vector of samples stored in the delays of the filter; that is, $w_p(n)= e_{p-1}^-(n-1)$, $p = 1, 2, \dots, M$ for the lattice case, and $w_p(n)= y_{n-p}$, $p = 1, 2, \dots, M$ for the direct-form case. By allowing the filter coefficients to change between calls, these functions can be used in adaptive implementations.

Next, we present a Wiener filter design example for a noise canceling application. The primary and secondary signals $x(n)$ and $y(n)$ are of the form

$$x(n)= s(n)+v_1(n)\,, \quad y(n)= v_2(n)$$

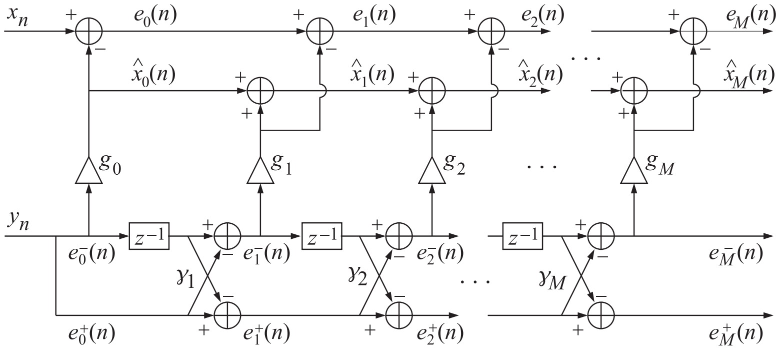
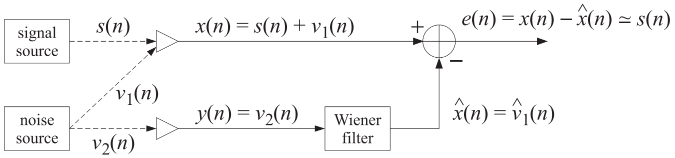


Fig. 12.11.3 Lattice realization of FIR Wiener filter.

where $s(n)$ is a desired signal corrupted by noise $v_1(n)$. The signal $v_2(n)$ is correlated with $v_1(n)$ but not with $s(n)$, and provides a reference noise signal. The noise canceler is to be implemented as a Wiener filter of order M, realized either in the direct or the lattice form. It is shown below:



Its basic operation is that of a correlation canceler; that is, the optimally designed filter $H(z)$ will transform the reference noise $v_2(n)$ into the best replica of $v_1(n)$, and then proceed to cancel it from the output, leaving a clean signal $s(n)$. For the purpose of the simulation, we took $s(n)$ to be a simple sinusoid

$$s(n)=\sin(\omega_0 n)\,,\quad \omega_0=0.075\pi\ \text{[rads/sample]}$$

and $v_1(n)$ and $v_2(n)$ were generated by the difference equations

$$\begin{aligned} v_1(n) &= -0.5v_1(n-1)+v(n)\\ v_2(n) &= 0.8v_2(n-1)+v(n) \end{aligned}$$

driven by a common, zero-mean, unit-variance, uncorrelated sequence $v(n)$. The difference equations establish a correlation between the two noise components v_1 and v_2, which is exploited by the canceler to effect the noise cancellation.

Figs. 12.11.4 and 12.11.5 show 100 samples of the signals $x(n)$, $s(n)$, and $y(n)$ generated by a particular realization of $v(n)$. For $M=4$ and $M=6$, the sample autocorrelation and cross-correlation lags, $R_{yy}(k)$, $R_{xy}(k)$, $k=0,1,\dots,M$, were computed and sent through the function **firw** to get the filter weights **g** and **h**.

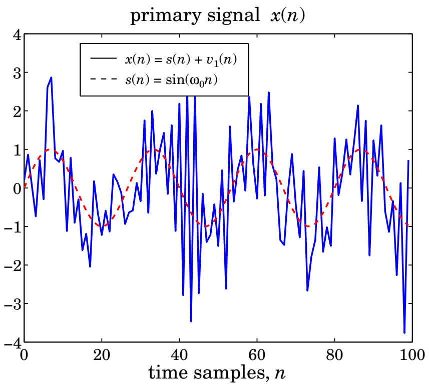


Fig. 12.11.4 Noise corrupted sinusoid.

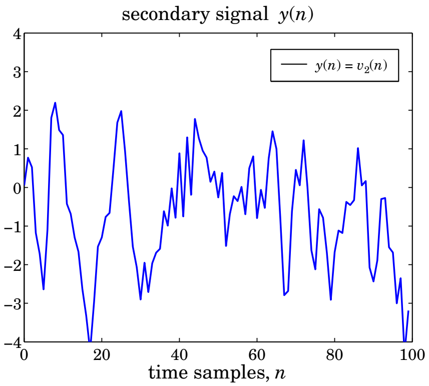


Fig. 12.11.5 Reference noise.

The reference signal y_n was filtered through $H(z)$ to get the estimate $\hat{x}_n$—which is really an estimate of $v_1(n)$—and the estimation error $e(n)=x(n)-\hat{x}(n)$, which is really an estimate of $s(n)$. This estimate of $s(n)$ is shown in Figs. (12.11.6) and 12.11.7, for the cases $M=4$ and $M=6$, respectively. The improvement afforded by a higher order filter is evident. For the particular realization of $x(n)$ and $y(n)$ that we used, the *sample* correlations $R_{yy}(k)$, $R_{xy}(k)$, $k=0,1,\dots,M$, were:

$$\begin{aligned} R_{yy} &= [2.5116,\ 1.8909,\ 1.2914,\ 0.6509,\ 0.3696,\ 0.2412,\ 0.1363]\\ R_{xy} &= [0.7791,\ -0.3813,\ 0.0880,\ -0.3582,\ 0.0902,\ -0.0684,\ 0.0046] \end{aligned}$$

and the resulting vector of lattice weights g_p, $p=0,1,\dots,M$, reflection coefficients γ_p,

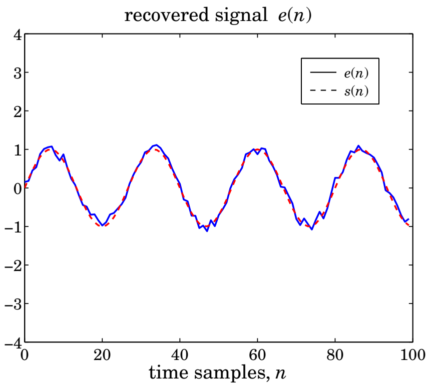


Fig. 12.11.6 Output of noise canceler ($M = 4$).

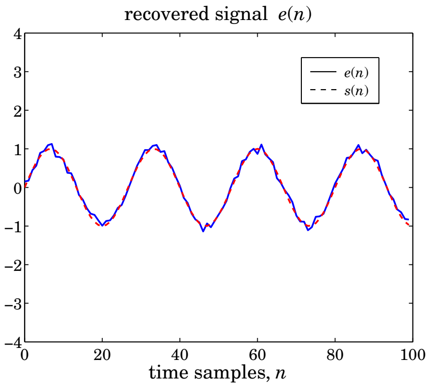


Fig. 12.11.7 Output of noise canceler ($M = 6$).

$p = 1, 2, \dots, M$, and direct-form weights h_m, $m = 0, 1, \dots, M$ were for $M = 6$,

$$\mathbf{g} = [0.3102,\ -0.8894,\ 0.4706,\ -0.2534,\ 0.1571,\ -0.0826,\ 0.0398]$$

$$\boldsymbol{\gamma} = [0.7528,\ -0.1214,\ -0.1957,\ 0.1444,\ 0.0354,\ -0.0937]$$

$$\mathbf{h} = [0.9713,\ -1.2213,\ 0.6418,\ -0.3691,\ 0.2245,\ -0.1163,\ 0.0398]$$

To get the $\mathbf{g}$ and $\boldsymbol{\gamma}$ of the case $M = 4$, simply ignore the last two entries in the above. The corresponding $\mathbf{h}$ is in this case:

$$\mathbf{h} = [0.9646,\ -1.2262,\ 0.6726,\ -0.3868,\ 0.1571]$$

Using the results of Problems 12.25 and 12.26, we may compute the theoretical filter weights for this example, and note that they compare fairly well with the estimated ones that were based on the length-100 data blocks. For $M = 6$, we have:

$$\mathbf{g} = [0.2571,\ -0.9286,\ 0.4643,\ -0.2321,\ 0.1161,\ -0.0580,\ 0.0290]$$

$$\boldsymbol{\gamma} = [0.8,\ 0,\ 0,\ 0,\ 0,\ 0]$$

$$\mathbf{h} = [1,\ -1.3,\ 0.65,\ -0.325,\ 0.1625,\ -0.0812,\ 0.0290]$$

As we discussed in Sec. 1.8, the lattice realizations based on the backward orthogonal basis have three major advantages over the direct-form realizations: (a) the filter processes non-redundant information only, and hence adaptive implementations would adapt faster; (b) the design of the optimal filter weights $\mathbf{g}$ does not require any matrix inversion; and (c) the lower-order portions of $\mathbf{g}$ are already optimal. Moreover, it appears that adaptive versions of the lattice realizations have better numerical properties than the direct-form versions. In array processing problems, because the data vector $\mathbf{y}(n)$ does not have the tapped-delay line form (12.11.1), the Gram-Schmidt orthogonalization cannot be done by a simple a lattice filter. It requires a more complicated structure that basically amounts to carrying out the lower-triangular linear transformation (12.11.6). The benefits, however, are the same. We discuss adaptive versions of Gram-Schmidt preprocessors for arrays in Chap. 16.

12.12 Autocorrelation, Covariance, and Burg's Methods

As mentioned in Sec. 12.3, the finite order linear prediction problem may be thought of as an approximation to the infinite order prediction problem. For large enough order p of the predictor, the prediction-error filter $A_p(z)$ may be considered to be an adequate approximation to the whitening filter $A(z)$ of the process y_n. In this case, the prediction-error sequence $e_p^+(n)$ is approximately white, and the inverse synthesis filter $1/A_p(z)$ is an approximation to the signal model $B(z)$ of y_n. Thus, we have obtained an approximate solution to the signal modeling problem depicted below:

$$e_p^+(n) \longrightarrow \boxed{1/A_p(z)} \longrightarrow y_n$$

The variance of $e_p^+(n)$ is E_p. Depending on the realization one uses, the model parameters are either the set $\{a_{p1}, a_{p2}, \dots, a_{pp}; E_p\}$, or, $\{\gamma_1, \gamma_2, \dots, \gamma_p; E_p\}$. Because these can be determined by solving a simple linear system of equations—that is, the normal equations (12.3.7)—this approach to the modeling problem has become widespread.

In this section, we present three widely used methods of extracting the model parameters from a given block of measured signal values y_n [917,920,926,927,1008–1018]. These methods are:

1. The autocorrelation, or Yule-Walker, method
2. The covariance method.
3. Burg's method.

We have already discussed the Yule-Walker method, which consists simply of replacing the theoretical autocorrelations $R_{yy}(k)$ with the corresponding sample autocorrelations $\hat{R}_{yy}(k)$ computed from the given frame of data. This method, like the other two, can be justified on the basis of an appropriate least-squares minimization criterion obtained by replacing the ensemble averages $E[e_p^+(n)^2]$ by appropriate time averages.

The theoretical minimization criteria for the optimal forward and backward predictors are

$$E[e_p^+(n)^2]=\min\,,\quad E[e_p^-(n)^2]=\min \tag{12.12.1}$$

where $e_p^+(n)$ and $e_p^-(n)$ are the result of filtering y_n through the prediction-error filter $\mathbf{a}=[1,a_{p1},\dots,a_{pp}]^T$ and its reverse $\mathbf{a}^R=[a_{pp},a_{p,p-1},\dots,a_{p1},1]^T$, respectively; namely,

$$\begin{aligned} e_p^+(n) &= y_n + a_{p1}y_{n-1} + a_{p2}y_{n-2} + \cdots + a_{pp}y_{n-p} \\ e_p^-(n) &= y_{n-p} + a_{p1}y_{n-p+1} + a_{p2}y_{n-p+2} + \cdots + a_{pp}y_n \end{aligned} \tag{12.12.2}$$

Note that in both cases the mean-square value of $e_p^\pm(n)$ can be expressed in terms of the $(p+1)\times(p+1)$ autocorrelation matrix

$$R(i,j)=R(i-j)=E[y_{n+i-j}y_n]=E[y_{n-j}y_{n-i}]\,,\quad 0\le i,j\le p$$

as follows

$$E[e_p^+(n)^2]=E[e_p^-(n)^2]=\mathbf{a}^TR\mathbf{a} \tag{12.12.3}$$

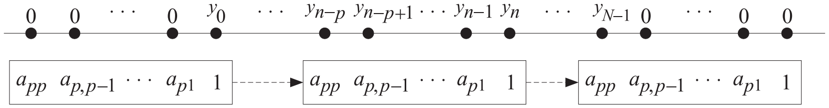
Consider a frame of length N of measured values of y_n

$$\boxed{y_0,y_1,\dots,y_{N-1}}$$

1. The *Yule-Walker*, or *autocorrelation*, method replaces the ensemble average (12.12.1) by the least-squares time-average criterion

$$\mathcal{E}=\sum_{n=0}^{N+p-1}e_p^+(n)^2=\min \tag{12.12.4}$$

where $e_p^+(n)$ is obtained by convolving the length-$(p+1)$ prediction-error filter $\mathbf{a}=[1,a_{p1},\dots,a_{pp}]^T$ with the length-N data sequence y_n. The length of the sequence $e_p^+(n)$ is, therefore, $N+(p+1)-1=N+p$, which justifies the upper-limit in the summation of Eq. (12.12.4). This convolution operation is equivalent to assuming that the block of data y_n has been extended both to the left and to the right by padding it with zeros and running the filter over this *extended* sequence. The last p output samples $e_p^+(n)$, $N\le n\le N+p-1$, correspond to running the filter off the ends of the data sequence to the right. These terms arise because the prediction-error filter has memory of p samples. This is depicted below:


0 0 · · · 0 y0 · · · yn−p yn−p+1 · · · yn−1 yn · · · yN−1 0 · · · 0 0
app ap,p−1 · · · ap1 1
app ap,p−1 · · · ap1 1
app ap,p−1 · · · ap1 1

Inserting Eq. (12.12.2) into (12.12.4), it is easily shown that $\mathcal{E}$ can be expressed in the equivalent form

$$\mathcal{E}=\sum_{n=0}^{N+p-1}e_p^+(n)^2=\sum_{i,j=0}^{p}a_i\hat{R}(i-j)a_j=\mathbf{a}^T\hat{R}\mathbf{a} \tag{12.12.5}$$

where $\hat{R}(k)$ denotes the sample autocorrelation of the length-N data sequence y_n:

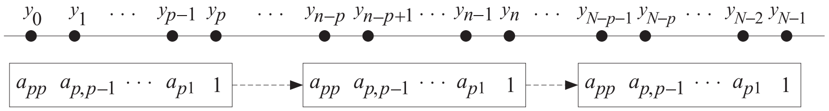
$$\hat{R}(k)=\hat{R}(-k)=\sum_{n=0}^{N-1-k}y_{n+k}y_n\,,\quad 0\le k\le N-1$$

where the usual normalization factor $1/N$ has been ignored. This equation is identical to Eq. (12.12.3) with R replaced by $\hat{R}$. Thus, the minimization of the time-average index (12.12.5) with respect to the prediction coefficients will lead exactly to the *same set* of normal equations (12.3.7) with R replaced by $\hat{R}$. The *positive definiteness* of the sample autocorrelation matrix also guarantees that the resulting prediction-error filter will be *minimum phase*, and thus also that all reflection coefficients will have magnitude less than one.

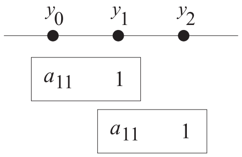
2. The *covariance method* replaces Eq. (12.12.1) by the time average

$$\mathcal{E}=\sum_{n=p}^{N-1}e_p^+(n)^2=\min \tag{12.12.6}$$

where the summation in n is such that the filter *does not* run off the ends of the data block, as shown below:


y0 y1 · · · yp−1 yp · · · yn−p yn−p+1 · · · yn−1 yn · · · yN−p−1 yN−p · · · yN−2 yN−1
app ap,p−1 · · · ap1 1
app ap,p−1 · · · ap1 1
app ap,p−1 · · · ap1 1

To explain the method and to see its potential problems with stability, consider a simple example of a length-three sequence and a first-order predictor:



$$\mathcal{E}=\sum_{n=1}^{2}e_1^+(n)^2=e_1^+(1)^2+e_1^+(2)^2=(y_1+a_{11}y_0)^2+(y_2+a_{11}y_1)^2$$

Differentiating with respect to a_{11} and setting the derivative to zero gives

$$(y_1+a_{11}y_0)y_0+(y_2+a_{11}y_1)y_1=0$$

$$a_{11}=-\frac{y_1y_0+y_2y_1}{y_0^2+y_1^2}$$

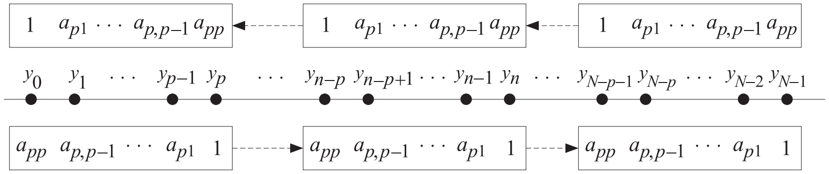
Note that the denominator does not depend on the variable y_2 and therefore it is possible, if y_2 is large enough, for a_{11} to have magnitude greater than one, making the prediction-error filter nonminimal phase. Although this potential stability problem exists, this method has been used with good success in speech processing, with few, if any, such stability problems. The autocorrelation method is sometimes preferred in speech processing because the resulting normal equations have a Toeplitz structure and their solution can be obtained efficiently using Levinson's algorithm. However, similar ways of solving the covariance equations have been developed recently that are just as efficient [1013].

3. Although the autocorrelation method is implemented efficiently, and the resulting prediction-error filter is guaranteed to be minimum phase, it suffers from the effect of windowing the data sequence y_n, by padding it with zeros to the left and to the right. This reduces the accuracy of the method somewhat, especially when the data record N is short. In this case, the effect of windowing is felt more strongly. The proper way to extend the sequence y_n, if it must be extended, is a way compatible with the signal model generating this sequence. Since we are trying to determine this model, the fairest way of proceeding is to try to use the available data block in a way which is maximally noncommittal as to what the sequence is like beyond the ends of the block.

Burg's method, also known as the *maximum entropy* method (MEM), arose from the desire on the one hand not to run off the ends of the data, and, on the other, to always result in a minimum-phase filter. Burg's minimization criterion is to minimize the sum-squared of both the forward and the backward prediction errors:

$$\mathcal{E}=\sum_{n=p}^{N-1}\left[e_p^+(n)^2+e_p^-(n)^2\right]=\min \tag{12.12.7}$$

where the summation range is the same as in the covariance method, but with both the forward and the reversed filters running over the data, as shown:


1 a_{p1} ··· $a_{p,p-1}$ a_{pp}
1 a_{p1} ··· $a_{p,p-1}$ a_{pp}
1 a_{p1} ··· $a_{p,p-1}$ a_{pp}
y_0 y_1 ··· y_{p-1} y_p ··· y_{n-p} y_{n-p+1} ··· y_{n-1} y_n ··· y_{N-p-1} y_{N-p} ··· y_{N-2} y_{N-1}
a_{pp} $a_{p,p-1}$ ··· a_{p1} 1
a_{pp} $a_{p,p-1}$ ··· a_{p1} 1
a_{pp} $a_{p,p-1}$ ··· a_{p1} 1

If the minimization is performed with respect to the coefficients a_{pi}, it is still possible for the resulting prediction-error filter not to be minimum phase. Instead, Burg suggests an *iterative procedure*: Suppose that the prediction-error filter $[1,a_{p-1,1},a_{p-1,2},\dots,a_{p-1,p-1}]$ of order $(p-1)$ has already been determined. Then, to determine the prediction-error filter of order p, one needs to know the reflection coefficient γ_p and to apply the Levinson recursion:

$$\begin{bmatrix}1\\a_{p1}\\a_{p2}\\\vdots\\a_{p,p-1}\\a_{pp}\end{bmatrix}=\begin{bmatrix}1\\a_{p-1,1}\\a_{p-1,2}\\\vdots\\a_{p-1,p-1}\\0\end{bmatrix}-\gamma_p\begin{bmatrix}0\\a_{p-1,p-1}\\a_{p-1,p-2}\\\vdots\\a_{p-1,1}\\1\end{bmatrix} \tag{12.12.8}$$

To guarantee the minimum-phase property, the reflection coefficient γ_p must have magnitude less than one. The best choice for γ_p is that which minimizes the performance index (12.12.7). Differentiating with respect to γ_p and setting the derivative to zero we find

$$\frac{\partial\mathcal{E}}{\partial\gamma_p}=2\sum_{n=p}^{N-1}\left[e_p^+(n)\frac{\partial e_p^+(n)}{\partial\gamma_p}+e_p^-(n)\frac{\partial e_p^-(n)}{\partial\gamma_p}\right]=0$$

Using the lattice relationships

$$\begin{aligned}e_p^+(n)&=e_{p-1}^+(n)-\gamma_p e_{p-1}^-(n-1)\\e_p^-(n)&=e_{p-1}^-(n-1)-\gamma_p e_{p-1}^+(n)\end{aligned} \tag{12.12.9}$$

both valid for $p\le n\le N-1$ if the filter is not to run off the ends of the data, we find the condition

$$\sum_{n=p}^{N-1}\left[e_p^+(n)e_{p-1}^-(n-1)+e_p^-(n)e_{p-1}^+(n)\right]=0\,,\quad\text{or,}$$

$$\sum_{n=p}^{N-1}\left[\big(e_{p-1}^+(n)-\gamma_p e_{p-1}^-(n-1)\big)e_{p-1}^-(n-1)+\big(e_{p-1}^-(n-1)-\gamma_p e_{p-1}^+(n)\big)e_{p-1}^+(n)\right]=0$$

which can be solved for γ_p to give

$$\gamma_p=\frac{2\displaystyle\sum_{n=p}^{N-1}e_{p-1}^+(n)e_{p-1}^-(n-1)}{\displaystyle\sum_{n=p}^{N-1}\left[e_{p-1}^+(n)^2+e_{p-1}^-(n-1)^2\right]} \tag{12.12.10}$$

This expression for γ_p is of the form

$$\gamma_p=\frac{2\mathbf{a}\cdot\mathbf{b}}{|\mathbf{a}|^2+|\mathbf{b}|^2}$$

where $\mathbf{a}$ and $\mathbf{b}$ are vectors. Using the Schwarz inequality, it is easily verified that γ_p has magnitude less than one. Equations (12.12.8) through (12.12.10) define Burg's method. The computational steps are summarized below:

0. Initialize in order as follows:

$$e_0^+(n)=e_0^-(n)=y_n\,,\ \text{for } 0\le n\le N-1\,,\quad\text{and } A_0(z)=1,\ E_0=\frac{1}{N}\sum_{n=0}^{N-1}y_n^2$$

1. At stage $(p-1)$, we have available the quantities:

$$A_{p-1}(z),\ E_{p-1},\ \text{and } e_{p-1}^{\pm}(n),\ \text{for } p-1\le n\le N-1$$

2. Using Eq. (12.12.10), compute the reflection coefficient γ_p.
3. Using (12.12.8), compute $A_p(z)$.
4. Using (12.12.9), compute $e_p^{\pm}(n)$, for $p\le n\le N-1$.
5. Update the mean-square error by $E_p=(1-\gamma_p^2)E_{p-1}$.
6. Go to stage p.

The function **burg** is an implementation of this method. The inputs to the function are the vector of data samples $\{y_0, y_1, \dots, y_{N-1}\}$ and the desired final order M of the predictor. The outputs are all the prediction-error filters of order up to M, arranged as usual into the lower triangular matrix L, and the corresponding mean-square prediction errors $\{E_0, E_1, \dots, E_M\}$.

Example 12.12.1: The length-six block of data

$$y_n = [4.684,\ 7.247,\ 8.423,\ 8.650,\ 8.640,\ 8.392]$$

for $n = 0, 1, 2, 3, 4, 5$, is known to have been generated by sending zero-mean, unit-variance, white-noise ϵ_n through the difference equation

$$y_n - 1.70y_{n-1} + 0.72y_{n-2} = \epsilon_n$$

Thus, the theoretical prediction-error filter and mean-square error are $A_2(z) = 1 - 1.70z^{-1} + 0.72z^{-2}$ and $E_2 = 1$. Using Burg's method, extract the model parameters for a second-order model. The reader is urged to go through the algorithm by hand. Sending the above six y_n samples through the function **burg**, we find the first- and second-order prediction-error filters and the corresponding errors:

$$A_1(z) = 1 - 0.987z^{-1}\,, \quad E_1 = 1.529$$

$$A_2(z) = 1 - 1.757z^{-1} + 0.779z^{-2}\,, \quad E_2 = 0.60$$

We note that the theoretical first-order filter obtained from $A_2(z) = 1 - 1.70z^{-1} + 0.72z^{-2}$ via the backward Levinson recursion is $A_1(z) = 1 - 0.9884\,z^{-1}$. □

The resulting set of LPC model parameters, from any of the above analysis methods, can be used in a number of ways as suggested in Sec. 1.13. One of the most successful applications has been to the analysis and synthesis of speech [920,1019–1027]. Each frame of speech, of duration of the order of 20 msec, is subjected to the Yule-Walker analysis method to extract the corresponding set of model parameters. The order M of the predictor is typically 10–15. Pitch and voiced/unvoiced information are also extracted. The resulting set of parameters represents that speech segment.

To synthesize the segment, the set of model parameters are recalled from memory and used in the synthesizer to drive the synthesis filter. The latter is commonly realized as a *lattice filter*. Lattice realizations are preferred because they are much better well-behaved under quantization of their coefficients (i.e., the reflection coefficients) than the direct-form realizations [920,1023,1024]. A typical speech analysis and synthesis system is shown in Fig. 12.12.1.

Linear predictive modeling techniques have also been applied to EEG signal processing in order to model EEG spectra, to *classify* EEGs automatically, to detect EEG *transients* that might have diagnostic significance, and to *predict* the onset of epileptic seizures [1028–1035].

LPC methods have been applied successfully to *signal classification* problems such as speech recognition [1022,1036–1041] or the automatic classification of EEGs [1032]. *Distance measures* between two sets of model parameters extracted from two signal frames can be used as measures of similarity between the frames. Itakura's *LPC distance measure* can be introduced as follows: Consider two *autoregressive* signal sequences, the test sequence $y_T(n)$ to be compared against the reference sequence $y_R(n)$. Let $A_T(z)$ and $A_R(z)$ be the two whitening filters, both of order M. The two signal models are

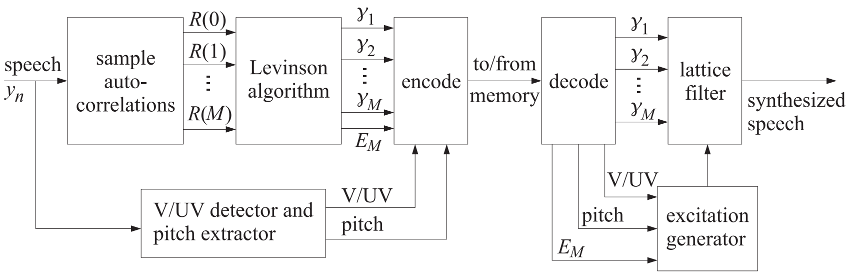


Fig. 12.12.1 LPC analysis and synthesis of speech.

$$\epsilon_T(n) \longrightarrow \boxed{1/A_T(z)} \longrightarrow y_T(n) \qquad \epsilon_R(n) \longrightarrow \boxed{1/A_R(z)} \longrightarrow y_R(n)$$

Now, suppose the sequence to be tested, $y_T(n)$, is filtered through the whitening filter of the reference signal

$$y_T(n) \longrightarrow \boxed{A_R(z)} \longrightarrow e_T(n)$$

resulting in the output signal $e_T(n)$. The mean output power is easily expressed as

$$E[e_T(n)^2] = \mathbf{a}_R^\dagger R_T \mathbf{a}_R = \int_{-\pi}^{\pi} S_{e_T e_T}(\omega)\,\frac{d\omega}{2\pi} = \int_{-\pi}^{\pi} |A_R(\omega)|^2 S_{y_T y_T}(\omega)\,\frac{d\omega}{2\pi}$$

$$= \int_{-\pi}^{\pi} |A_R(\omega)|^2 \frac{\sigma_{\epsilon_T}^2}{|A_T(\omega)|^2}\,\frac{d\omega}{2\pi}$$

where R_T is the autocorrelation matrix of $y_T(n)$. On the other hand, if $y_T(n)$ is filtered through its own whitening filter, it will produce $\epsilon_T(n)$. Thus, in this case

$$\sigma_{\epsilon_T}^2 = E[\epsilon_T(n)^2] = \mathbf{a}_T^\dagger R_T \mathbf{a}_T$$

It follows that

$$\frac{E[e_T(n)^2]}{E[\epsilon_T(n)^2]} = \frac{\mathbf{a}_R^\dagger R_T \mathbf{a}_R}{\mathbf{a}_T^\dagger R_T \mathbf{a}_T} = \int_{-\pi}^{\pi} \frac{|A_R(\omega)|^2}{|A_T(\omega)|^2}\,\frac{d\omega}{2\pi} \tag{12.12.11}$$

The log of this quantity is Itakura's LPC distance measure

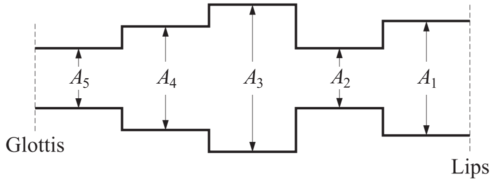
$$d(\mathbf{a}_T, \mathbf{a}_R) = \log\left(\frac{E[e_T(n)^2]}{E[\epsilon_T(n)^2]}\right) = \log\left(\frac{\mathbf{a}_R^\dagger R_T \mathbf{a}_R}{\mathbf{a}_T^\dagger R_T \mathbf{a}_T}\right) = \log\left[\int_{-\pi}^{\pi} \frac{|A_R(\omega)|^2}{|A_T(\omega)|^2}\,\frac{d\omega}{2\pi}\right]$$

In practice, the quantities $\mathbf{a}_T$, R_T, and $\mathbf{a}_R$ are extracted from a frame of $y_T(n)$ and a frame of $y_R(n)$. If the model parameters are equal, the distance is zero. This distance measure effectively provides a comparison between the two spectra of the processes y_T and y_R, but instead of comparing them directly, a prewhitening of $y_T(n)$ is carried out by sending it through the whitening filter of the other signal. If the two spectra are close, the filtered signal $e_T(n)$ will be close to white—that is, with a spectrum close to being flat; a measure of this flatness is precisely the above integrated spectrum of Eq. (12.12.11).

12.13 Dynamic Predictive Deconvolution—Waves in Layered Media

The analysis and synthesis lattice filters, implemented via the Levinson recursion, were obtained within the context of linear prediction. Here, we would like to point out the remarkable fact that the same analysis and synthesis lattice structures also occur naturally in the problem of wave propagation in layered media [920–925,974,976,1010,1019,1042–1059]. This is perhaps the reason behind the great success of linear prediction methods in speech and seismic signal processing. In fact, historically many linear prediction techniques were originally developed within the context of these two application areas.

In speech, the vocal tract is modeled as an acoustic tube of varying cross-sectional area. It can be approximated by the piece-wise constant area approximation shown below:

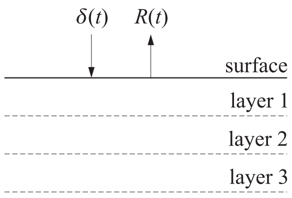


The acoustic impedance of a sound wave varies inversely with the tube area

$$Z = \frac{\rho c}{A}$$

where ρ, c, A are the air density, speed of sound, and tube area, respectively. Therefore, as the sound wave propagates from the glottis to the lips, it will suffer reflections every time it encounters an interface; that is, every time it enters a tube segment of different diameter. Multiple reflections will be set up within each segment and the tube will *reverberate* in a complicated manner depending on the number of segments and the diameter of each segment. By measuring the speech wave that eventually comes out of the lips, it is possible to remove, or deconvolve, the reverberatory effects of the tube and, in the process, extract the *tube parameters*, such as the areas of the segments or, equivalently, the reflection coefficients at the interfaces. During speech, the configuration of the vocal tract tube changes continuously. But being a mechanical system, it does so fairly slowly, and for short periods of time (of the order of 20–30 msec) it may be assumed to maintain a fixed configuration. From each such short segment of speech, a set of configuration parameters (e.g., reflection coefficients) may be extracted. This set may be used to synthesize the speech segment.

The seismic problem is somewhat different. Here it is not the transmitted wave that is experimentally accessible, but rather the overall reflected wave:

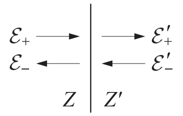


An impulsive input to the earth, such as a dynamite explosion near the surface, will set up seismic elastic waves propagating downwards. As the various earth layers are encountered, reflections will take place. Eventually each layer will be reverberating and an overall reflected wave will be measured at the surface. On the basis of this reflected wave, the layered structure (i.e., reflection coefficients, impedances, etc.) must be extracted by deconvolution techniques. These are essentially identical to the linear prediction methods.

In addition to geophysical and speech applications, this wave problem and the associated *inverse* problem of extracting the structure of the medium from the observed (reflected or transmitted) response have a number of other applications. Examples include the probing of dielectric materials by electromagnetic waves, the study of the optical properties of thin films, the probing of tissues by ultrasound, and the design of broadband terminations of transmission lines. The mathematical analysis of such wave propagation problems has been done more or less independently in each of these application areas, and is well known dating back to the time of Stokes.

In this type of wave propagation problem there are always *two* associated propagating field quantities, the ratio of which is constant and equal to the corresponding *characteristic impedance* of the propagation medium. Examples of these include the electric and magnetic fields in the case of EM waves, the air pressure and particle volume velocity for sound waves, the stress and particle displacement for seismic waves, and the voltage and current waves in the case of TEM transmission lines.

As a concrete example, we have chosen to present in some detail the case of EM waves propagating in lossless dielectrics. The simplest and most basic scattering problem arises when there is a single interface separating two semi-infinite dielectrics of characteristic impedances Z and Z', as shown



where $\mathcal{E}_+$ and $\mathcal{E}_-$ are the right and left moving electric fields in medium Z, and $\mathcal{E}'_+$ and $\mathcal{E}'_-$ are those in medium Z'. The arrows indicate the directions of propagation, the fields are perpendicular to these directions. Matching the boundary conditions (i.e., continuity of the tangential fields at the interface), gives the two equations:

$$\mathcal{E}_+ + \mathcal{E}_- = \mathcal{E}'_+ + \mathcal{E}'_- \qquad \text{(continuity of electric field)}$$

$$\frac{1}{Z}(\mathcal{E}_+ - \mathcal{E}_-) = \frac{1}{Z'}(\mathcal{E}'_+ - \mathcal{E}'_-) \qquad \text{(continuity of magnetic field)}$$

Introducing the reflection and transmission coefficients,

$$\rho = \frac{Z' - Z}{Z' + Z}, \quad \tau = 1 + \rho, \quad \rho' = -\rho, \quad \tau' = 1 + \rho' = 1 - \rho \tag{12.13.1}$$

the above equations can be written in a *transmission matrix* form

$$\begin{bmatrix} \mathcal{E}_+ \\ \mathcal{E}_- \end{bmatrix} = \frac{1}{\tau}\begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix}\begin{bmatrix} \mathcal{E}'_+ \\ \mathcal{E}'_- \end{bmatrix} \tag{12.13.2}$$

The flow of energy carried by these waves is given by the Poynting vector

$$P = \frac{1}{2}\,\mathrm{Re}\left[(\mathcal{E}_+ + \mathcal{E}_-)^*\frac{1}{Z}(\mathcal{E}_+ - \mathcal{E}_-)\right] = \frac{1}{2Z}(\mathcal{E}_+^*\mathcal{E}_+ - \mathcal{E}_-^*\mathcal{E}_-) \tag{12.13.3}$$

One consequence of the above matching conditions is that the total energy flow to the right is preserved *across* the interface; that is,

$$\frac{1}{2Z}(\mathcal{E}_+^*\mathcal{E}_+ - \mathcal{E}_-^*\mathcal{E}_-) = \frac{1}{2Z'}(\mathcal{E}'^*_+\mathcal{E}'_+ - \mathcal{E}'^*_-\mathcal{E}'_-) \tag{12.13.4}$$

It proves convenient to absorb the factors $1/2Z$ and $1/2Z'$ into the definitions for the fields by renormalizing them as follows:

$$\begin{bmatrix} E_+ \\ E_- \end{bmatrix} = \frac{1}{\sqrt{2Z}}\begin{bmatrix} \mathcal{E}_+ \\ \mathcal{E}_- \end{bmatrix}, \qquad \begin{bmatrix} E'_+ \\ E'_- \end{bmatrix} = \frac{1}{\sqrt{2Z'}}\begin{bmatrix} \mathcal{E}'_+ \\ \mathcal{E}'_- \end{bmatrix}$$

Then, Eq. (12.13.4) reads

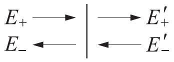
$$E_+^*E_+ - E_-^*E_- = E'^*_+E'_+ - E'^*_-E'_- \tag{12.13.5}$$

and the matching equations (12.13.2) can be written in the normalized form

$$\begin{bmatrix} E_+ \\ E_- \end{bmatrix} = \frac{1}{t}\begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix}\begin{bmatrix} E'_+ \\ E'_- \end{bmatrix}, \quad t = \sqrt{1-\rho^2} = \sqrt{\tau\tau'} \tag{12.13.6}$$

They may also be written in a *scattering matrix* form that relates the *outgoing* fields to the *incoming* ones, as follows:

$$\begin{bmatrix} E'_+ \\ E_- \end{bmatrix} = \begin{bmatrix} t & \rho' \\ \rho & t \end{bmatrix}\begin{bmatrix} E_+ \\ E'_- \end{bmatrix} = S\begin{bmatrix} E_+ \\ E'_- \end{bmatrix} \tag{12.13.7}$$



This is the most elementary scattering matrix of all, and ρ and t are the most elementary reflection and transmission responses. From these, the reflection and transmission response of more complicated structures can be built up. In the more general case, we have a dielectric structure consisting of M slabs stacked together as shown in Fig. 12.13.1.

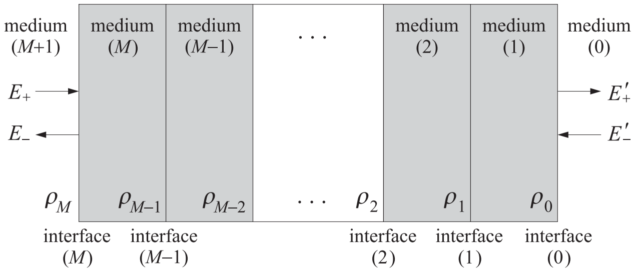


Fig. 12.13.1 Layered structure.

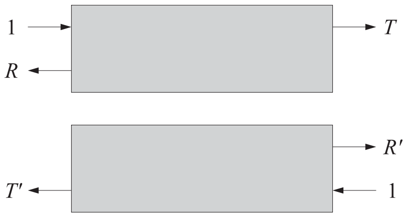


Fig. 12.13.2 Reflection and transmission responses.

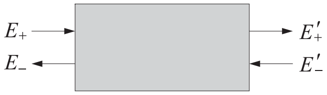
The media to the left and right in the figure are assumed to be semi-infinite. The reflection and transmission responses (from the left, or from the right) of the structure are defined as the responses of the structure to an impulse (incident from the left, or from the right) as shown in Fig. 12.13.2.

The corresponding scattering matrix is defined as

$$S = \begin{bmatrix} T & R' \\ R & T' \end{bmatrix}$$

and by linear superposition, the relationship between arbitrary incoming and outgoing waves is

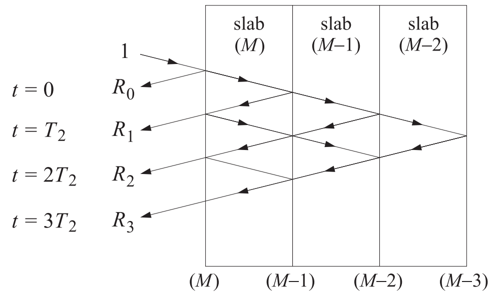
$$\begin{bmatrix} E'_+ \\ E_- \end{bmatrix} = \begin{bmatrix} T & R' \\ R & T' \end{bmatrix}\begin{bmatrix} E_+ \\ E'_- \end{bmatrix}$$



The *inverse scattering problem* that we pose is how to extract the detailed properties of the layered structure, such as the reflection coefficients $\rho_0, \rho_1, \dots, \rho_M$ from the knowledge of the scattering matrix S; that is, from observations of the reflection response R or the transmission response T.

Without loss of generality, we may assume the M slabs have *equal travel time*. We denote the common one-way travel time by T_1 and the two-way travel time by $T_2 = 2T_1$.

As an impulse $\delta(t)$ is incident from the left on interface M, there will be immediately a reflected wave and a transmitted wave into medium M. When the latter reaches interface $M-1$, part of it will be transmitted into medium $M-1$, and part will be reflected back towards interface M where it will be partially rereflected towards $M-1$ and partially transmitted to the left into medium $M+1$, thus contributing towards the overall reflection response. Since the wave had to travel to interface $M-1$ and back, this latter contribution will occur at time T_2. Similarly, another wave will return back to interface M due to reflection from the second interface $M-2$; this wave will return $2T_2$ seconds later and will add to the contribution from the zig-zag path within medium M which is also returning at $2T_2$, and so on. The timing diagram below shows all the possible return paths up to time $t=3T_2$, during which the original impulse can only travel as far as interface $M-3$:



When we add the contributions of all the returned waves we see that the reflection response will be a linear superposition of returned impulses

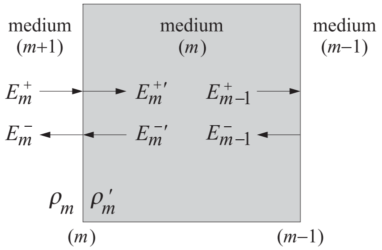
$$R(t)=\sum_{k=0}^{\infty}R_k\delta(t-kT_2)$$

It has a Fourier transform expressible more conveniently as the z-transform

$$R(z)=\sum_{k=0}^{\infty}R_kz^{-k}\,,\quad z=e^{j\omega T_2}\,,\quad \text{(here, }\omega\text{ is in rads/sec)}$$

We observe that R is *periodic* in frequency ω with period $2\pi/T_2$, which plays a role analogous to the sampling frequency in a sample-data system. Therefore, it is enough to specify R within the *Nyquist interval* $[-\pi/T_2,\pi/T_2]$.

Next, we develop the *lattice recursions* that facilitate the solution of the direct and the inverse scattering problems. Consider the mth slab and let $E_m^{\pm}$ be the right/left moving waves incident on the left side of the mth interface. To relate them to the same quantities $E_{m-1}^{\pm}$ incident on the left side of the $(m-1)$st interface, first we use the *matching* equations to “pass” to the other side of the mth interface and into the mth slab, and then we *propagate* these quantities to reach the left side of the $(m-1)$st interface. This is shown below.



The matching equations are:

$$\begin{bmatrix}E_m^+\\E_m^-\end{bmatrix}=\frac{1}{t_m}\begin{bmatrix}1&\rho_m\\\rho_m&1\end{bmatrix}\begin{bmatrix}E_m^{+\prime}\\E_m^{-\prime}\end{bmatrix}\,,\quad t_m=(1-\rho_m^2)^{1/2} \tag{12.13.8}$$

Since the left-moving wave $E_m^{-\prime}$ is the delayed replica of E_{m-1}^- by T_1 seconds, and $E_m^{+\prime}$ is the advanced replica of E_{m-1}^+ by T_1 seconds, it follows that

$$E_m^{+\prime}=z^{1/2}E_{m-1}^+\,,\quad E_m^{-\prime}=z^{-1/2}E_{m-1}^-$$

or, in matrix form

$$\begin{bmatrix}E_m^{+\prime}\\E_m^{-\prime}\end{bmatrix}=\begin{bmatrix}z^{1/2}&0\\0&z^{-1/2}\end{bmatrix}\begin{bmatrix}E_{m-1}^+\\E_{m-1}^-\end{bmatrix} \tag{12.13.9}$$

where the variable z^{-1} was defined above and represents the two-way travel time delay, while $z^{-1/2}$ represents the one-way travel time delay. Combining the matching and propagation equations (12.13.8) and (12.13.9), we obtain the desired relationship between $E_m^{\pm}$ and $E_{m-1}^{\pm}$:

$$\begin{bmatrix}E_m^+\\E_m^-\end{bmatrix}=\frac{z^{1/2}}{t_m}\begin{bmatrix}1&\rho_mz^{-1}\\\rho_m&z^{-1}\end{bmatrix}\begin{bmatrix}E_{m-1}^+\\E_{m-1}^-\end{bmatrix} \tag{12.13.10}$$

Or, written in a convenient vector notation

$$E_m(z)=\psi_m(z)E_{m-1}(z) \tag{12.13.11}$$

where we defined

$$E_m(z)=\begin{bmatrix}E_m^+(z)\\E_m^-(z)\end{bmatrix}\,,\quad \psi_m(z)=\frac{z^{1/2}}{t_m}\begin{bmatrix}1&\rho_mz^{-1}\\\rho_m&z^{-1}\end{bmatrix} \tag{12.13.12}$$

The “match-and-propagate” transition matrix $\psi_m(z)$ has two interesting properties; namely, defining $\bar{\psi}_m(z)=\psi_m(z^{-1})$

$$\bar{\psi}_m(z)^TJ_3\psi_m(z)=J_3\,,\quad J_3=\begin{bmatrix}1&0\\0&-1\end{bmatrix} \tag{12.13.13}$$

$$\bar{\psi}_m(z)=J_1\psi_m(z)J_1\,,\quad J_1=\begin{bmatrix}0&1\\1&0\end{bmatrix} \tag{12.13.14}$$

where J_1, J_3 are recognized as two of the three Pauli spin matrices. From Eq. (12.3.13), we have with $\bar{E}_m^{\pm}(z)= E_m^{\pm}(z^{-1})$:

$$\begin{aligned}\bar{E}_m^+E_m^+ - \bar{E}_m^-E_m^- &= \bar{E}_m^TJ_3E_m = \bar{E}_{m-1}^T\bar{\psi}_m^TJ_3\psi_mE_{m-1} = \bar{E}_{m-1}^TJ_3\bar{E}_{m-1}\\ &= \bar{E}_{m-1}^+E_{m-1}^+ - \bar{E}_{m-1}^-E_{m-1}^-\end{aligned} \tag{12.13.15}$$

which is equivalent to *energy conservation*, according to Eq. (12.13.5). The second property, Eq. (12.13.14), expresses *time-reversal invariance* and allows the construction of a second, linearly independent, solution of the recursive equations (12.13.11), Using the property $J_1^2 = I$, we have

$$\hat{E}_m = J_1\bar{E}_m = \begin{bmatrix}\bar{E}_m^-\\ \bar{E}_m^+\end{bmatrix} = J_1\bar{\psi}_m\bar{E}_{m-1} = J_1\bar{\psi}_mJ_1J_1\bar{E}_{m-1} = \psi_m\hat{E}_{m-1} \tag{12.13.16}$$

The recursions (12.13.11) may be iterated now down to $m = 0$. By an additional boundary match, we may pass to the right side of interface $m = 0$:

$$E_m = \psi_m\psi_{m-1}\cdots\psi_1E_0 = \psi_m\psi_{m-1}\cdots\psi_1\psi_0E_0'$$

where we defined ψ_0 by

$$\psi_0 = \frac{1}{t_0}\begin{bmatrix}1 & \rho_0\\ \rho_0 & 1\end{bmatrix}$$

or, more explicitly

$$\begin{bmatrix}E_m^+\\ E_m^-\end{bmatrix} = \frac{z^{m/2}}{t_mt_{m-1}\cdots t_1t_0}\begin{bmatrix}1 & \rho_mz^{-1}\\ \rho_m & z^{-1}\end{bmatrix}\cdots\begin{bmatrix}1 & \rho_1z^{-1}\\ \rho_1 & z^{-1}\end{bmatrix}\begin{bmatrix}1 & \rho_0\\ \rho_0 & 1\end{bmatrix}\begin{bmatrix}E_0^{+\prime}\\ E_0^{-\prime}\end{bmatrix} \tag{12.13.17}$$

To deal with this product of matrices, we define

$$\begin{bmatrix}A_m & C_m\\ B_m & D_m\end{bmatrix} = \begin{bmatrix}1 & \rho_mz^{-1}\\ \rho_m & z^{-1}\end{bmatrix}\cdots\begin{bmatrix}1 & \rho_1z^{-1}\\ \rho_1 & z^{-1}\end{bmatrix}\begin{bmatrix}1 & \rho_0\\ \rho_0 & 1\end{bmatrix} \tag{12.13.18}$$

where A_m, C_m, B_m, D_m are *polynomials of degree m* in the variable z^{-1}. The energy conservation and time-reversal invariance properties of the ψ_m matrices imply similar properties for these polynomials. Writing Eq. (12.13.18) in terms of the ψ_ms, we have

$$\begin{bmatrix}A_m & C_m\\ B_m & D_m\end{bmatrix} = z^{-m/2}\sigma_m\psi_m\psi_{m-1}\cdots\psi_1\psi_0$$

where we defined the quantity

$$\sigma_m = t_mt_{m-1}\cdots t_1t_0 = \prod_{i=0}^{m}(1-\rho_i^2)^{1/2} \tag{12.13.19}$$

Property (12.13.13) implies the same for the above product of matrices; that is, with $\bar{A}_m(z)= A_m(z^{-1})$, etc.,

$$\begin{bmatrix}\bar{A}_m & \bar{C}_m\\ \bar{B}_m & \bar{D}_m\end{bmatrix}\begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix}\begin{bmatrix}A_m & C_m\\ B_m & D_m\end{bmatrix} = \begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix}\sigma_m^2$$

which implies that the quantity $\bar{A}_m(z)A_m(z)-\bar{B}_m(z)B_m(z)$ is independent of z:

$$\bar{A}_m(z)A_m(z)-\bar{B}_m(z)B_m(z)= \sigma_m^2 \tag{12.13.20}$$

Property (12.13.14) implies that C_m and D_m are the reverse polynomials B_m^R and A_m^R, respectively; indeed

$$\begin{aligned}\begin{bmatrix}A_m^R & C_m^R\\ B_m^R & D_m^R\end{bmatrix} &= z^{-m}\begin{bmatrix}\bar{A}_m & \bar{C}_m\\ \bar{B}_m & \bar{D}_m\end{bmatrix} = z^{-m}z^{m/2}\sigma_m\bar{\psi}_m\cdots\bar{\psi}_1\bar{\psi}_0\\ &= z^{-m/2}\sigma_mJ_1(\psi_m\cdots\psi_0)J_1 = J_1\begin{bmatrix}A_m & C_m\\ B_m & D_m\end{bmatrix}J_1\\ &= \begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}\begin{bmatrix}A_m & C_m\\ B_m & D_m\end{bmatrix}\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix} = \begin{bmatrix}D_m & B_m\\ C_m & A_m\end{bmatrix}\end{aligned} \tag{12.13.21}$$

from which it follows that $C_m(z)= B_m^R(z)$ and $D_m(z)= A_m^R(z)$. The definition (12.13.18) implies also the recursion

$$\begin{bmatrix}A_m & B_m^R\\ B_m & A_m^R\end{bmatrix} = \begin{bmatrix}1 & \rho_mz^{-1}\\ \rho_m & z^{-1}\end{bmatrix}\begin{bmatrix}A_{m-1} & B_{m-1}^R\\ B_{m-1} & A_{m-1}^R\end{bmatrix}$$

Therefore each column of the *ABCD* matrix satisfies the same recursion. To summarize, we have

$$\begin{bmatrix}A_m(z) & B_m^R(z)\\ B_m(z) & A_m^R(z)\end{bmatrix} = \begin{bmatrix}1 & \rho_mz^{-1}\\ \rho_m & z^{-1}\end{bmatrix}\cdots\begin{bmatrix}1 & \rho_1z^{-1}\\ \rho_1 & z^{-1}\end{bmatrix}\begin{bmatrix}1 & \rho_0\\ \rho_0 & 1\end{bmatrix} \tag{12.13.22}$$

with the *lattice recursion*

$$\begin{bmatrix}A_m(z)\\ B_m(z)\end{bmatrix} = \begin{bmatrix}1 & \rho_mz^{-1}\\ \rho_m & z^{-1}\end{bmatrix}\begin{bmatrix}A_{m-1}(z)\\ B_{m-1}(z)\end{bmatrix} \tag{12.13.23}$$

and the property (12.13.20). The lattice recursion is initialized at $m = 0$ by:

$$A_0(z)= 1\,,\quad B_0(z)= \rho_0\,,\quad \text{or,}\quad \begin{bmatrix}A_0(z) & B_0^R(z)\\ B_0(z) & A_0^R(z)\end{bmatrix} = \begin{bmatrix}1 & \rho_0\\ \rho_0 & 1\end{bmatrix} \tag{12.13.24}$$

Furthermore, it follows from the lattice recursion (12.13.23) that the reflection coefficients ρ_m always appear in the *first* and *last* coefficients of the polynomials $A_m(z)$ and $B_m(z)$, as follows

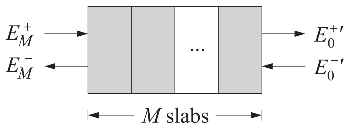
$$a_m(0)= 1\,,\quad a_m(m)= \rho_0\rho_m\,,\quad b_m(0)= \rho_m\,,\quad b_m(m)= \rho_0 \tag{12.13.25}$$

Eq. (12.13.17) for the field components reads now

$$\begin{bmatrix}E_m^+\\ E_m^-\end{bmatrix} = \frac{z^{m/2}}{\sigma_m}\begin{bmatrix}A_m & B_m^R\\ B_m & A_m^R\end{bmatrix}\begin{bmatrix}E_0^{+\prime}\\ E_0^{-\prime}\end{bmatrix}$$

Setting $m = M$, we find the relationship between the fields incident on the dielectric slab structure from the left to those incident from the right:

$$\begin{bmatrix}E_M^+\\ E_M^-\end{bmatrix} = \frac{z^{M/2}}{\sigma_M}\begin{bmatrix}A_M & B_M^R\\ B_M & A_M^R\end{bmatrix}\begin{bmatrix}E_0^{+\prime}\\ E_0^{-\prime}\end{bmatrix} \tag{12.13.26}$$



All the multiple reflections and reverberatory effects of the structure are buried in the transition matrix

$$\begin{bmatrix} A_M & B_M^R \\ B_M & A_M^R \end{bmatrix}$$

In reference to Fig. 12.13.2, the reflection and transmission responses R, T, R', T' of the structure can be obtained from Eq. (12.13.26) by noting that

$$\begin{bmatrix} 1 \\ R \end{bmatrix} = \frac{z^{M/2}}{\sigma_M}\begin{bmatrix} A_M & B_M^R \\ B_M & A_M^R \end{bmatrix}\begin{bmatrix} T \\ 0 \end{bmatrix}, \qquad \begin{bmatrix} 0 \\ T' \end{bmatrix} = \frac{z^{M/2}}{\sigma_M}\begin{bmatrix} A_M & B_M^R \\ B_M & A_M^R \end{bmatrix}\begin{bmatrix} R' \\ 1 \end{bmatrix}$$

which may be combined into one equation:

$$\begin{bmatrix} 1 & 0 \\ R & T' \end{bmatrix} = \frac{z^{M/2}}{\sigma_M}\begin{bmatrix} A_M & B_M^R \\ B_M & A_M^R \end{bmatrix}\begin{bmatrix} T & R' \\ 0 & 1 \end{bmatrix}$$

that can be written as follows:

$$\frac{z^{M/2}}{\sigma_M}\begin{bmatrix} A_M & B_M^R \\ B_M & A_M^R \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ R & T' \end{bmatrix}\begin{bmatrix} T & R' \\ 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} 1 & 0 \\ R & 1 \end{bmatrix}\begin{bmatrix} T^{-1} & 0 \\ 0 & T' \end{bmatrix}\begin{bmatrix} 1 & -R' \\ 0 & 1 \end{bmatrix}$$

Solving these for the reflection and transmission responses, we find:

$$\begin{aligned} R(z) &= \frac{B_M(z)}{A_M(z)}, \qquad & T(z) &= \frac{\sigma_M z^{-M/2}}{A_M(z)} \\ R'(z) &= -\frac{B_M^R(z)}{A_M(z)}, \qquad & T'(z) &= \frac{\sigma_M z^{-M/2}}{A_M(z)} \end{aligned} \tag{12.13.27}$$

Note that $T(z)=T'(z)$. Since on physical grounds the transmission response $T(z)$ must be a *stable and causal* z-transform, it follows that necessarily the polynomial $A_M(z)$ must be a *minimum-phase polynomial*. The overall delay factor $z^{-M/2}$ in $T(z)$ is of no consequence. It just means that before anything can be transmitted through the structure, it must traverse all M slabs, each with a travel time delay of T_1 seconds; that is, with overall delay of MT_1 seconds.

Let $R_{m-1}(z)$ and $T_{m-1}(z)$ be the reflection and transmission responses based on $m-1$ layers. The addition of one more layer will change the responses to $R_m(z)$ and $T_m(z)$. Using the lattice recursions, we may derive a recursion for these responses:

$$R_m(z) = \frac{B_m(z)}{A_m(z)} = \frac{\rho_m A_{m-1}(z) + z^{-1}B_{m-1}(z)}{A_{m-1}(z) + \rho_m z^{-1}B_{m-1}(z)}$$

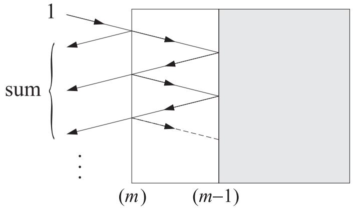
Dividing numerator and denominator by $A_{m-1}(z)$ we obtain

$$R_m(z) = \frac{\rho_m + z^{-1}R_{m-1}(z)}{1 + \rho_m z^{-1}R_{m-1}(z)} \tag{12.13.28}$$

It describes the effect of adding a layer. Expanding it in a power series, we have

$$R_m(z) = \rho_m + (1-\rho_m^2)\big[z^{-1}R_{m-1}(z)\big] - (1-\rho_m^2)\rho_m\big[z^{-1}R_{m-1}(z)\big]^2 + \cdots$$

It can be verified easily that the various terms in this sum correspond to the multiple reflections taking place within the mth layer, as shown below:



The first term in the expansion is always ρ_m; that is, $\rho_m = R_m(\infty)$. Thus, from the knowledge of $R_m(z)$ we may extract ρ_m. With ρ_m known, we may invert Eq. (12.13.28) to get $R_{m-1}(z)$ from which we can extract ρ_{m-1}; and so on, we may extract the series of reflection coefficients. The inverse of Eq. (12.13.28), which describes the effect of removing a layer, is

$$R_{m-1}(z) = z\,\frac{R_m(z)-\rho_m}{1-\rho_m R_m(z)} \tag{12.13.29}$$

Up to a difference in the sign of ρ_m, this is recognized as the Schur recursion (12.10.25). It provides a nice physical interpretation of that recursion; namely, the Schur functions represent the overall reflection responses at the successive layer interfaces, which on physical grounds must be stable, causal, and bounded $|R_m(z)| \leq 1$ for all z in their region of convergence that includes, at least, the unit circle and all the points outside it. We may also derive a recursion for the transmission responses, which requires the simultaneous recursion of $R_m(z)$:

$$T_m(z) = \frac{t_m z^{-1/2}T_{m-1}(z)}{1+\rho_m z^{-1}R_{m-1}(z)}, \quad T_{m-1}(z) = z^{1/2}\,\frac{t_m T_m(z)}{1-\rho_m R_m(z)} \tag{12.13.30}$$

The dynamic predictive deconvolution method is an alternative method of extracting the sequence of reflection coefficients and is discussed below.

The equations (12.13.27) for the scattering responses R, T, R', T' imply the *unitarity* of the scattering matrix S given by

$$S = \begin{bmatrix} T & R' \\ R & T' \end{bmatrix}$$

that is,

$$\bar{S}(z)^T S(z) = S(z^{-1})^T S(z) = I \tag{12.13.31}$$

where I is the 2×2 unit matrix. On the unit circle $z = e^{j\omega T_2}$ the scattering matrix becomes a unitary matrix: $S(\omega)^\dagger S(\omega) = I$. Component-wise, Eq. (12.13.31) becomes

$$\bar{T}T + \bar{R}R = \bar{T}'T' + \bar{R}'R' = 1, \quad \bar{T}R' + \bar{R}T' = 0 \tag{12.13.32}$$

Robinson and Treitel's *dynamic predictive deconvolution method* [974] of solving the inverse scattering problem is based on the above unitarity equation. In the inverse problem, it is required to extract the set of reflection coefficients from measurements of either the reflection response R or the transmission response T. In speech processing it is the transmission response that is available. In geophysical applications, or in studying the reflectivity properties of thin films, it is the reflection response that is available. The problem of designing terminations of transmission lines also falls in the latter category. In this case, an appropriate termination is desired that must have a specified reflection response $R(z)$; for example, to be reflectionless over a wide band of frequencies about some operating frequency.

The solution of both types of problems follows the same steps. First, from the knowledge of the reflection response $R(z)$, or the transmission response $T(z)$, the *spectral function* of the structure is defined:

$$\Phi(z)=1-R(z)\bar{R}(z)=T(z)\bar{T}(z)=\frac{\sigma_M^2}{A_M(z)\bar{A}_M(z)} \tag{12.13.33}$$

This is recognized as the *power spectrum* of the transmission response, and it is of the autoregressive type. Thus, linear prediction methods can be used in the solution.

In the time domain, the autocorrelation lags $\phi(k)$ of the spectral function are obtained from the sample autocorrelations of the reflection sequence, or the transmission sequence:

$$\phi(k)=\delta(k)-C(k)=D(k) \tag{12.13.34}$$

where $C(k)$ and $D(k)$ are the sample autocorrelations of the reflection and transmission time responses:

$$C(k)=\sum_n R(n+k)R(n)\,,\quad D(k)=\sum_n T(n+k)T(n) \tag{12.13.35}$$

In practice, only a finite record of the reflection (or transmission) sequence will be available, say $\{R(0),R(1),\dots,R(N-1)\}$. Then, an approximation to $C(k)$ must be used, as follows:

$$C(k)=\sum_{n=0}^{N-1-k}R(n+k)R(n)\,,\quad k=0,1,\dots,M \tag{12.13.36}$$

The polynomial $A_M(z)$ may be recovered from the knowledge of the first M lags of the spectral function; that is, $\{\phi(0),\phi(1),\dots,\phi(M)\}$. The determining equations for the coefficients of $A_M(z)$ are precisely the normal equations of linear prediction. In the present context, they may be derived directly by noting that $\Phi(z)$ is a stable spectral density and is already factored into its minimum-phase factors in Eq. (12.13.33). Thus, writing

$$\Phi(z)A_M(z)=\frac{\sigma_M^2}{A_M(z^{-1})}$$

it follows that the right-hand side is expandable in positive powers of z; the negative powers of z in the left-hand side must be set equal to zero. This gives the normal equations:

$$\begin{bmatrix}\phi(0) & \phi(1) & \phi(2) & \cdots & \phi(M)\\ \phi(1) & \phi(0) & \phi(1) & \cdots & \phi(M-1)\\ \phi(2) & \phi(1) & \phi(0) & \cdots & \phi(M-2)\\ \vdots & \vdots & \vdots & & \vdots\\ \phi(M) & \phi(M-1) & \phi(M-2) & \cdots & \phi(0)\end{bmatrix}\begin{bmatrix}1\\ a_M(1)\\ a_M(2)\\ \vdots\\ a_M(M)\end{bmatrix}=\begin{bmatrix}\sigma_M^2\\ 0\\ 0\\ \vdots\\ 0\end{bmatrix} \tag{12.13.37}$$

which can be solved efficiently using Levinson's algorithm. Having obtained $A_M(z)$ and noting the $B_M(z)=A_M(z)R(z)$, the coefficients of the polynomial $B_M(z)$ may be recovered by convolution:

$$b_M(n)=\sum_{m=0}^{n}a_M(n-m)R(m)\,,\quad n=0,1,\dots,M \tag{12.13.38}$$

Having obtained both $A_M(z)$ and $B_M(z)$ and noting that $\rho_M=b_M(0)$, the lattice recursion (12.13.23) may be inverted to recover the polynomials $A_{M-1}(z)$ and $B_{M-1}(z)$ as well as the next reflection coefficient $\rho_{M-1}=b_{M-1}(0)$, and so on. The inverse of the lattice recursion matrix is

$$\begin{bmatrix}1 & \rho_m z^{-1}\\ \rho_m & z^{-1}\end{bmatrix}^{-1}=\frac{1}{1-\rho_m^2}\begin{bmatrix}1 & -\rho_m\\ -\rho_m z & z\end{bmatrix}$$

Therefore, the *backward* recursion becomes:

$$\rho_m=b_m(0)\,,\quad \begin{bmatrix}A_{m-1}(z)\\ B_{m-1}(z)\end{bmatrix}=\frac{1}{1-\rho_m^2}\begin{bmatrix}1 & -\rho_m\\ -\rho_m z & z\end{bmatrix}\begin{bmatrix}A_m(z)\\ B_m(z)\end{bmatrix} \tag{12.13.39}$$

In this manner, all the reflection coefficients $\{\rho_0,\rho_1,\dots,\rho_M\}$ can be extracted. The computational algorithm is summarized as follows:

1. Measure $R(0),R(1),\dots,R(N-1)$.
2. Select a reasonable value for the number of slabs M.
3. Compute the $M+1$ sample autocorrelation lags $C(0),C(1),\dots,C(M)$ of the reflection response $R(n)$, using Eq. (12.13.36).
4. Compute $\phi(k)=\delta(k)-C(k)$, $k=0,1,\dots,M$.
5. Using Levinson's algorithm, solve the normal equations (12.13.37) for the coefficients of $A_M(z)$.
6. Convolve $A_M(z)$ with $R(z)$ to find $B_M(z)$.
7. Compute $\rho_M=b_M(0)$ and iterate the backward recursion (12.13.39) from $m=M$ down to $m=0$.

The function **dpd** is an implementation of the dynamic predictive deconvolution procedure. The inputs to the function are N samples of the reflection response $\{R(0),R(1),\dots,R(N-1)\}$ and the number of layers M. The outputs are the lattice polynomials $A_i(z)$ and

$B_i(z)$, for $i = 0, 1, \dots, M$, arranged in the two lower-triangular matrices A and B whose rows hold the coefficients of these polynomials; that is, $A(i,j) = a_i(j)$, or

$$A_i(z) = \sum_{j=0}^{i} A(i,j)\, z^{-j}$$

and similarly for $B_i(z)$. The function invokes the function **lev** to solve the normal equations (12.13.34). The *forward scattering problem* is implemented by the function **scatt**, whose inputs are the set of reflection coefficients $\{\rho_0, \rho_1, \dots, \rho_M\}$ and whose outputs are the lattice polynomials $A_i(z)$ and $B_i(z)$, for $i = 0, 1, \dots, M$, as well as a pre-specified number N of reflection response samples $\{R(0), R(1), \dots, R(N-1)\}$. It utilizes the forward lattice recursion (12.13.23) to obtain the lattice polynomials, and then computes the reflection response samples by taking the inverse z-transform of Eq. (12.13.27).

Next, we present a number of deconvolution examples simulated by means of the functions **scatter** and **dpd**. In each case, we specified the five reflection coefficients of a structure consisting of four layers. Using **scatter** we generated the exact lattice polynomials whose coefficients are arranged in the matrices A and B, and also generated 16 samples of the reflection response $R(n)$, $n = 0, 1, \dots, 15$. These 16 samples were sent through the **dpd** function to extract the lattice polynomials A and B.

The first figure of each example displays a table of the reflection response samples and the exact and extracted polynomials. Note that the first column of the matrix B is the vector of reflection coefficients, according to Eq. (12.13.25). The remaining two graphs of each example show the reflection response R in the time domain and in the frequency domain. Note that the frequency response is plotted only over one Nyquist interval $[0, 2\pi/T_2]$, and it is symmetric about the Nyquist frequency π/T_2.

Figs. 12.13.3 and 12.13.4 correspond to the case of equal reflection coefficients $\{\rho_0, \rho_1, \rho_2, \rho_3, \rho_4\} = \{0.5, 0.5, 0.5, 0.5, 0.5\}$.

In Figs. 12.13.5 and 12.13.6 the reflection coefficients have been tapered somewhat at the ends (windowed) and are $\{0.3, 0.4, 0.5, 0.4, 0.3\}$. Note the effect of tapering on the lobes of the reflection frequency response. Figs. 12.13.7 and 12.13.8 correspond to the set of reflection coefficients $\{0.1, 0.2, 0.3, 0.2, 0.1\}$. Note the broad band of frequencies about the Nyquist frequency for which there is very little reflection. In contrast, the example in Figs. 12.13.9 and 12.13.10 exhibits high reflectivity over a broad band of frequencies about the Nyquist frequency. Its set of reflection coefficients is $\{0.5, -0.5, 0.5, -0.5, 0.5\}$.

In this section we have discussed the inverse problem of unraveling the structure of a medium from the knowledge of its reflection response. The connection of the dynamic predictive deconvolution method to the conventional inverse scattering methods based on the *Gelfand-Levitan-Marchenko* approach [1054] has been discussed in [1043,1055,1056].‘The lattice recursions characteristic of the wave propagation problem were derived as a direct consequence of the boundary conditions at the interfaces between media, whereas the lattice recursions of linear prediction were a direct consequence of the Gram-Schmidt orthogonalization process and the minimization of the prediction-error performance index. Is there a deeper connection between these two problems [1005–1007]? One notable result in this direction has been to show that the

$$A_{\text{exact}} = \begin{bmatrix} 1.0000 & 0 & 0 & 0 & 0 \\ 1.0000 & 0.2500 & 0 & 0 & 0 \\ 1.0000 & 0.5000 & 0.2500 & 0 & 0 \\ 1.0000 & 0.7500 & 0.5625 & 0.2500 & 0 \\ 1.0000 & 1.0000 & 0.9375 & 0.6250 & 0.2500 \end{bmatrix}$$

$$A_{\text{extract}} = \begin{bmatrix} 1.0000 & 0 & 0 & 0 & 0 \\ 1.0000 & 0.2509 & 0 & 0 & 0 \\ 1.0000 & 0.5009 & 0.2510 & 0 & 0 \\ 1.0000 & 0.7509 & 0.5638 & 0.2508 & 0 \\ 1.0000 & 1.0009 & 0.9390 & 0.6263 & 0.2504 \end{bmatrix}$$

$$B_{\text{exact}} = \begin{bmatrix} 0.5000 & 0 & 0 & 0 & 0 \\ 0.5000 & 0.5000 & 0 & 0 & 0 \\ 0.5000 & 0.6250 & 0.5000 & 0 & 0 \\ 0.5000 & 0.7500 & 0.7500 & 0.5000 & 0 \\ 0.5000 & 0.8750 & 1.0313 & 0.8750 & 0.5000 \end{bmatrix}$$

$$B_{\text{extract}} = \begin{bmatrix} 0.5010 & 0 & 0 & 0 & 0 \\ 0.5000 & 0.5010 & 0 & 0 & 0 \\ 0.5000 & 0.6255 & 0.5010 & 0 & 0 \\ 0.5000 & 0.7505 & 0.7510 & 0.5010 & 0 \\ 0.5000 & 0.8755 & 1.0323 & 0.8764 & 0.5010 \end{bmatrix}$$

k	$R(k)$
0	0.5000
1	0.3750
2	0.1875
3	0.0234
4	−0.0586
5	−0.1743
6	0.1677
7	0.0265
8	−0.0601
9	−0.0259
10	0.0238
11	0.0314
12	−0.0225
13	−0.0153
14	0.0109
15	0.0097

Fig. 12.13.3 Reflection response and lattice polynomials.

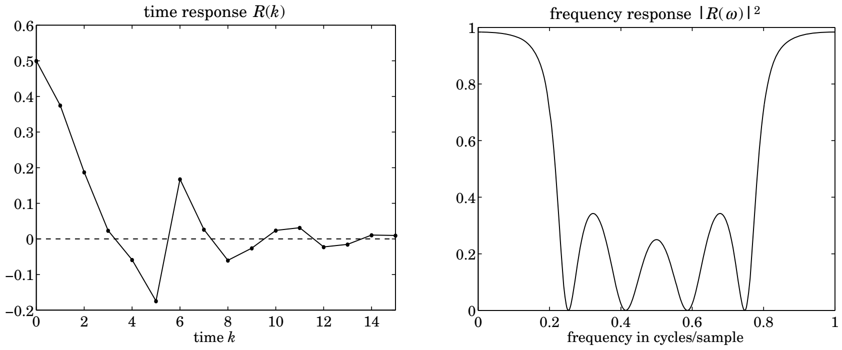


Fig. 12.13.4 Reflection responses in the time and frequency domains.

Cholesky factorization of Toeplitz or near-Toeplitz matrices via the Schur algorithm can be cast in a wave propagation model and derived as a simple consequence of energy conservation [1002].

$$A_{\text{exact}} = \begin{bmatrix} 1.0000 & 0 & 0 & 0 & 0 \\ 1.0000 & 0.1200 & 0 & 0 & 0 \\ 1.0000 & 0.3200 & 0.1500 & 0 & 0 \\ 1.0000 & 0.5200 & 0.3340 & 0.1200 & 0 \\ 1.0000 & 0.6400 & 0.5224 & 0.2760 & 0.0900 \end{bmatrix}$$

$$A_{\text{extract}} = \begin{bmatrix} 1.0000 & 0 & 0 & 0 & 0 \\ 1.0000 & 0.1200 & 0 & 0 & 0 \\ 1.0000 & 0.3200 & 0.1500 & 0 & 0 \\ 1.0000 & 0.5200 & 0.3340 & 0.1200 & 0 \\ 1.0000 & 0.6400 & 0.5224 & 0.2760 & 0.0900 \end{bmatrix}$$

$$B_{\text{exact}} = \begin{bmatrix} 0.3000 & 0 & 0 & 0 & 0 \\ 0.4000 & 0.3000 & 0 & 0 & 0 \\ 0.5000 & 0.4600 & 0.3000 & 0 & 0 \\ 0.4000 & 0.6280 & 0.5200 & 0.3000 & 0 \\ 0.3000 & 0.5560 & 0.7282 & 0.5560 & 0.3000 \end{bmatrix}$$

$$B_{\text{extract}} = \begin{bmatrix} 0.3000 & 0 & 0 & 0 & 0 \\ 0.4000 & 0.3000 & 0 & 0 & 0 \\ 0.5000 & 0.4600 & 0.3000 & 0 & 0 \\ 0.4000 & 0.6280 & 0.5200 & 0.3000 & 0 \\ 0.3000 & 0.5560 & 0.7282 & 0.5560 & 0.3000 \end{bmatrix}$$

k	$R(k)$
0	0.3000
1	0.3640
2	0.3385
3	0.0664
4	−0.0468
5	−0.1309
6	0.0594
7	0.0373
8	−0.0146
9	−0.0148
10	0.0014
11	0.0075
12	−0.0001
13	−0.0029
14	−0.0003
15	0.0010

Fig. 12.13.5 Reflection response and lattice polynomials.

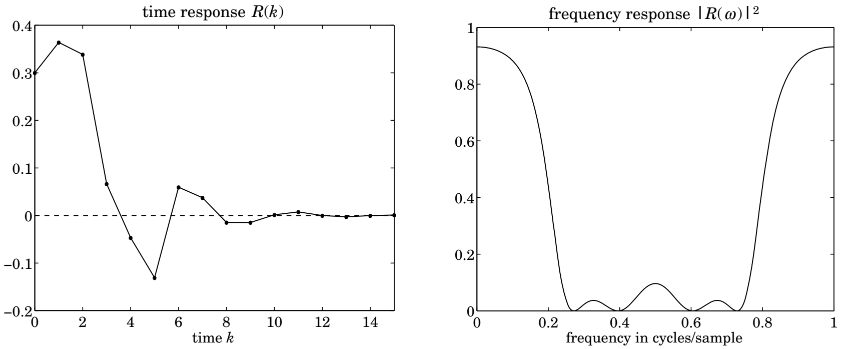


Fig. 12.13.6 Reflection responses in the time and frequency domains.

$$A_{\text{exact}} = \begin{bmatrix} 1.0000 & 0 & 0 & 0 & 0 \\ 1.0000 & 0.0200 & 0 & 0 & 0 \\ 1.0000 & 0.0800 & 0.0300 & 0 & 0 \\ 1.0000 & 0.1400 & 0.0712 & 0.0200 & 0 \\ 1.0000 & 0.1600 & 0.1028 & 0.0412 & 0.0100 \end{bmatrix}$$

$$A_{\text{extract}} = \begin{bmatrix} 1.0000 & 0 & 0 & 0 & 0 \\ 1.0000 & 0.0200 & 0 & 0 & 0 \\ 1.0000 & 0.0800 & 0.0300 & 0 & 0 \\ 1.0000 & 0.1400 & 0.0712 & 0.0200 & 0 \\ 1.0000 & 0.1600 & 0.1028 & 0.0412 & 0.0100 \end{bmatrix}$$

$$B_{\text{exact}} = \begin{bmatrix} 0.1000 & 0 & 0 & 0 & 0 \\ 0.2000 & 0.1000 & 0 & 0 & 0 \\ 0.3000 & 0.2060 & 0.1000 & 0 & 0 \\ 0.2000 & 0.3160 & 0.2120 & 0.1000 & 0 \\ 0.1000 & 0.2140 & 0.3231 & 0.2140 & 0.1000 \end{bmatrix}$$

$$B_{\text{extract}} = \begin{bmatrix} 0.1000 & 0 & 0 & 0 & 0 \\ 0.2000 & 0.1000 & 0 & 0 & 0 \\ 0.3000 & 0.2060 & 0.1000 & 0 & 0 \\ 0.2000 & 0.3160 & 0.2120 & 0.1000 & 0 \\ 0.1000 & 0.2140 & 0.3231 & 0.2140 & 0.1000 \end{bmatrix}$$

k	$R(k)$
0	0.1000
1	0.1980
2	0.2812
3	0.1445
4	0.0388
5	−0.0346
6	−0.0072
7	0.0017
8	0.0015
9	0.0002
10	−0.0002
11	−0.0001
12	0.0000
13	0.0000
14	0.0000
15	−0.0000

Fig. 12.13.7 Reflection response and lattice polynomials.

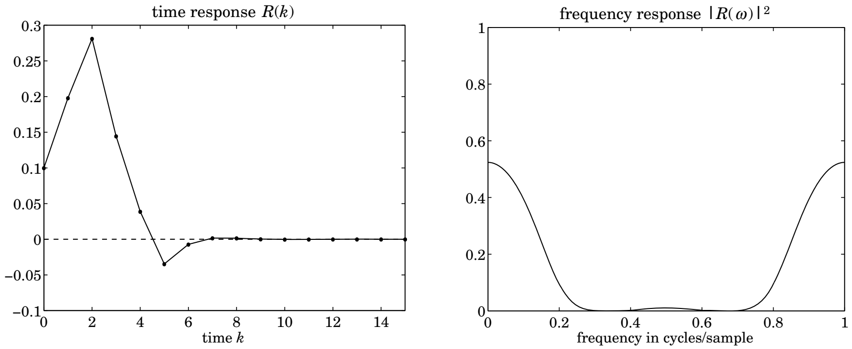


Fig. 12.13.8 Reflection responses in the time and frequency domains.

$$A_{\text{exact}} = \begin{bmatrix} 1.0000 & 0 & 0 & 0 & 0 \\ 1.0000 & -0.2500 & 0 & 0 & 0 \\ 1.0000 & -0.5000 & 0.2500 & 0 & 0 \\ 1.0000 & -0.7500 & 0.5625 & -0.2500 & 0 \\ 1.0000 & -1.0000 & 0.9375 & -0.6250 & 0.2500 \end{bmatrix}$$

$$A_{\text{extract}} = \begin{bmatrix} 1.0000 & 0 & 0 & 0 & 0 \\ 1.0000 & -0.2509 & 0 & 0 & 0 \\ 1.0000 & -0.5009 & 0.2510 & 0 & 0 \\ 1.0000 & -0.7509 & 0.5638 & -0.2508 & 0 \\ 1.0000 & -1.0009 & 0.9390 & -0.6263 & 0.2504 \end{bmatrix}$$

$$B_{\text{exact}} = \begin{bmatrix} 0.5000 & 0 & 0 & 0 & 0 \\ -0.5000 & 0.5000 & 0 & 0 & 0 \\ 0.5000 & -0.6250 & 0.5000 & 0 & 0 \\ -0.5000 & 0.7500 & -0.7500 & 0.5000 & 0 \\ 0.5000 & -0.8750 & 1.0313 & -0.8750 & 0.5000 \end{bmatrix}$$

$$B_{\text{extract}} = \begin{bmatrix} 0.5010 & 0 & 0 & 0 & 0 \\ -0.5000 & 0.5010 & 0 & 0 & 0 \\ 0.5000 & -0.6255 & 0.5010 & 0 & 0 \\ -0.5000 & 0.7505 & -0.7510 & 0.5010 & 0 \\ 0.5000 & -0.8755 & 1.0323 & -0.8764 & 0.5010 \end{bmatrix}$$

k	$R(k)$
0	0.5000
1	−0.3750
2	0.1875
3	−0.0234
4	−0.0586
5	0.1743
6	0.1677
7	−0.0265
8	−0.0601
9	0.0259
10	0.0238
11	−0.0314
12	−0.0225
13	0.0153
14	0.0109
15	−0.0097

Fig. 12.13.9 Reflection response and lattice polynomials.

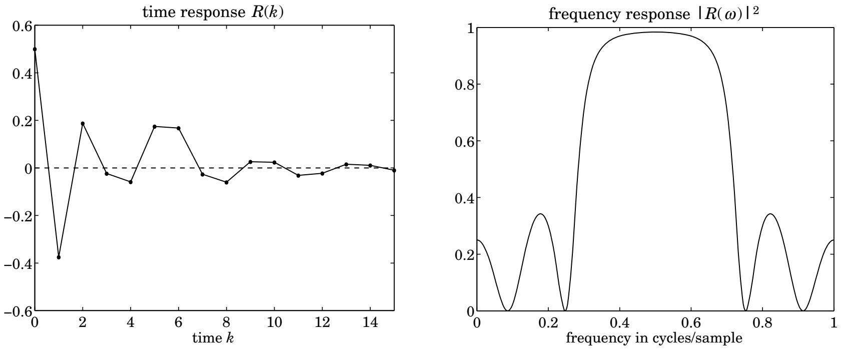


Fig. 12.13.10 Reflection responses in the time and frequency domains.

12.14 Least-Squares Waveshaping and Spiking Filters

In linear prediction, the three practical methods of estimating the prediction error filter coefficients were all based on replacing the ensemble mean-square minimization criterion by a least-squares criterion based on time averages. Similarly, the more general Wiener filtering problem may be recast in terms of such time averages. A practical formulation, which is analogous to the Yule-Walker or autocorrelation method, is as follows [974,975,1010,1059]. Given a record of available data

$$\boxed{y_0, y_1, \dots, y_N}$$

find the best linear FIR filter of order M

$$\boxed{h_0, h_1, \dots, h_M}$$

which reshapes y_n into a desired signal x_n, specified in terms of the samples:

$$\boxed{x_0, x_1, \dots, x_{N+M}}$$

where for consistency of convolution, we assumed we know $N+M+1$ samples of the desired signal. The actual convolution output of the waveshaping filter will be:

$$\hat{x}_n = \sum_{m=\max(0,n-N)}^{\min(n,M)} h_m x_{n-m}\,, \quad 0 \le n \le N+M \tag{12.14.1}$$

and the estimation error:

$$e_n = x_n - \hat{x}_n\,, \quad 0 \le n \le N+M \tag{12.14.2}$$

As the optimality criterion, we choose the *least-squares* criterion:

$$\mathcal{E} = \sum_{n=0}^{N+M} e_n^2 = \min \tag{12.14.3}$$

The optimal filter weights h_m are selected to minimize $\mathcal{E}$. It is convenient to recast the above in a compact matrix form. Define the $(N+M+1)\times(M+1)$ convolution data matrix Y, the $(M+1)\times 1$ vector of filter weights $\mathbf{h}$, the $(N+M+1)\times 1$ vector of desired samples $\mathbf{x}$, (and estimates $\hat{\mathbf{x}}$ and estimation errors $\mathbf{e}$), as follows:

$$Y = \begin{bmatrix} y_0 & 0 & 0 & \cdots & 0 \\ y_1 & y_0 & 0 & \cdots & 0 \\ y_2 & y_1 & y_0 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ y_N & y_{N-1} & y_{N-2} & \cdots & y_{N-M} \\ 0 & y_N & y_{N-1} & \cdots & y_{N-M+1} \\ 0 & 0 & y_N & \cdots & y_{N-M+2} \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & y_N \end{bmatrix}, \quad \mathbf{h} = \begin{bmatrix} h_0 \\ h_1 \\ \vdots \\ h_M \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N+M} \end{bmatrix} \tag{12.14.4}$$

Equations (12.14.1) through (12.14.3) now become

$$\hat{\mathbf{x}} = Y\mathbf{h}\,,\quad \mathbf{e} = \mathbf{x} - \hat{\mathbf{x}}\,,\quad \mathcal{E} = \mathbf{e}^T\mathbf{e} \tag{12.14.5}$$

Minimizing $\mathcal{E}$ with respect to the weight vector $\mathbf{h}$ results in the *orthogonality* equations:

$$Y^T\mathbf{e} = Y^T(\mathbf{x} - Y\mathbf{h}) = 0 \tag{12.14.6}$$

which are equivalent to the *normal* equations:

$$Y^TY\mathbf{h} = Y^T\mathbf{x} \tag{12.14.7}$$

Solving for $\mathbf{h}$, we find

$$\mathbf{h} = (Y^TY)^{-1}Y^T\mathbf{x} = R^{-1}\mathbf{r} \tag{12.14.8}$$

where the quantities

$$R = Y^TY\,,\quad \mathbf{r} = Y^T\mathbf{x} \tag{12.14.9}$$

may be recognized (see Sec. 1.11) as the $(M+1)\times(M+1)$ autocorrelation matrix formed by the sample autocorrelations $\hat{R}_{yy}(0), \hat{R}_{yy}(1), \dots \hat{R}_{yy}(M)$ of y_n, and as the $(M+1)\times 1$ vector of sample cross-correlations $\hat{R}_{xy}(0), \hat{R}_{xy}(1), \dots \hat{R}_{xy}(M)$ between the desired and the available vectors x_n and y_n. We have already used this expression for the weight vector $\mathbf{h}$ in the example of Sec. 12.11. Here we have justified it in terms of the least-squares criterion (12.14.3). The function **firw** may be used to solve for the weights (12.14.8) and, if so desired, to give the corresponding lattice realization. The actual filter output $\hat{\mathbf{x}}$ is expressed as

$$\hat{\mathbf{x}} = Y\mathbf{h} = YR^{-1}Y^T\mathbf{x} = P\mathbf{x} \tag{12.14.10}$$

where

$$P = YR^{-1}Y^T = Y(Y^TY)^{-1}Y^T \tag{12.14.11}$$

The error vector becomes $\mathbf{e} = (I - P)\mathbf{x}$. The "performance" matrix P is a projection matrix, and thus, so is $(I - P)$. Then, the error square becomes

$$\mathcal{E} = \mathbf{e}^T\mathbf{e} = \mathbf{x}^T(I-P)^2\mathbf{x} = \mathbf{x}^T(I-P)\mathbf{x} \tag{12.14.12}$$

The $(N+M+1)\times(N+M+1)$ matrix P has trace equal to $M+1$, as can be checked easily. Since its eigenvalues as a projection matrix are either 0 or 1, it follows that in order for the sum of all the eigenvalues (the trace) to be equal to $M+1$, there must necessarily be $M+1$ eigenvalues that are equal to 1, and N eigenvalues equal to 0. Therefore, the matrix P has rank $M+1$, and if the desired vector $\mathbf{x}$ is selected to be any of the $M+1$ eigenvectors belonging to eigenvalue 1, the corresponding estimation error will be zero.

Among all possible waveshapes that may be chosen for the desired vector $\mathbf{x}$, of particular importance are the *spikes*, or impulses. In this case, $\mathbf{x}$ is a unit impulse, say at the origin; that is, $x_n = \delta_n$. The convolution $\hat{x}_n = h_n * y_n$ of the corresponding filter with y_n is the best least-squares approximation to the unit impulse. In other words, h_n is the best least-squares inverse filter to y_n that attempts to reshape, or compress, y_n into a unit impulse. Such least squares inverse filters are used extensively in deconvolution applications. More generally. the vector $\mathbf{x}$ may be chosen to be any one of the unit vectors

$$\mathbf{x} = \mathbf{u}_i = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \begin{matrix} \\ \\ \\ \leftarrow\ i\text{th slot}\,, \quad i = 0, 1, \dots, N+M \\ \\ \\ \\ \end{matrix} \tag{12.14.13}$$

which corresponds to a unit impulse occurring at the ith time instant instead of at the origin; that is, $x_n = \delta(n-i)$. The actual output from the spiking filter is given by

$$\hat{\mathbf{x}} = P\mathbf{x} = P\mathbf{u}_i = i\text{th column of } P \tag{12.14.14}$$

Thus, the ith column of the matrix P is the output of the ith spiking filter which attempts to compress y_n into a spike with i delays. The corresponding ith filter is $\mathbf{h} = R^{-1}Y^T\mathbf{u}_i$. Therefore, the columns of the matrix

$$H = R^{-1}Y^T = (Y^TY)^{-1}Y^T \tag{12.14.15}$$

are all the optimal spiking filters. The estimation error of the ith filter is

$$\mathcal{E}_i = \mathbf{u}_i^T(I-P)\mathbf{u}_i = 1 - P_{ii} \tag{12.14.16}$$

where P_{ii}, is the ith diagonal element of P. Since the delay i may be positioned anywhere from $i = 0$ to $i = N+M$, there are $N+M+1$ such spiking filters, each with error $\mathcal{E}_i$. Among these, there will be one that has the optimal delay i which corresponds to the smallest of the $\mathcal{E}_i$s; or, equivalently, to the maximum of the diagonal elements P_{ii}.

The design procedure for least-squares spiking filters for a given finite signal y_n, $n = 0, 1, \dots, N-1$ is summarized as follows:

1. Compute $R = Y^TY$.
2. Compute the inverse R^{-1} (preferably by the Levinson recursion).
3. Compute $H = R^{-1}Y^T$ = all the spiking filters.
4. Compute $P = YH = YR^{-1}Y^T$ = all spiking filter outputs.
5. Select that column i of P for which P_{ii} is the largest.

If the Levinson-Cholesky algorithm is used to compute the inverse R^{-1}, this design procedure becomes fairly efficient. An implementation of the procedure is given by the function **spike**. The inputs to the function are the $N+1$ samples $\{y_0, y_1, \dots, y_N\}$, the desired order M of the spiking filter, and a so-called "prewhitening" or Backus-Gilbert parameter ϵ, which will be explained below. The outputs of the function are the matrices P and H.

To explain the role of the parameter ϵ, let us go back to the waveshaping problem. When the data sequence y_n to be reshaped into x_n is inaccurately known—if, for example, it has been contaminated by white noise v_n—the least-squares minimization criterion

(12.14.3) can be extended slightly to accomplish the double task of (1) producing the best estimate of x_n and (2) reducing the noise at the output of the filter h_n as much as possible.

The input to the filter is the noisy sequence $y_n + v_n$ and its output is $h_n * y_n + h_n * v_n = \hat{x}_n + u_n$, where we set $u_n = h_n * v_n$. The term u_n represents the filtered noise. The minimization criterion (12.14.3) may be replaced by

$$\mathcal{E} = \sum_n e_n^2 + \lambda E[u_n^2] = \min \tag{12.14.17}$$

where λ is a positive parameter which can be chosen by the user. Large λ emphasizes large reduction of the output noise, but this is done at the expense of resolution; that is, at the expense of obtaining a very good estimate. On the other hand, small λ emphasizes higher resolution but with lesser noise reduction. This tradeoff between resolution and noise reduction is the basic property of this performance index. Assuming that v_n is white with variance σ_v^2, we have

$$E[u_n^2] = \sigma_v^2 \sum_{n=0}^{M} h_n^2 = \sigma_v^2 \mathbf{h}^T\mathbf{h}$$

Thus, Eq. (12.14.17) may be written as

$$\mathcal{E} = \mathbf{e}^T\mathbf{e} + \lambda\sigma_v^2 \mathbf{h}^T\mathbf{h} = \min \tag{12.14.18}$$

Its minimization with respect to $\mathbf{h}$ gives the normal equations:

$$(Y^TY + \lambda\sigma_v^2 I)\mathbf{h} = Y^T\mathbf{x} \tag{12.14.19}$$

from which it is evident that the diagonal of Y^TY is shifted by an amount $\lambda\sigma_v^2$; that is,

$$\hat{R}_{yy}(0) \longrightarrow \hat{R}_{yy}(0) + \lambda\sigma_v^2 \equiv (1+\epsilon)\hat{R}_{yy}(0)\,, \quad \epsilon = \frac{\lambda\sigma_v^2}{\hat{R}_{yy}(0)}$$

In practice, ϵ may be taken to be a few percent or less. It is evident from Eq. (12.14.19) that one beneficial effect of the parameter ϵ is the stabilization of the inverse of the matrix $Y^TY + \lambda\sigma_v^2 I$.

The main usage of spiking filters is in deconvolution problems [59,60,95,144–146], where the desired and the available signals x_n and y_n are related to each other by the convolutional relationship

$$y_n = f_n * x_n = \sum_m f_m x_{n-m} \tag{12.14.20}$$

where f_n is a "blurring" function which is assumed to be approximately known. The basic deconvolution problem is to recover x_n from y_n if f_n is known. For example, y_n may represent the image of an object x_n recorded through an optical system with a point-spread function f_n. Or, y_n might represent the recorded seismic trace arising from the excitation of the layered earth by an impulsive waveform f_n (the source wavelet) which is convolved with the reflection impulse response x_n of the earth (in the previous section x_n was denoted by R_n.) If the effect of the source wavelet f_n can be "deconvolved away," the resulting reflection sequence x_n may be subjected to the dynamic predictive deconvolution procedure to unravel the earth structure. Or, f_n may represent the impulse response of a channel, or a magnetic recording medium, which broadens and blurs (intersymbol interference) the desired message x_n.

The least-squares inverse spiking filters offer a way to solve the deconvolution problem: Simply design a least-squares spiking filter h_n corresponding to the blurring function f_n; that is, $h_n * f_n \simeq \delta_n$, in the least-squares sense. Then, filtering y_n through h_n will recover the desired signal x_n:

$$\hat{x}_n = h_n * y_n = (h_n * f_n) * x_n \simeq \delta_n * x_n = x_n \tag{12.14.21}$$

If the ith spiking filter is used, which compresses f_n into an impulse with i delays, $h_n * f_n \simeq \delta(n-i)$, then the desired signal x_n will be recovered with a delay of i units of time.

This and all other approaches to deconvolution work well when the data y_n are not noisy. In presence of noise, Eq. (12.14.20) becomes

$$y_n = f_n * x_n + v_n \tag{12.14.22}$$

where v_n may be assumed to be zero-mean white noise of variance σ_v^2. Even if the blurring function f_n is known exactly and a good least-squares inverse filter h_n can be designed, the presence of the noise term can distort the deconvolved signal beyond recognition. This may be explained as follows. Filtering y_n through the inverse filter h_n results in

$$h_n * y_n = (h_n * f_n) * x_n + h_n * v_n \simeq x_n + u_n$$

where $u_n = h_n * v_n$ is the filtered noise. Its variance is

$$E[u_n^2] = \sigma_v^2 \mathbf{h}^T\mathbf{h} = \sigma_v^2 \sum_{n=0}^{M} h_n^2$$

which, depending on the particular shape of h_n may be much larger than the original variance σ_v^2. This happens, for example, when f_n consists mainly of low frequencies. For h_n to compress f_n into a spike with a high frequency content, the impulse response h_n itself must be very spiky, which can result in values for $\mathbf{h}^T\mathbf{h}$ which are greater than one.

To combat the effects of noise, the least-squares design criterion for $\mathbf{h}$ must be changed by adding to it a term $\lambda E[u_n^2]$ as was done in Eq. (12.14.17). The modified design criterion is then

$$\mathcal{E} = \sum_n (\delta_n - h_n * f_n)^2 + \lambda\sigma_v^2 \sum_{n=0}^{M} h_n^2$$

which effectively amounts to changing the autocorrelation lag $\hat{R}_{ff}(0)$ into $(1+\epsilon)\hat{R}_{ff}(0)$. The first term in this performance index tries to produce a good inverse filter; the second term tries to minimize the output power of the noise after filtering by the deconvolution filter h_n. Note that conceptually this index is somewhat different from that of

Eq. (12.14.17), because now v_n represents the noise in the data y_n whereas there v_n represented inaccuracies in the knowledge of the wavelet f_n.

In this approach to deconvolution we are not attempting to determine the best least-squares estimate of the desired signal x_n, but rather the best least-squares inverse to the blurring function f_n. If the second order statistics of x_n were known, we could, of course, determine the optimal (Wiener) estimate $\hat{x}_n$ of x_n. This is also done in many applications.

The performance of the spiking filters and their usage in deconvolution are illustrated by the following example: The blurring function f_n to be spiked was chosen as

$$f_n = \begin{cases} g(n-25), & n = 0, 1, \dots, 65 \\ 0, & \text{for other } n \end{cases}$$

where $g(k)$ was the "gaussian hat" function:

$$g(k) = \cos(0.15k)\exp(-0.004k^2)$$

The signal x_n to be recovered was taken to be the series of delayed spikes:

$$x_n = \sum_{i=0}^{9} a_i \delta(n - n_i)$$

where the amplitudes a_i and delays n_i were chosen as

$$a_i = 1,\ 0.8,\ 0.5,\ 0.95,\ 0.7,\ 0.5,\ 0.3,\ 0.9,\ 0.5,\ 0.85$$

$$n_i = 25,\ 50,\ 60,\ 70,\ 80,\ 90,\ 100,\ 120,\ 140,\ 160$$

for $i = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9$.

Fig. 12.14.1 shows the signal f_n to be spiked. Since the gaussian hat is symmetric about the origin, we chose the spiking delay to be at $i = 25$. The order of the spiking filter h_n was $M = 50$. The right graph in Fig. 12.14.1 shows the impulse response h_n versus time. Note the spiky nature of h_n which is required here because f_n has a fairly low frequency content. Fig. 12.14.2 shows the results of the convolution $h_n * f_n$, which is the best least-squares approximation to the impulse $\delta(n-25)$.

The "goodness" of the spiking filter is judged by the diagonal entries of the performance matrix P, according to Eq. (12.14.16). For the chosen delay $k = 25$, we find $P(25, 25) = 0.97$. To obtain a better picture of the overall performance of the spiking filters, on the right in Fig. 12.14.2 we have plotted the diagonal elements $P(k, k)$ versus k. It is seen that the chosen delay $k = 25$ is nearly optimal. Fig. 12.14.3 shows the composite signal y_n obtained by convolving f_n and x_n, according to Eq. (12.14.20).

Fig. 12.14.3 shows on the right the deconvolved signal x_n according to Eq. (12.14.21). The recovery of the amplitudes a_i and delays n_i of x_n is very accurate. These results represent the idealistic case of noise-free data y_n and perfect knowledge of the blurring function f_n. To study the sensitivity of the deconvolution technique to inaccuracies in the knowledge of the signal f_n we have added a small high frequency perturbation on f_n as follows:

$$f_n' = f_n + 0.05\sin\bigl(1.5(n-25)\bigr)$$

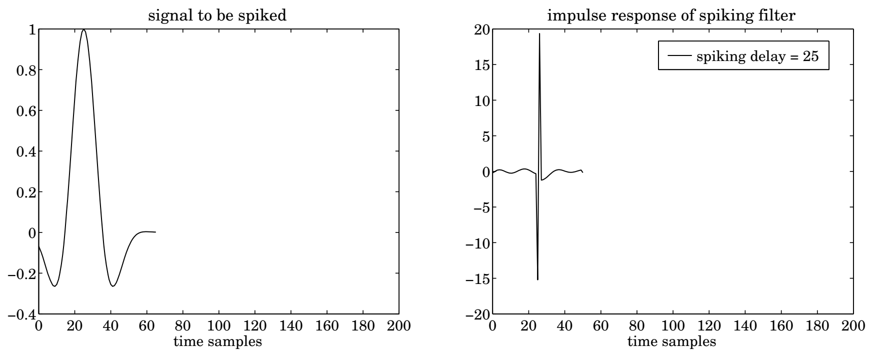


Fig. 12.14.1 Spiking filter and its inverse.

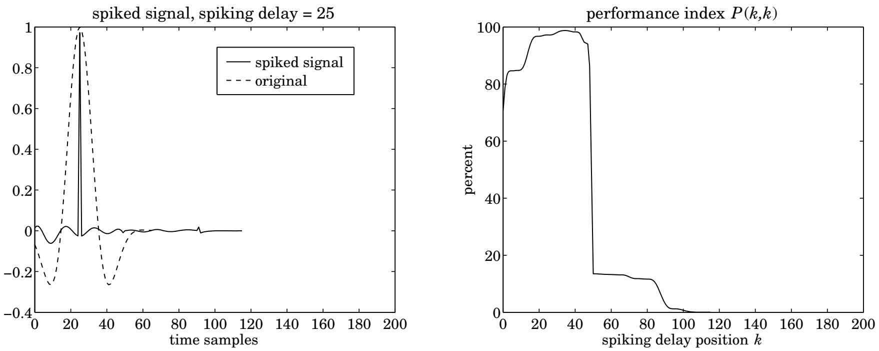


Fig. 12.14.2 Deconvolved signal and performance index.

The approximate signal f_n' is shown in Fig. 12.14.4. The spiking filter was designed on the basis of f_n' rather than f_n. The result of filtering the same composite signal y_n through the corresponding inverse filter is shown on the right in Fig. 12.14.4. The delays and amplitudes a_i and n_i are not well resolved, but the basic nature of x_n can still be seen. Inspecting Fig. 12.14.1 we note the large spikes that are present in the impulse response h_n of the inverse filter; these can cause the amplification of any additive noise component. Indeed, the noise reduction ratio of the filter h_n is $\mathbf{h}^T\mathbf{h} = 612$, thus it will tend to amplify even small amounts of noise.

To study the effect of noise, we have added a noise term v_n, as in Eq. (12.14.22), with variance equal to 10^{-4} (this corresponds to just 1% of the amplitude a_0); the composite signal y_n is shown on the left in Fig. 12.14.5. One can barely see the noise. Yet, after filtering with the inverse filter h_n of Fig. 12.14.1, the noise component is amplified to a great extent. The result of deconvolving the noisy y_n with h_n is shown on the right in Fig. 12.14.5. To reduce the effects of noise, the prewhitening parameter ϵ must be chosen to be nonzero. Even a small nonzero value of ϵ can have a beneficial effect. The

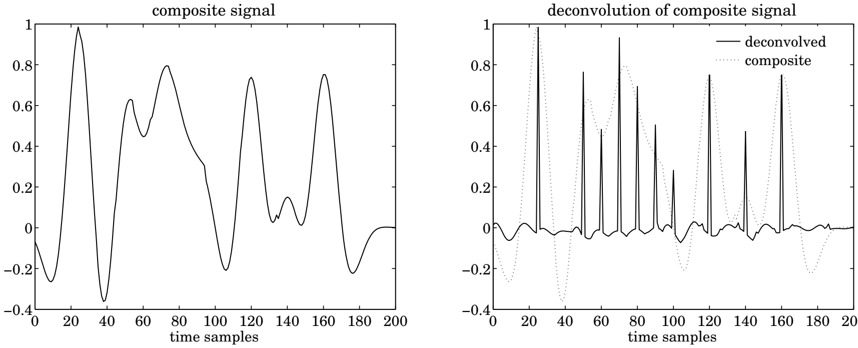


Fig. 12.14.3 Composite and deconvolved signal.

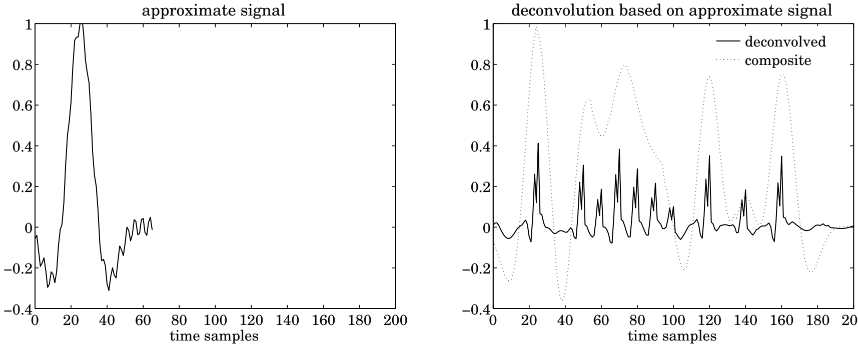


Fig. 12.14.4 Composite and deconvolved signal.

graphs in Fig. 12.14.6 show the deconvolved signal x_n when the filter h_n was designed with the choices $\epsilon = 0.0001$ and $\epsilon = 0.001$, respectively. Note the trade-off between the noise reduction and the loss of resolution in the recovered spikes of x_n.

Based on the studies of Robinson and Treitel [974], Oldenburg [1065], and others, the following summary of the use of the above deconvolution method may be made:

1. If the signal f_n to be spiked is a minimum-phase signal, the optimal spiking delay must be chosen at the origin $i = 0$. The optimality of this choice is not actually seen until the filter order M is sufficiently high. The reason for this choice has to do with the minimum-delay property of such signals which implies that most of their energy is concentrated at the beginning, therefore, they may be more easily compressed to spikes with zero delay.
2. If f_n is a mixed-delay signal, as in the above example, then the optimal spiking delay will have some intermediate value.
3. Even if the shape of f_n is not accurately known, the deconvolution procedure based on the approximate f_n might have some partial success in deconvolving the replicas of f_n.
4. In the presence of noise in the data y_n to deconvolved, some improvement may result by introducing a nonzero value for the prewhitening parameter ϵ, where effectively the sample autocorrelation $R_{ff}(0)$ is replaced by $(1+\epsilon)\hat{R}_{ff}(0)$. The trade-off is a resulting loss of resolution.

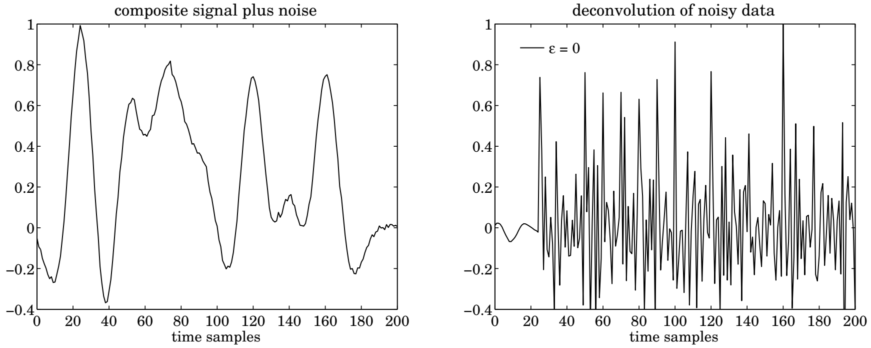


Fig. 12.14.5 Composite and deconvolved signal.

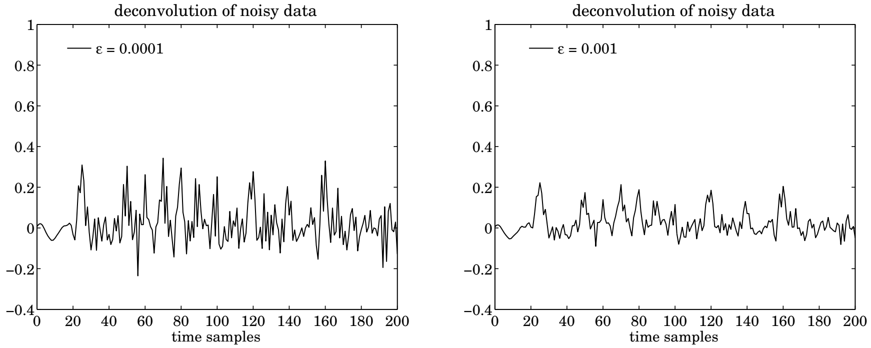


Fig. 12.14.6 Deconvolved signals.

The deconvolution problem of Eq. (12.14.20) and (12.14.22) has been approached by a wide variety of other methods. Typically, a finite number of samples y_n, $n = 0, 1, \dots, N$ is available. Collecting these into a vector $\mathbf{y} = [y_0, y_1, \dots, y_N]^T$, we write Eq. (12.14.22) in an obvious vectorial form

$$\mathbf{y} = F\mathbf{x} + \mathbf{v} \tag{12.14.23}$$

Instead of determining an approximate inverse filter for the blurring function F, an alternative method is to attempt to determine the best—in some sense—vector $\mathbf{x}$ which is compatible with these equations. A popular method is based on the least-squares criterion [1060].

$$\mathcal{E} = \sum_{n=0}^{N} v_n^2 = \mathbf{v}^T\mathbf{v} = (\mathbf{y}-F\mathbf{x})^T(\mathbf{y}-F\mathbf{x}) = \min \tag{12.14.24}$$

That is, $\mathbf{x}$ is chosen so as to minimize $\mathcal{E}$. Setting the derivative with respect to $\mathbf{x}$ to zero gives the standard least-squares solution

$$\hat{\mathbf{x}} = (F^TF)^{-1}F^T\mathbf{y}$$

A prewhitening term can be added to the right of the performance index to stabilize the indicated inverse

$$\mathcal{E} = \mathbf{v}^T\mathbf{v} + \lambda\mathbf{x}^T\mathbf{x}$$

with solution $\hat{\mathbf{x}} = (F^TF+\lambda I)^{-1}F^T\mathbf{y}$. Another approach that has been used with success is based on the L_1-norm criterion

$$\mathcal{E} = \sum_{n=0}^{N} |v_n| = \min \tag{12.14.25}$$

This quantity is referred to as the L_1 norm of the vector $\mathbf{v}$. The minimization of this norm with respect to $\mathbf{x}$ may be formulated as a *linear programming* problem [1063–1073]. It has been observed that this method performs very well in the presence of noise, and it tends to ignore a few "bad" data points—that is, those for which the noise value v_n might be abnormally high—in favor of the good points, whereas the standard least-squares method based on the L_2-norm (12.14.24) will spend all its efforts trying to minimize the few large terms in the sum (12.14.24), and might not result in as good an estimate of $\mathbf{x}$ as it would if the few bad data points were to be ignored. We discuss this approach further in Sec. 15.11

Another class of deconvolution methods are *iterative* methods, reviewed in [1074]. Such methods, like the linear programming method mentioned above, offer the additional option of enforcing *priori constraints* that may be known to be satisfied by $\mathbf{x}$, for example, positivity, band-limiting, or time-limiting constraints. The imposition of such constraints can improve the restoration process dramatically.

12.15 Computer Project – ARIMA Modeling

The Box-Jenkins airline data set has served as a benchmark in testing seasonal ARIMA models. In particular, it has led to the popular "airline model", which, for monthly data with yearly periodicity, is defined by the following innovations signal model:

$$\boxed{(1-Z^{-1})(1-Z^{-12})y_n = (1-bZ^{-1})(1-BZ^{-12})\epsilon_n} \tag{12.15.1}$$

where Z^{-1} denotes the delay operator and b, B are constants such that $|b| < 1$ and $|B| < 1$. In this experiment, we briefly consider this model, but then replace it with the following ARIMA model,

$$\boxed{(1-Z^{-12})y_n = \frac{1}{A(Z)}\,\epsilon_n}\,,\quad A(z) = 1 + a_1z^{-1} + a_2z^{-2} + \cdots + a_pz^{-p} \tag{12.15.2}$$

The airline data set can be loaded with the MATLAB commands:

```
Y = load('airline.dat');          % in OSP data file folder
Y = Y'; Y = Y(:);                 % concatenate rows
y = log(Y);                       % log data
```

The data represent monthly airline passengers for the period Jan. 1949 to Dec. 1960. There are $N = 144$ data points. In this experiment, we will work with a subset of the first $n_0 = 108$ points, that is, y_n, $0 \le n \le n_0 - 1$, and attempt to predict the future 36 months of data (108 + 36 = 144.)

a. Plot Y_n and the log-data $y_n = \ln Y_n$ versus n and note the yearly periodicity. Note how the log-transformation tends to equalize the apparent increasing amplitude of the original data.

b. Compute and plot the normalized sample ACF, $\rho_k = R(k)/R(0)$, of the *zero-mean* log-data for lags $0 \le k \le 40$ and note the peaks at multiples of 12 months.

c. Let $x_n = (1-Z^{-1})(1-Z^{-12})y_n$ in the model of Eq. (12.15.1). The signal x_n follows an MA model with spectral density:

$$S_{xx}(z) = \sigma_\epsilon^2(1-bz^{-1})(1-bz)(1-Bz^{-12})(1-Bz^{12})$$

Multiply the factors out and perform an inverse z-transform to determine the autocorrelation lags $R_{xx}(k)$. Show in particular, that

$$\frac{R_{xx}(1)}{R_{xx}(0)} = -\frac{b}{1+b^2}\,,\qquad \frac{R_{xx}(12)}{R_{xx}(0)} = -\frac{B}{1+B^2} \tag{12.15.3}$$

Filter the subblock $y_n, 0 \le n \le n_0 - 1$ through the filter $(1-z^{-1})(1-z^{-12})$ to determine x_n. You may discard the first 13 transient outputs of x_n. Use the rest of x_n to calculate its sample ACF and apply Eq. (12.15.3) to solve for the model parameters b, B.

This simple method gives values that are fairly close to the Box/Jenkins values determined by maximum likelihood methods, namely, $b = 0.4$ and $B = 0.6$, see Ref. [22].

d. Consider, next, the model of Eq. (12.15.2) with a second-order AR model i.e., $A(z)$ filter order of $p = 2$. Define $x_n = (1-Z^{-12})y_n = y_n - y_{n-12}$ and denote its autocorrelation function by R_k. Since x_n follows an AR(2) model, its model parameters $a_1, a_2, \sigma_\epsilon^2$ can be computed from:

$$\begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = -\begin{bmatrix} R_0 & R_1 \\ R_1 & R_0 \end{bmatrix}^{-1}\begin{bmatrix} R_1 \\ R_2 \end{bmatrix}\,,\qquad \sigma_\epsilon^2 = R_0 + a_1R_1 + a_2R_2 \tag{12.15.4}$$

Using the data subset $y_n, 0 \le n \le n_0 - 1$, calculate the signal x_n and discard the first 12 transient samples. Using the rest of x_n, compute its sample ACF, $\hat{R}_k$ for $0 \le k \le M$ with $M = 40$, and use the first 3 computed lags $\hat{R}_0, \hat{R}_1, \hat{R}_2$ in Eqs. (12.15.4) to estimate $a_1, a_2, \sigma_\epsilon^2$ (in computing the ACF, the signal x_n need not be replaced by its zero-mean version).

Because of the assumed autoregressive model, it is possible to calculate all the autocorrelation lags R_k for $k \ge p + 1$ from the first $p + 1$ lags. This can accomplished by the MATLAB "autocorrelation sequence extension" function:

```
M=40; Rext = acext(R(1:p+1), zeros(1,M-p));
```

On the same graph, plot the sample and extended ACFs, $\hat{R}(k)$ and $R_{\text{ext}}(k)$ versus $0 \le k \le M$ normalized by their lag-0 values.

e. Let $\hat{x}_n$ denote the estimate/prediction of x_n based on the data subset $\{x_m, m \le n_0 - 1\}$. Clearly, $\hat{x}_n = x_n$, if $n \le n_0 - 1$. Writing Eq. (12.15.2) recursively, we obtain for the predicted values into the future beyond n_0:

$$\begin{aligned} x_n &= -\big(a_1 x_{n-1} + a_2 x_{n-2}\big) + \varepsilon_n \\ \hat{x}_n &= -\big(a_1 \hat{x}_{n-1} + a_2 \hat{x}_{n-2}\big)\,, \quad \text{for } n \ge n_0 \end{aligned} \tag{12.15.5}$$

where we set $\hat{\varepsilon}_n = 0$ because all the observations are in the strict past of ε_n when $n \ge n_0$.

Calculate the predicted values 36 steps into the future, $\hat{x}_n$ for $n_0 \le n \le N-1$, using the fact that $\hat{x}_n = x_n$, if $n \le n_0 - 1$. Once you have the predicted x_n's, you can calculate the predicted y_n's by the recursive equation:

$$\hat{y}_n = \hat{y}_{n-12} + \hat{x}_n \tag{12.15.6}$$

where you must use $\hat{y}_n = y_n$, if $n \le n_0 - 1$. Compute $\hat{y}_n$ for $n_0 \le n \le N-1$, and plot it on the same graph with the original data y_n, $0 \le n \le N-1$. Indicate the start of the prediction horizon with a vertical line at $n = n_0$ (see example graph at end.)

f. Repeat parts (d,e) using an AR(4) model (i.e., $p = 4$), with signal model equations:

$$x_n = y_n - y_{n-12}\,, \quad x_n = -\big(a_1 x_{n-1} + a_2 x_{n-2} + a_3 x_{n-3} + a_4 x_{n-4}\big) + \varepsilon_n$$

and normal equations:

$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} = -\begin{bmatrix} R_0 & R_1 & R_2 & R_3 \\ R_1 & R_0 & R_1 & R_2 \\ R_2 & R_1 & R_0 & R_1 \\ R_3 & R_2 & R_1 & R_0 \end{bmatrix}^{-1} \begin{bmatrix} R_1 \\ R_2 \\ R_3 \\ R_4 \end{bmatrix}, \quad \sigma_\varepsilon^2 = R_0 + a_1 R_1 + a_2 R_2 + a_3 R_3 + a_4 R_4$$

with predictions:

$$\begin{aligned} \hat{x}_n &= -\big(a_1 \hat{x}_{n-1} + a_2 \hat{x}_{n-2} + a_3 \hat{x}_{n-3} + a_4 \hat{x}_{n-4}\big) \\ \hat{y}_n &= \hat{y}_{n-12} + \hat{x}_n \end{aligned}$$

taking into account the properties that $\hat{x}_n = x_n$ and $\hat{y}_n = y_n$, if $n \le n_0 - 1$.

g. Inspecting the log-data, it is evident that there is an almost linear trend as well as a twelve-month, or even, a six-month periodicity. In this part, we will attempt to fit the the data using a conventional basis-functions method and then use this model to predict the last 36 months.

Consider the following model for the first n_0 points of the log data y_n that assumes a quadratic trend and 12-month, 6-month, and 4-month periodicities, i.e., three harmonics of the fundamental frequency of $1/12$ cycle/month, for $0 \le n \le n_0 - 1$,

$$\begin{aligned} \hat{y}_n = c_0 + c_1 n + c_2 n^2 &+ c_3 \sin\Big(\frac{2\pi n}{12}\Big) + c_4 \cos\Big(\frac{2\pi n}{12}\Big) + \\ &+ c_5 \sin\Big(\frac{4\pi n}{12}\Big) + c_6 \cos\Big(\frac{4\pi n}{12}\Big) \\ &+ c_7 \sin\Big(\frac{6\pi n}{12}\Big) + c_8 \cos\Big(\frac{6\pi n}{12}\Big) \end{aligned} \tag{12.15.7}$$

Carry out a least-squares fit to determine the nine coefficients c_i, that is, determine the c_i that minimize the quadratic performance index,

$$\mathcal{J} = \sum_{n=0}^{n_0-1} \big(y_n - \hat{y}_n\big)^2 = \min \tag{12.15.8}$$

In addition, carry out a fit based on the L_1 criterion,

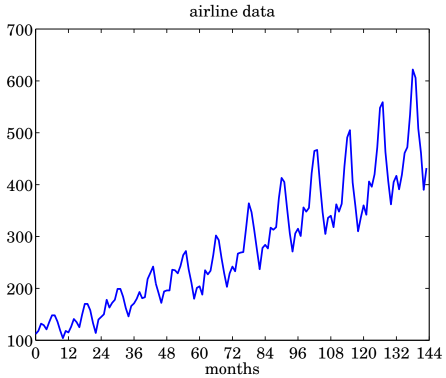
$$\mathcal{J} = \sum_{n=0}^{n_0-1} \big|y_n - \hat{y}_n\big| = \min \tag{12.15.9}$$

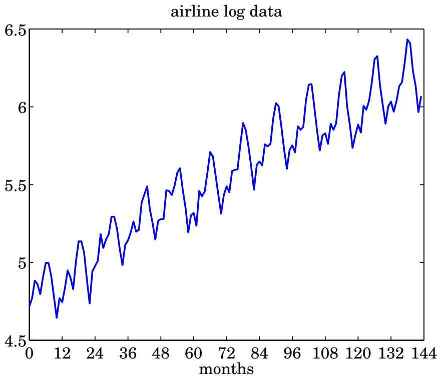
Show that the coefficients are, for the two criteria,

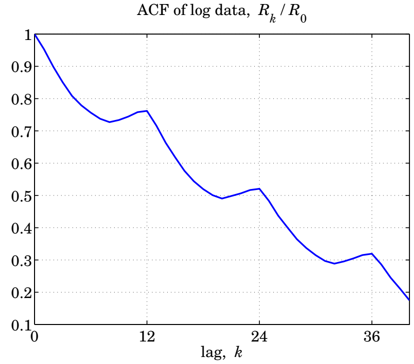
	L_2	L_1
c_0	4.763415	4.764905
c_1	0.012010	0.011912
c_2	−0.000008	−0.000006
c_3	0.036177	0.026981
c_4	−0.135907	−0.131071
c_5	0.063025	0.067964
c_6	0.051890	0.051611
c_7	−0.025511	−0.017619
c_8	−0.009168	−0.012753

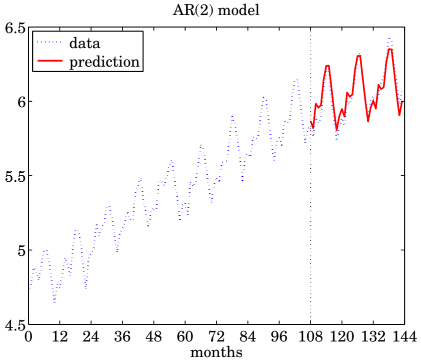
Using the found coefficients for each criterion, evaluate the fitted quantity $\hat{y}_n$ of Eq. (12.15.7) over the entire data record, $0 \le n \le N-1$, and plot it on the same graph together with the actual data, and with the AR(4) prediction. It appears that this approach also works well but over a shorter prediction interval, i.e., 24 months instead of 36 for the AR(4) method.

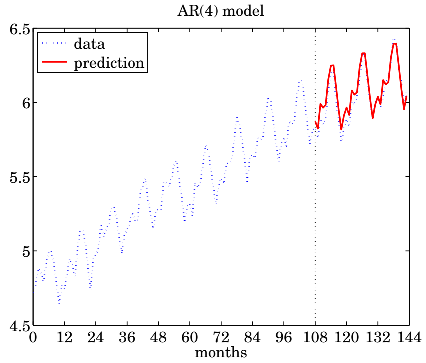
Some example graphs for this experiment are included below.

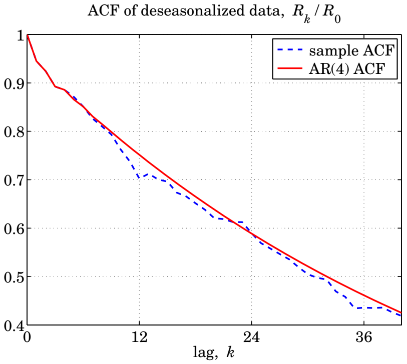


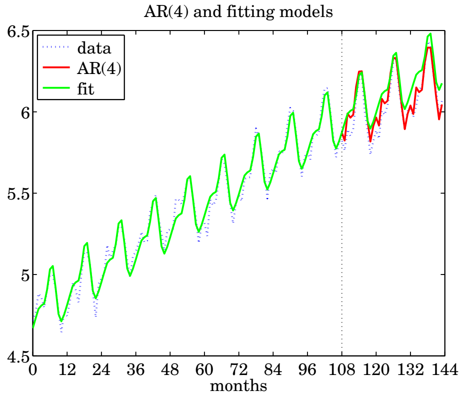












12.16 Problems

12.1 (a) Following the methods of Sec. 12.1, show that the optimal filter for predicting D steps into the future—i.e., estimating $y(n+D)$ on the basis of $\{y(m);\ m \le n\}$—is given by

$$H(z) = \frac{1}{B(z)}\big[z^D B(z)\big]_+$$

(b) Express $\big[z^D B(z)\big]_+$ in terms of $B(z)$ itself and the first $D-1$ impulse response coefficients b_m, $m = 1, 2, \dots, D-1$ of $B(z)$.

(c) For the two random signals y_n defined in Examples 12.1.1 and 12.1.2, find the optimal prediction filters for $D = 2$ and $D = 3$, and write the corresponding I/O equations.

12.2 Consider the order-p autoregressive sequence y_n defined by the difference equation (12.2.3). Show that a direct consequence of this difference equation is that the projection of y_n onto the subspace spanned by the entire past $\{y_{n-i};\ 1 \le i < \infty\}$ is the same as the projection of y_n onto the subspace spanned only by the past p samples $\{y_{n-i};\ 1 \le i \le p\}$.

12.3 (a) Show that the performance index (12.3.2) may be written as

$$\mathcal{E} = E[e_n^2] = \mathbf{a}^T R\mathbf{a}$$

where $\mathbf{a} = [1, a_1, \dots, a_p]^T$ is the order-p prediction-error filter, and R the autocorrelation matrix of y_n; that is, $R_{ij} = E[y_{n-i}y_{n-j}]$.

(b) Derive Eq. (12.3.7) by minimizing the index $\mathcal{E}$ with respect to the weights $\mathbf{a}$, subject to the linear constraint that $a_0 = 1$, and incorporating this constraint by means of a Lagrange multiplier.

12.4 Take the inverse z-transform of Eq. (12.3.17) and compare the resulting equation with Eq. (12.3.15).

12.5 Verify that Eq. (12.3.22) and (12.3.23) are inverses of each other.

12.6 A fourth order all-pole random signal process $y(n)$ is represented by the following set of signal model parameters (reflection coefficients and input variance):

$$\{\gamma_1, \gamma_2, \gamma_3, \gamma_4, \sigma_\epsilon^2\} = \{0.5,\ -0.5,\ 0.5,\ -0.5,\ 40.5\}$$

(a) Using the Levinson recursion, find the prediction error filter $A_4(z)$.

(b) Determine $\sigma_y^2 = R_{yy}(0)$. Using intermediate results from part (a), determine the autocorrelation lags $R_{yy}(k)$, $k = 1, 2, 3, 4$.

12.7 The first five lags of the autocorrelation function of a fourth-order autoregressive random sequence $y(n)$ are

$$\{R(0), R(1), R(2), R(3), R(4)\} = \{256,\ 128,\ -32,\ -16,\ 22\}$$

Determine the best prediction-error filters and the corresponding mean-square errors of orders $p = 1, 2, 3, 4$ by using Levinson's algorithm in matrix form.

12.8 The fourth-order prediction-error filter and mean-square prediction error of a random signal have been determined to be

$$A_4(z) = 1 - 1.25z^{-1} + 1.3125z^{-2} - z^{-3} + 0.5z^{-4}, \quad E_4 = 0.81$$

Using the function **rlev**, determine the autocorrelation lags $R(k)$, $0 \le k \le 4$, the four reflection coefficients, and all the lower order prediction-error filters.

12.9 Verify the results of Example 12.3.1 using the functions **lev**, **frwlev**, **bkwlev**, and **rlev**, as required.

12.10 (a) Given the five signal samples

$$\{y_0, y_1, y_2, y_3, y_4\} = \{1,\ -1,\ 1,\ -1,\ 1\}$$

compute the corresponding sample autocorrelation lags $\hat{R}(k)$, $k = 0, 1, 2, 3, 4$, and send them through the function **lev** to determine the fourth-order prediction error filter $A_4(z)$.

(b) Predict the sixth sample in this sequence.

(c) Repeat (a) and (b) for the sequence of samples $\{1, 2, 3, 4, 5\}$.

12.11 Find the infinite autoregressive or maximum-entropy extension of the two autocorrelation sequences

(a) $\{R(0), R(1)\} = \{1,\ 0.5\}$

(b) $\{R(0), R(1), R(2)\} = \{4,\ 0,\ 1\}$

In both cases, determine the corresponding power spectrum density $S_{yy}(z)$ and from it calculate the $R(k)$ for all lags k.

12.12 Write Eq. (12.3.24) for order $p + 1$. Derive Eq. (12.5.1) from Eq. (12.3.24) by replacing the filter $\mathbf{a}_{p+1}$ in terms of the filter $\mathbf{a}_p$ via the Levinson recursion.

12.13 Do Problem 12.7 using the split Levinson algorithm.

12.14 Draw the lattice realization of the analysis and synthesis filters $A_4(a)$ and $1/A_4(z)$ obtained in Problems 12.6, 12.7, and 12.8.

12.15 Test the minimum-phase property of the two polynomials

$$A(z) = 1 - 1.08z^{-1} + 0.13z^{-2} + 0.24z^{-3} - 0.5z^{-4}$$
$$A(z) = 1 + 0.18z^{-1} - 0.122z^{-2} - 0.39z^{-3} - 0.5z^{-4}$$

12.16 (a) The entropy of an M-dimensional random vector is defined by $S = -\int p(\mathbf{y})\ln p(\mathbf{y})\, d^M\mathbf{y}$. Show that the entropy of a zero-mean gaussian $\mathbf{y}$ with covariance matrix R is given, up to an additive constant, by $S = \frac{1}{2}\ln(\det R)$.

(b) With the help of the LU factorization (12.9.1), show that ratio of the determinants of an order M autocorrelation matrix and its order p $(p < M)$ submatrix is

$$\frac{\det R_M}{\det R_p} = \prod_{i=p+1}^{M} E_i$$

(c) Consider all possible autocorrelation extensions of the set $\{R(0), R(1), \dots, R(p)\}$ up to order M. For gaussian processes, use the results in parts (a) and (b) to show that the particular extension defined by the choice $\gamma_i = 0$, $i = p+1, \dots, M$ maximizes the entropy of the order-M process; hence, the name maximum entropy extension.

12.17 Consider the LU factorization $LRL^T = D$ of an order-M autocorrelation matrix R. Denote by $\mathbf{b}_p^T$, $p = 0, 1, \dots, M$ the rows of L. They are the backward prediction filters with zeros padded to their ends to make them $(M+1)$-dimensional vectors.

(a) Show that the inverse factorization $R^{-1} = L^T D^{-1} L$ can be written as

$$R^{-1} = \sum_{p=0}^{M} \frac{1}{E_p}\mathbf{b}_p\mathbf{b}_p^T$$

(b) Define the "phasing" vectors $\mathbf{s}(z) = [1, z^{-1}, z^{-2}, \dots, z^{-M}]^T$. Show that the z-transform of an order-M filter and its inverse can be expressed compactly as

$$A(z) = \mathbf{s}(z)^T\mathbf{a}, \quad \mathbf{a} = \oint_{\text{u.c}} A(z)\mathbf{s}(z^{-1})\,\frac{dz}{2\pi jz}$$

(c) Define the "kernel" vector $\mathbf{k}(w) = R^{-1}\mathbf{s}(w)$. The z-transform of this vector is called a *reproducing kernel* [972,973,981]. Show that it can be written in the alternative forms

$$K(z,w) = \mathbf{s}(z)^T\mathbf{k}(w) = \mathbf{k}(z)^T\mathbf{s}(w) = \mathbf{k}(z)^T R\,\mathbf{k}(w) = \mathbf{s}(z)^T R^{-1}\mathbf{s}(w)$$

(d) Let J denote the $(M+1)\times(M+1)$ reversing matrix. Show that $J\mathbf{s}(z) = z^{-M}\mathbf{s}(z^{-1})$. And that $K(z,w) = z^{-M}w^{-M}K(z^{-1}, w^{-1})$.

(e) Show that $K(z,w)$ admits the following representations in terms of the backward and forward prediction polynomials

$$K(z,w) = \sum_{p=0}^{M}\frac{1}{E_p}B_p(z)B_p(w) = \sum_{p=0}^{M}\frac{1}{E_p}A_p(z)A_p(w)z^{-(M-p)}w^{-(M-p)}$$

12.18 Let $S_{yy}(z)$ be the power spectral density of the autocorrelation function $R(k)$ from which we build the matrix R of the previous problem. Show that R and R^{-1} admit the following representations in terms of the phasing and kernel vectors:

$$R = \oint_{\text{u.c.}} S_{yy}(z)\mathbf{s}(z^{-1})\mathbf{s}(z)^T\,\frac{dz}{2\pi jz}, \quad R^{-1} = \oint_{\text{u.c.}} S_{yy}(z)\mathbf{k}(z^{-1})\mathbf{k}(z)^T\,\frac{dz}{2\pi jz}$$

Then, show the *reproducing kernel* property

$$K(z,w) = \oint_{\text{u.c.}} K(z,u^{-1})K(w,u)S_{yy}(u)\,\frac{du}{2\pi ju}$$

12.19 (a) Let $\mathbf{s}_p(z)=[1,\, z^{-1},\, z^{-2},\, \dots,\, z^{-p}]^T$. Using the order-updating formulas for R_p^{-1} show that the kernel vector $\mathbf{k}_p(w)=R_p^{-1}\mathbf{s}_p(w)$ satisfies the following order-recursive equations

$$\mathbf{k}_p(w)=\begin{bmatrix}\mathbf{k}_{p-1}(w)\\ 0\end{bmatrix}+\frac{1}{E_p}\mathbf{b}_pB_p(w)\,,\quad \mathbf{k}_p(w)=\begin{bmatrix}0\\ w^{-1}\mathbf{k}_{p-1}(w)\end{bmatrix}+\frac{1}{E_p}\mathbf{a}_pA_p(w)$$

(b) Show that the corresponding reproducing kernels satisfy

$$\begin{aligned}K_p(z,w) &= K_{p-1}(z,w)+\frac{1}{E_p}B_p(z)B_p(w)\\ K_p(z,w) &= z^{-1}w^{-1}K_{p-1}(z,w)+\frac{1}{E_p}A_p(z)A_p(w)\end{aligned}$$

(c) Using part (b), show the Christoffel-Darboux formula [972,973,981]

$$K_p(z,w)=\frac{1}{E_p}\,\frac{A_p(z)A_p(w)-z^{-1}w^{-1}B_p(z)B_p(w)}{1-z^{-1}w^{-1}}$$

(d) Let z_i be the ith zero of the prediction polynomial $A_p(z)$. Using part (c), evaluate $K_p(z_i,z_i^*)$ and thereby show that necessarily $|z_i|\le 1$. This is yet another proof of the minimum-phase property of the prediction-error filters. Show further that if the prediction filter $\mathbf{a}_p$ is symmetric; i.e., $\mathbf{a}_p=\mathbf{a}_p^R$, then its zeros lie on the unit circle.

(e) Show the Christoffel-Darboux formula [972,973,981]

$$K_{p-1}(z,w)=\frac{1}{E_p}\,\frac{A_p(z)A_p(w)-B_p(z)B_p(w)}{1-z^{-1}w^{-1}}$$

and use this expression to prove the result in (d) that $|z_i|\le 1$.

12.20 Do Problem 12.7 using the Schur algorithm, determine the Cholesky factor G, and verify $R=GD^{-1}G^T$ by explicit matrix multiplication.

12.21 For the Example 12.10.2, compute the entries of the output matrices $Y^{\pm}$ by directly convolving the forward/backward prediction filters with the input autocorrelation lags.

12.22 Do Problem 12.7 using the split Schur algorithm, and determine the Cholesky factor G by the recursion (12.10.21).

12.23 (a) Show the identity

$$\left|\frac{-a^*+z^{-1}}{1-az^{-1}}\right|^2=1-\frac{\big(1-|z^{-1}|^2\big)\big(1-|a|^2\big)}{|1-az^{-1}|^2}$$

(b) Using part (a), show that the all-pass Schur function $S_p(z)$ defined by Eq. (12.10.22) satisfies the boundedness inequality (12.10.23), with equality attained on the unit circle. Show that it also satisfies $|S_p(z)|>1$ for $|z|<1$.

12.24 Define the Schur function

$$S_3(z)=\frac{0.125-0.875z^{-2}+z^{-3}}{1-0.875z^{-1}+0.125z^{-3}}$$

Carry out the recursions (12.10.24) and (12.10.25) to construct the lower order Schur functions $S_p(z)$, $p=2,1,0$, and, in the process, extract the corresponding reflection coefficients.

12.25 Consider a generalized version of the simulation example discussed in Sec. 12.11, defined by

$$x(n)=s(n)+v_1(n)\,,\quad y(n)=v_2(n)$$

where

$$\begin{aligned}s(n) &= \sin(\omega_0 n+\phi)\\ v_1(n) &= a_1v_1(n-1)+v(n)\\ v_2(n) &= a_2v_2(n-1)+v(n)\end{aligned}$$

where $v(n)$ is zero-mean, unit-variance, white noise, and ϕ is a random phase independent of $v(n)$. This ensures that the $s(n)$ component is uncorrelated with $v_1(n)$ and $v_2(n)$.

(a) Show that

$$R_{xy}(k)=\frac{a_1^k}{1-a_1a_2}\,,\quad R_{yy}(k)=\frac{a_2^k}{1-a_2^2}\,,\quad k\ge 0$$

(b) Show that the infinite-order Wiener filter for estimating $x(n)$ on the basis of $y(n)$ has a (causal) impulse response

$$h_0=1\,,\quad h_k=(a_1-a_2)a_1^{k-1}\,,\quad k\ge 1$$

(c) Next, consider the order-M FIR Wiener filter. Send the theoretical correlations of part (a) for $k=0,1,\dots,M$ through the function **firw** to obtain the theoretical Mth order Wiener filter realized both in the direct and the lattice forms. Draw these realizations. Compare the theoretical values of the weights $\mathbf{h}$, $\mathbf{g}$, and $\boldsymbol{\gamma}$ with the simulated values presented in Sec. 12.11 that correspond to the choice of parameters $M=4$, $a_1=-0.5$, and $a_2=0.8$. Also compare the answer for $\mathbf{h}$ with the first $(M+1)$ samples of the infinite-order Wiener filter impulse response of part (b).

(d) Repeat (c) with $M=6$.

12.26 A closed form solution of Problem 12.25 can be obtained as follows.

(a) Show that the inverse of the $(M+1)\times(M+1)$ autocorrelation matrix defined by the autocorrelation lags $R_{yy}(k)$, $k=0,1,\dots,M$ of Problem 12.25(a) is given by the tridiagonal matrix:

$$R_{yy}^{-1}=\begin{bmatrix}1 & -a_2 & 0 & \cdots & 0 & 0\\ -a_2 & b & -a_2 & \cdots & 0 & 0\\ 0 & -a_2 & b & \cdots & 0 & 0\\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots\\ 0 & 0 & 0 & \cdots & b & -a_2\\ 0 & 0 & 0 & \cdots & -a_2 & 1\end{bmatrix}$$

where $b=1+a_2^2$.

(b) Using this inverse, show that the optimal Mth order Wiener filter has impulse response

$$h_0=1\,,\quad h_k=(a_1-a_2)a_1^{k-1}\,,\quad \text{for } 1\le k\le M-1\,,\quad \text{and } h_M=\frac{a_1-a_2}{1-a_1a_2}\,a_1^{M-1}$$

(c) Show that the lattice weights $\mathbf{g}$ can be obtained from $\mathbf{h}$ by the backward substitution

$$g_M=h_M\,,\quad \text{and}\quad g_m=a_2g_{m+1}+h_m\,,\quad m=M-1,M-2,\dots,1,0$$

(d) For $M = 4$, $a_1 = -0.5$, $a_2 = 0.8$, compute the numerical values of $\mathbf{h}$ and $\mathbf{g}$ using the above expressions and compare them with those of Problem 12.25(c).

12.27 *Computer Experiment.* Consider the noise canceling example of Sec. 12.11 and Problem 12.25, defined by the choice of parameters

$$\omega_0 = 0.075\pi \text{ [radians/sample]}, \quad \phi = 0, \quad a_1 = -0.5, \quad a_2 = 0.8, \quad M = 4$$

(a) Generate 100 samples of the signals $x(n)$, $s(n)$, and $y(n)$. On the same graph, plot $x(n)$ and $s(n)$ versus n. Plot $y(n)$ versus n.

(b) Using these samples, compute the sample correlations $\hat{R}_{yy}(k)$, $\hat{R}_{xy}(k)$, for $k = 0, 1, \dots, M$, and compare them with the theoretical values obtained in Problem 12.25(a).

(c) Send these lags through the function **firw** to get the optimal Wiener filter weights $\mathbf{h}$ and $\mathbf{g}$, and the reflection coefficients $\boldsymbol{\gamma}$. Draw the lattice and direct-form realizations of the Wiener filter.

(d) Filter $y(n)$ through the Wiener filter realized in the lattice form, and plot the output $e(n) = x(n) - \hat{x}(n)$ versus n.

(e) Repeat (d) using the direct-form realization of the Wiener filter.

(f) Repeat (d) when $M = 6$.

12.28 The following six samples

$$\{y_0, y_1, y_2, y_3, y_4, y_5\} = \{4.684,\ 7.247,\ 8.423,\ 8.650,\ 8.640,\ 8.392\}$$

have been generated by sending zero-mean unit-variance white noise through the difference equation

$$y_n = a_1 y_{n-1} + a_2 y_{n-2} + \epsilon_n$$

where $a_1 = 1.70$ and $a_2 = -0.72$. Iterating Burg's method by hand, obtain estimates of the model parameters a_1, a_2, and σ_ϵ^2.

12.29 Derive Eq. (12.12.11).

12.30 *Computer Experiment.* Ten samples from a fourth-order autoregressive process $y(n)$ are given. It is desired to extract the model parameters $\{a_1, a_2, a_3, a_4, \sigma_\epsilon^2\}$ as well as the equivalent parameter set $\{\gamma_1, \gamma_2, \gamma_3, \gamma_4, \sigma_\epsilon^2\}$.

(a) Determine these parameters using Burg's method.
(b) Repeat using the Yule-Walker method.

Note: The exact parameter values by which the above simulated samples were generated are

$$\{a_1, a_2, a_3, a_4, \sigma_\epsilon^2\} = \{-2.2137,\ 2.9403,\ -2.2697,\ 0.9606,\ 1\}$$

n	$y(n)$
0	4.503
1	−10.841
2	−24.183
3	−25.662
4	−14.390
5	1.453
6	10.980
7	13.679
8	15.517
9	15.037

12.31 Using the continuity equations at an interface, derive the transmission matrix equation (12.13.2) and the energy conservation equation (12.13.4).

12.32 Show Eq. (12.13.6).

12.33 Fig. 12.13.2 defines the scattering matrix S. Explain how the principle of linear superposition may be used to show the general relationship

$$\begin{bmatrix} E'_+ \\ E_- \end{bmatrix} = S \begin{bmatrix} E_+ \\ E'_- \end{bmatrix}$$

between incoming and outgoing fields.

12.34 Show the two properties of the matrix $\psi_m(z)$ stated in Eqs. (12.13.13) and (12.13.14).

12.35 Show Eqs. (12.13.25).

12.36 The reflection response of a stack of four dielectrics has been found to be

$$R(z) = \frac{-0.25 + 0.0313z^{-1} + 0.2344z^{-2} - 0.2656z^{-3} + 0.25z^{-4}}{1 - 0.125z^{-1} + 0.0664z^{-3} - 0.0625z^{-4}}$$

Determine the reflection coefficients $\{\rho_0, \rho_1, \rho_2, \rho_3, \rho_4\}$.

12.37 *Computer Experiment.* It is desired to probe the structure of a stack of dielectrics from its reflection response. To this end, a unit impulse is sent incident on the stack and the reflection response is measured as a function of time.

It is known in advance (although this is not necessary) that the stack consists of four equal travel-time slabs stacked in front of a semi-infinite medium.

Thirteen samples of the reflection response are collected as shown here. Determine the reflection coefficients $\{\rho_0, \rho_1, \rho_2, \rho_3, \rho_4\}$ by means of the dynamic predictive deconvolution procedure.

k	$R(k)$
0	−0.2500
1	0.0000
2	0.2344
3	−0.2197
4	0.2069
5	0.0103
6	0.0305
7	−0.0237
8	0.0093
9	−0.0002
10	0.0035
11	−0.0017
12	0.0004

12.38 *Computer Experiment.* Generate the results of Figures 5.16–5.17 and 5.25–5.26.

12.39 *Computer Experiment.* This problem illustrates the use of the dynamic predictive deconvolution method in the design of broadband terminations of transmission lines. The termination is constructed by the cascade of M equal travel-time segments of transmission lines such that the overall reflection response of the structure approximates the desired reflection response. The characteristic impedances of the various segments are obtainable from the reflection coefficients $\{\rho_0, \rho_1, \dots, \rho_M\}$. The reflection response $R(\omega)$ of the structure is a periodic function of ω with period $\omega_s = 2\pi/T_2$, where T_2 is the two-way travel time delay of each segment. The design procedure is illustrated by the following example: The desired frequency response $R(\omega)$ is defined over one Nyquist period, as shown in Fig. 5.39:

$$R(\omega) = \begin{cases} 0, & \text{for } 0.25\omega_2 \le \omega \le 0.75\omega_s \\ 0.9, & \text{for } 0 \le \omega < 0.25\omega_s \text{ and } 0.75\omega_s < \omega \le \omega_s \end{cases}$$

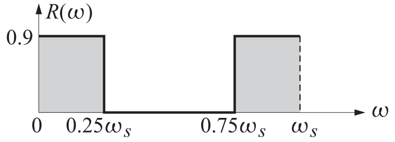


Fig. 12.16.1 Desired reflection frequency response.

(a) Using the Fourier series method of designing digital filters, design an $N = 21$-tap filter with impulse response $R(k)$, $k = 0, 1, \dots, N-1$, whose frequency response approximates the desired response defined above. Window the designed reflection impulse response $R(k)$ by a length-N Hamming window. Plot the magnitude frequency response of the windowed reflection series over one Nyquist interval $0 \le \omega \le \omega_s$.

(b) For $M = 6$, send the N samples of the windowed reflection series through the dynamic predictive deconvolution function **dpd** to obtain the polynomials $A_M(z)$ and $B_M(z)$ and the reflection coefficients $\{\rho_0, \rho_1, \dots, \rho_M\}$. Plot the magnitude response of the structure; that is, plot

$$|R(\omega)| = \left|\frac{B_M(z)}{A_M(z)}\right|, \quad z = \exp(j\omega T_2) = \exp\left(2\pi j \frac{\omega}{\omega_s}\right)$$

and compare it with the windowed response of part (a). To facilitate the comparison, plot both responses of parts (a) and (b) on the same graph.

(c) Repeat part (b) for $M = 2$, $M = 3$, and $M = 10$.

(d) Repeat parts (a) through (c) for $N = 31$ reflection series samples.

(e) Repeat parts (a) through (c) for $N = 51$.

12.40 Show that the performance matrix P defined by Eq. (12.14.11) has trace equal to $M + 1$.

12.41 *Computer Experiment.* Reproduce the results of Figs. 12.14.1 through 12.14.6.

12.42 *Computer Experiment – ARIMA Models.* The file GNPC96.dat contains the data for the U.S. Real Gross National Product (in billions of chained 2005 dollars, measured quarterly and seasonally adjusted.) In this experiment, you will test whether an ARIMA$(p,1,0)$ model is appropriate for these data.

Extract the data from 1980 onwards and take their log. This results in $N = 119$ observations, $y_n = \ln(\text{GNP}_n)$, $n = 0, 1, \dots, N-1$. Because of the upward trend of the data, define the length-$(N-1)$ differenced signal $z_n = y_n - y_{n-1}$, for $n = 1, 2, \dots, N-1$. Then, subtract its sample mean, say μ, to get the signal $x_n = z_n - \mu$, for $n = 1, 2, \dots, N-1$.

a. Calculate and plot the first $M = 24$ sample autocorrelation lags $R(k)$, $0 \le k \le M$, of the signal x_n. Send these into the Levinson algorithm to determine and plot the corresponding reflection coefficients γ_p, for $p = 1, 2 \dots, M$. Add on that graph the 95% confidence bands, that is, the horizontal lines at $\pm 1.96/\sqrt{N}$. Based on this plot, determine a reasonable value for the order p of an autoregressive model of fitting the x_n data.

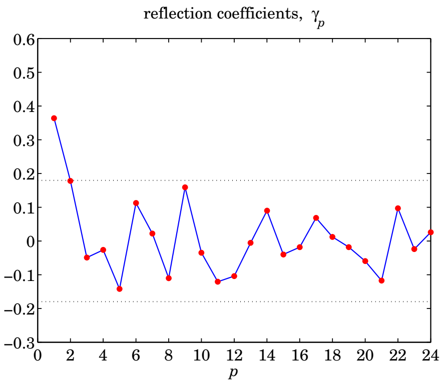
b. Check the chosen value of p against also the FPE, AIC, and MDL criteria, that is, by plotting them versus $p = 1, 2, \dots, M$, and identifying their minimum:

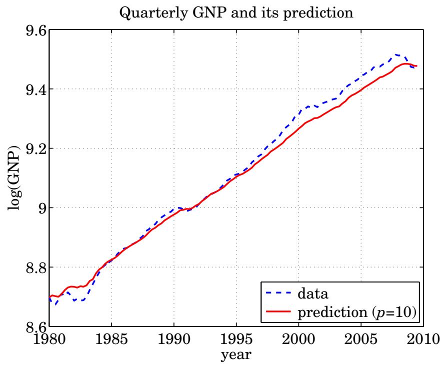
$$\text{FPE}_p = E_p \frac{N + p + 1}{N - p - 1}, \quad \text{AIC}_p = N \ln E_p + 2(p+1), \quad \text{MDL}_p = N \ln E_p + (p+1)\ln N$$

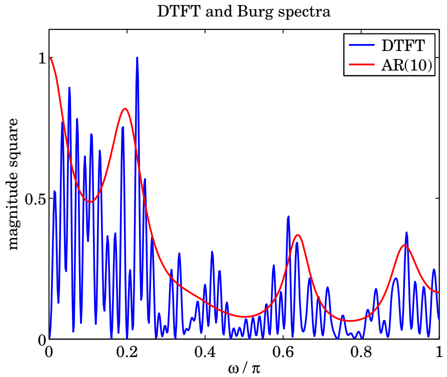
c. For the chosen value of p, use the Yule-Walker method to determine the linear prediction error filter $\mathbf{a}_p$ of order p, then, calculate the one-step-ahead prediction $\hat{x}_{n/n-1}$, add the mean μ to get the prediction $\hat{z}_{n/n-1}$, and undo the differencing operation to compute the prediction $\hat{y}_{n/n-1}$ of y_n. There is a small subtlety here that has to do with the initial value of y_n. On the same graph plot y_n and its prediction.

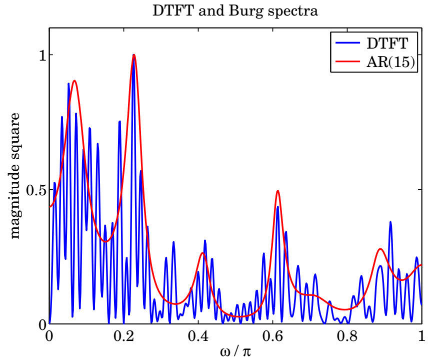
d. Repeat part (c) for a couple of other values of p, say $p = 1$ and $p = 10$.

e. Calculate the DTFT $|X(\omega)|^2$ of the data x_n over $0 \le \omega \le \pi$. For the chosen order p, calculate the corresponding AR power spectrum but this time use the Burg method to determine the order-p prediction filter. Plot the DTFT and AR spectra on the same graph, but for convenience in comparing them, normalize both spectra to their maximum values. Investigate if higher values of p can model these spectra better, for example, try the orders $p = 10$ and $p = 15$. Some example graphs are included below.









12.43 *Computer Experiment – Wiener Filter Design.* It is desired to design a Wiener filter to enhance a sinusoidal signal buried in noise. The noisy sinusoidal signal is given by

$$x_n = s_n + v_n, \qquad \text{where } s_n = \sin(\omega_0 n)$$

with $\omega_0 = 0.075\pi$. The noise component v_n is related to the secondary signal y_n by

$$v_n = y_n + y_{n-1} + y_{n-2} + y_{n-3} + y_{n-4} + y_{n-5} + y_{n-6}$$

a. Generate $N = 200$ samples of the signal y_n by assuming that it is an AR(4) process with reflection coefficients:

$$\{\gamma_1, \gamma_2, \gamma_3, \gamma_4\} = \{0.5, -0.5, 0.5, -0.5\}$$

The variance σ_ϵ^2 of the driving white noise of the model must be chosen in such a way as to make the variance σ_v^2 of the noise component v_n approximately $\sigma_v^2 = 0.5$, such that the two terms s_n and v_n of x_n have approximately equal strengths, that is, 0 dB signal-to-noise ratio.

(This can be done by writing $v_n = \mathbf{c}^T \mathbf{y}(n)$ and therefore, $\sigma_v^2 = \mathbf{c}^T R \mathbf{c}$, where $\mathbf{c}$ is a 7-dimensional column vector of ones, and R is the order-6 autocorrelation matrix, you can then write $\sigma_v^2 = \sigma_y^2 \mathbf{c}^T R_{\text{norm}} \mathbf{c}$, where R_{norm} has all its entries normalized by

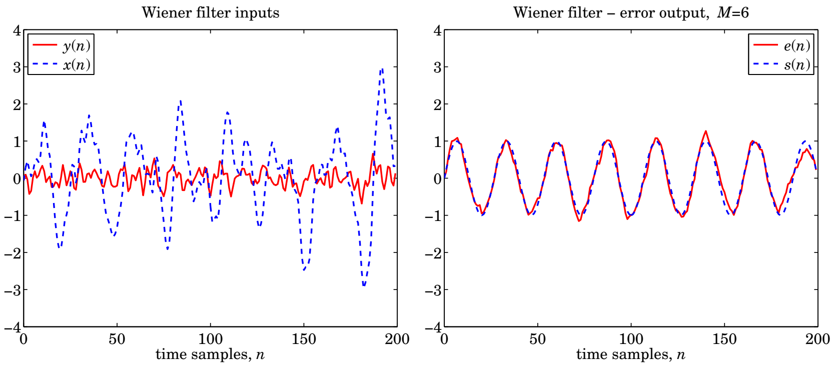
$R(0)=\sigma_y^2$. You can easily determine R_{norm} by doing a maximum entropy extension to order six, starting with the four reflection coefficients and setting $\gamma_5=\gamma_6=0$.)

In generating y_n make sure that the transients introduced by the filter have died out. Then, generate the corresponding N samples of the signal x_n. On the same graph, plot x_n together with the desired signal s_n. On a separate graph (but using the same vertical scales as the previous one) plot the reference signal y_n versus n.

b. For $M=4$, design a Wiener filter of order-M based on the generated signal blocks $\{x_n,y_n\}$, $n=0,1,\dots,N-1$, and realize it in both the direct and lattice forms.

c. Using the *lattice* form, filter the signals x_n,y_n through the designed filter and generate the outputs $\hat{x}_n,e_n$. Explain why e_n should be an estimate of the desired signal s_n. On the same graph, plot e_n and s_n using the same vertical scales as in part (a).

d. Repeat parts (b) and (c) for filter orders $M=5,6,7,8$. Discuss the improvement obtained with increasing order. What is the smallest M that would, at least theoretically, result in $e_n=s_n$? Some example graphs are included below.



13 Kalman Filtering

13.1 State-Space Models

The Kalman filter is based on a state/measurement model of the form:

$$\boxed{\begin{aligned}\mathbf{x}_{n+1} &= A_n\mathbf{x}_n+\mathbf{w}_n \\ \mathbf{y}_n &= C_n\mathbf{x}_n+\mathbf{v}_n\end{aligned}}\qquad\begin{aligned}&\text{(state model)}\\ &\text{(measurement model)}\end{aligned}\tag{13.1.1}$$

where $\mathbf{x}_n$ is a p-dimensional state vector and $\mathbf{y}_n$, an r-dimensional vector of observations. The $p\times p$ state-transition matrix A_n and $r\times p$ measurement matrix C_n may depend on time n. The signals $\mathbf{w}_n,\mathbf{v}_n$ are assumed to be mutually-independent, zero-mean, white-noise signals with known covariance matrices Q_n and R_n:

$$\boxed{\begin{aligned}E[\mathbf{w}_n\mathbf{w}_i^T] &= Q_n\delta_{ni}\\ E[\mathbf{v}_n\mathbf{v}_i^T] &= R_n\delta_{ni}\\ E[\mathbf{w}_n\mathbf{v}_i^T] &= 0\end{aligned}}\tag{13.1.2}$$

The model is iterated starting at $n=0$. The initial state vector $\mathbf{x}_0$ is assumed to be random and independent of $\mathbf{w}_n,\mathbf{v}_n$, but with a known mean $\bar{\mathbf{x}}_0=E[\mathbf{x}_0]$ and covariance matrix $\Sigma_0=E[(\mathbf{x}_0-\bar{\mathbf{x}}_0)(\mathbf{x}_0-\bar{\mathbf{x}}_0)^T]$. We will assume, for now, that $\mathbf{x}_0,\mathbf{w}_n,\mathbf{v}_n$ are normally distributed, and therefore, their statistical description is completely determined by their means and covariances. A non-zero cross-covariance $E[\mathbf{w}_n\mathbf{v}_i^T]=S_n\delta_{ni}$ may also be assumed. A scalar version of the model was discussed in Chap. 11.

The covariance matrices Q_n,R_n have dimensions $p\times p$ and $r\times r$, but they need not have full rank (which would mean that some of the components of $\mathbf{x}_n$ or $\mathbf{y}_n$ would, in an appropriate basis, be noise-free.) For example, to allow the possibility of fewer state noise components, the model (13.1.1) is often written in the form:

$$\begin{aligned}\mathbf{x}_{n+1} &= A_n\mathbf{x}_n+G_n\mathbf{w}_n \\ \mathbf{y}_n &= C_n\mathbf{x}_n+\mathbf{v}_n\end{aligned}\qquad\begin{aligned}&\text{(state model)}\\ &\text{(measurement model)}\end{aligned}\tag{13.1.3}$$

where the new $\mathbf{w}_n$ is lower-dimensional with (full-rank) covariance Q_n. In this model, the covariances of the noise components will be $E[(G_n\mathbf{w}_n)(G_n\mathbf{w}_n)^T]= G_nQ_nG_n^T$, In addition, external deterministic inputs may be present, for example,

$$\begin{aligned} \mathbf{x}_{n+1} &= A_n\mathbf{x}_n + B_n\mathbf{u}_n + G_n\mathbf{w}_n \qquad && \text{(state model)}\\ \mathbf{y}_n &= C_n\mathbf{x}_n + \mathbf{v}_n && \text{(measurement model)} \end{aligned} \tag{13.1.4}$$

where $\mathbf{u}_n$ is the deterministic input. Such modifications do not affect much the essential form of the Kalman filter and, therefore, we will use the simpler model (13.1.1).

The iterated solution of the state equation (13.1.1) may be obtained with the help of the corresponding $p{\times}p$ state-transition matrix $\Phi_{n,k}$ defined as follows:

$$\boxed{\begin{aligned} \Phi_{n,k} &= A_{n-1}\cdots A_k\,, \quad \text{for } n>k\\ \Phi_{n,n} &= I\\ \Phi_{n,k} &= \Phi_{k,n}^{-1}\,, \quad \text{for } n<k \end{aligned}} \tag{13.1.5}$$

where I is the $p{\times}p$ identity matrix and the third equation is valid only if the inverse exists. In particular, we have $\Phi_{n,0} = A_{n-1}\cdots A_0$ for $n > 0$, and $\Phi_{0,0} = I$. If the state matrix A_n is independent of n, then the above definitions imply that $\Phi_{n,k} = A^{n-k}$. We note also the properties:

$$\begin{aligned} \Phi_{n,n-1} &= A_n\,, \quad n\ge 1\\ \Phi_{n+1,k} &= A_n\Phi_{n,k}\,, \quad n\ge k\\ \Phi_{n,k} &= \Phi_{n,i}\Phi_{i,k}\,, \quad n\ge i\ge k \end{aligned} \tag{13.1.6}$$

It is easily verified that the solution of Eq. (13.1.1) is given by:

$$\mathbf{x}_n = \Phi_{n,0}\,\mathbf{x}_0 + \sum_{k=1}^{n}\Phi_{n,k}\mathbf{w}_{k-1}\,, \quad n\ge 1 \tag{13.1.7}$$

so that $\mathbf{x}_n$ depends only on $\{\mathbf{x}_0, \mathbf{w}_0, \mathbf{w}_1, \dots, \mathbf{w}_{n-1}\}$, for example,

$$\begin{aligned} \mathbf{x}_1 &= \Phi_{1,0}\,\mathbf{x}_0 + \Phi_{1,1}\mathbf{w}_0\\ \mathbf{x}_2 &= \Phi_{2,0}\,\mathbf{x}_0 + \Phi_{2,1}\mathbf{w}_0 + \Phi_{2,2}\mathbf{w}_1\\ \mathbf{x}_3 &= \Phi_{3,0}\,\mathbf{x}_0 + \Phi_{3,1}\mathbf{w}_0 + \Phi_{3,2}\mathbf{w}_1 + \Phi_{3,3}\mathbf{w}_2\\ &\;\;\vdots\\ \mathbf{x}_n &= \Phi_{n,0}\,\mathbf{x}_0 + \Phi_{n,1}\mathbf{w}_0 + \Phi_{n,2}\mathbf{w}_1 + \cdots + \Phi_{n,n}\mathbf{w}_{n-1} \end{aligned}$$

and more generally, starting at time $n = i$,

$$\mathbf{x}_n = \Phi_{n,i}\,\mathbf{x}_i + \sum_{k=i+1}^{n}\Phi_{n,k}\mathbf{w}_{k-1}\,, \quad n>i \tag{13.1.8}$$

Let $\bar{\mathbf{x}}_n = E[\mathbf{x}_n]$ and $\Sigma_n = E[(\mathbf{x}_n-\bar{\mathbf{x}}_n)(\mathbf{x}_n-\bar{\mathbf{x}}_n)^T]$ be the mean and covariance matrix of the state vector $\mathbf{x}_n$. Then, it follows from Eqs. (13.1.2) and (13.1.7) and the independence of $\mathbf{x}_0$ and $\mathbf{w}_n$ that,

$$\begin{aligned} \bar{\mathbf{x}}_n &= \Phi_{n,0}\bar{\mathbf{x}}_0\\ \Sigma_n &= \Phi_{n,0}\Sigma_0\Phi_{n,0}^T + \sum_{k=1}^{n}\Phi_{n,k}Q_{k-1}\Phi_{n,k}^T\,, \quad n\ge 1 \end{aligned} \tag{13.1.9}$$

It is straightforward to show from (13.1.9) or (13.1.1) that $\bar{\mathbf{x}}_n$ and Σ_n satisfy the recursions:

$$\begin{aligned} \bar{\mathbf{x}}_{n+1} &= A_n\bar{\mathbf{x}}_n\\ \Sigma_{n+1} &= A_n\Sigma_nA_n^T + Q_n\,, \quad n\ge 1 \end{aligned} \tag{13.1.10}$$

Indeed, subtracting (13.1.1) and (13.1.10) and using the independence of $\mathbf{x}_n$ and $\mathbf{w}_n$, we find:

$$\mathbf{x}_{n+1}-\bar{\mathbf{x}}_{n+1} = A_n(\mathbf{x}_n-\bar{\mathbf{x}}_n)+\mathbf{w}_n$$

$$\Sigma_{n+1} = E[(\mathbf{x}_{n+1}-\bar{\mathbf{x}}_{n+1})(\mathbf{x}_{n+1}-\bar{\mathbf{x}}_{n+1})^T]= E[(A_n(\mathbf{x}_n-\bar{\mathbf{x}}_n)+\mathbf{w}_n)(A_n(\mathbf{x}_n-\bar{\mathbf{x}}_n)+\mathbf{w}_n)^T]$$

$$= A_nE[(\mathbf{x}_n-\bar{\mathbf{x}}_n)(\mathbf{x}_n-\bar{\mathbf{x}}_n)^T]A_n^T + E[\mathbf{w}_n\mathbf{w}_n^T]= A_n\Sigma_nA_n^T + Q_n$$

In a similar fashion, we can obtain the statistical properties of the observations $\mathbf{y}_n$ from those of $\mathbf{x}_n$ and $\mathbf{v}_n$:

$$\begin{aligned} \bar{\mathbf{y}}_n &= C_n\bar{\mathbf{x}}_n\\ \mathbf{y}_n-\bar{\mathbf{y}}_n &= C_n(\mathbf{x}_n-\bar{\mathbf{x}}_n)+\mathbf{v}_n\\ \Sigma_{y_ny_n} &= E[(\mathbf{y}_n-\bar{\mathbf{y}}_n)(\mathbf{y}_n-\bar{\mathbf{y}}_n)^T]= C_n\Sigma_nC_n^T + R_n\,, \quad n\ge 0 \end{aligned} \tag{13.1.11}$$

Example 13.1.1: *Local Level Model.* The local-level model discussed in Chap. 6 is already in state-space form:

$$\begin{aligned} x_{n+1} &= x_n + w_n\\ y_n &= x_n + v_n \end{aligned}$$

and represents a random-walk process x_n observed in noise. The noise variances are defined as $Q = \sigma_w^2$ and $R = \sigma_v^2$. □

Example 13.1.2: *Local Trend Model.* The local-trend model was discussed in Sec. 6.13. Let a_n, b_n be the local level and local slope. The model is defined by,

$$\begin{aligned} a_{n+1} &= a_n + b_n + w_n\\ b_{n+1} &= b_n + u_n\\ y_n &= a_n + v_n \end{aligned}$$

with mutually uncorrelated noise components w_n, u_n, v_n. The model can be written in state-space form as follows:

$$\begin{bmatrix} a_{n+1}\\ b_{n+1} \end{bmatrix} = \begin{bmatrix} 1 & 1\\ 0 & 1 \end{bmatrix}\begin{bmatrix} a_n\\ b_n \end{bmatrix} + \begin{bmatrix} w_n\\ u_n \end{bmatrix}, \quad y_n = [1,0]\begin{bmatrix} a_n\\ b_n \end{bmatrix} + v_n$$

The noise covariances are:

$$\mathbf{w}_n = \begin{bmatrix} w_n\\ u_n \end{bmatrix}, \quad Q = E[\mathbf{w}_n\mathbf{w}_n^T]= \begin{bmatrix} \sigma_w^2 & 0\\ 0 & \sigma_u^2 \end{bmatrix}, \quad R = \sigma_v^2$$

As we mentioned in Sec. 6.13, the steady-state version of the Kalman filter for this model is equivalent to Holt's exponential smoothing method. We demonstrate this later. □

Example 13.1.3: *Kinematic Models for Radar Tracking.* Consider the one-dimensional motion of an object moving with constant acceleration, $\ddot{x}(t)= a$. By integrating this equation, the object's position $x(t)$ and velocity $\dot{x}(t)$ are,

$$\begin{aligned} x(t)&= x(t_0)+(t-t_0)\dot{x}(t_0)+\frac{1}{2}(t-t_0)^2 a \\ \dot{x}(t)&= \dot{x}(t_0)+(t-t_0)a \end{aligned} \qquad (13.1.12)$$

Suppose the motion is sampled at time intervals T, i.e., at the time instants $t_n = nT$, and let us assume that the acceleration is not necessarily constant for all t, but is constant within each interval T, that is, $a(t)= a(t_n)$, for $t_n \le t < t_{n+1}$. Then, applying Eq. (13.1.12) at $t = t_{n+1}$ and $t_0 = t_n$, and denoting $x(t_n)= x_n$, $\dot{x}(t_n)= \dot{x}_n$, and $a(t_n)= a_n$, we obtain,

$$\begin{aligned} x_{n+1} &= x_n + T\dot{x}_n + \frac{1}{2}T^2 a_n \\ \dot{x}_{n+1} &= \dot{x}_n + Ta_n \end{aligned} \qquad (13.1.13)$$

To describe an object that is trying to move at constant speed but is subject to random accelerations, we may assume that a_n is a zero-mean random variable with variance σ_a^2. If the object's position x_n is observed in noise v_n, we obtain the model:

$$\begin{aligned} x_{n+1} &= x_n + T\dot{x}_n + \frac{1}{2}T^2 a_n \\ \dot{x}_{n+1} &= \dot{x}_n + Ta_n \\ y_n &= x_n + v_n \end{aligned} \qquad (13.1.14)$$

which may be written in state-space form:

$$\begin{aligned} \begin{bmatrix} x_{n+1} \\ \dot{x}_{n+1} \end{bmatrix} &= \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_n \\ \dot{x}_n \end{bmatrix} + \begin{bmatrix} T^2/2 \\ T \end{bmatrix} a_n \\ y_n &= [1,0] \begin{bmatrix} x_n \\ \dot{x}_n \end{bmatrix} + v_n \end{aligned} \qquad (13.1.15)$$

with measurement noise variance $R = \sigma_v^2$, and state noise vector and covariance matrix:

$$\mathbf{w}_n = \begin{bmatrix} T^2/2 \\ T \end{bmatrix} a_n \quad \Rightarrow \quad Q = E[\mathbf{w}_n \mathbf{w}_n^T]= \begin{bmatrix} T^4/4 & T^3/2 \\ T^3/2 & T^2 \end{bmatrix} \sigma_a^2 \qquad (13.1.16)$$

This model is, of course, very similar to the local-trend model of the previous example if we set $T = 1$, except now the state noise arises from a single acceleration noise a_n affecting both components of the state vector, whereas in the local-trend model, we assumed independent noises for the local level and local slope.

We will see later that the steady-state Kalman filter for the model defined by Eqs. (13.1.15) and (13.1.16) is equivalent to an optimum version of the popular α–β radar tracking filter [868,869]. An alternative model, which leads to a somewhat different α–β tracking model [870,874], has state noise added only to the velocity component:

$$\begin{aligned} \begin{bmatrix} x_{n+1} \\ \dot{x}_{n+1} \end{bmatrix} &= \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_n \\ \dot{x}_n \end{bmatrix} + \begin{bmatrix} 0 \\ w_n \end{bmatrix} \\ y_n &= [1,0] \begin{bmatrix} x_n \\ \dot{x}_n \end{bmatrix} + v_n \end{aligned} \qquad (13.1.17)$$

with $R = \sigma_v^2$ and

$$\mathbf{w}_n = \begin{bmatrix} 0 \\ w_n \end{bmatrix}, \quad Q = E[\mathbf{w}_n \mathbf{w}_n^T]= \begin{bmatrix} 0 & 0 \\ 0 & \sigma_w^2 \end{bmatrix}$$

The models (13.1.15) and (13.1.17) are appropriate for uniformly moving objects subject to random accelerations. In order to describe a maneuvering, accelerating, object, we may start with the model (13.1.15) and make the acceleration a_n part of the state vector and assume that it deviates from a constant acceleration by an additive white noise term, i.e., replace a_n by $a_n + w_n$. Denoting a_n by $\ddot{x}_n$, we obtain the model [874,875]:

$$\begin{aligned} x_{n+1} &= x_n + T\dot{x}_n + \frac{1}{2}T^2(\ddot{x}_n + w_n) \\ \dot{x}_{n+1} &= \dot{x}_n + T(\ddot{x}_n + w_n) \\ \ddot{x}_{n+1} &= \ddot{x}_n + w_n \\ y_n &= x_n + v_n \end{aligned} \qquad (13.1.18)$$

which may be written in the matrix form:

$$\begin{aligned} \begin{bmatrix} x_{n+1} \\ \dot{x}_{n+1} \\ \ddot{x}_{n+1} \end{bmatrix} &= \begin{bmatrix} 1 & T & T^2/2 \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_n \\ \dot{x}_n \\ \ddot{x}_n \end{bmatrix} + \begin{bmatrix} T^2/2 \\ T \\ 1 \end{bmatrix} w_n \\ y_n &= [1,0,0] \begin{bmatrix} x_n \\ \dot{x}_n \\ \ddot{x}_n \end{bmatrix} + v_n \end{aligned} \qquad (13.1.19)$$

This leads to the so-called α–β–γ tracking filter. An alternative model may be derived by starting with a linearly increasing acceleration $\ddot{x}(t)= a(t)= a(t_0)+(t-t_0)\dot{a}(t_0)$, whose integration gives:

$$\begin{aligned} x(t)&= x(t_0)+(t-t_0)u(t_0)+\frac{1}{2}(t-t_0)^2 a(t_0)+\frac{1}{6}(t-t_0)^3\dot{a}(t_0) \\ \dot{x}(t)&= \dot{x}(t_0)+(t-t_0)a(t_0)+\frac{1}{2}(t-t_0)^2\dot{a}(t_0) \\ a(t)&= a(t_0)+(t-t_0)\dot{a}(t_0) \end{aligned} \qquad (13.1.20)$$

Its sampled version is obtained by treating the acceleration rate $\dot{a}_n$ as a zero-mean white-noise term with variance $\sigma_{\dot{a}}^2$, resulting in the state model [876]:

$$\begin{aligned} \begin{bmatrix} x_{n+1} \\ \dot{x}_{n+1} \\ \ddot{x}_{n+1} \end{bmatrix} &= \begin{bmatrix} 1 & T & T^2/2 \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_n \\ \dot{x}_n \\ \ddot{x}_n \end{bmatrix} + \begin{bmatrix} T^3/3 \\ T^2/2 \\ T \end{bmatrix} \dot{a}_n \\ y_n &= [1,0,0] \begin{bmatrix} x_n \\ \dot{x}_n \\ \ddot{x}_n \end{bmatrix} + v_n \end{aligned} \qquad (13.1.21)$$

Later on we will look at the Kalman filters for such kinematic models and discuss their connection to the α–β and α–β–γ tracking filters. □

13.2 Kalman Filter

The Kalman filter is a time-recursive procedure for estimating the state vector $\mathbf{x}_n$ from the observations signal $\mathbf{y}_n$. Let $Y_n = \{\mathbf{y}_0, \mathbf{y}_1, \dots, \mathbf{y}_n\}$ be the linear span of the observations up to the current time instant n. The Kalman filter estimate of $\mathbf{x}_n$ is defined as the optimum linear combination of these observations that minimizes the mean-square estimation error, and as we already know, it is given by the projection of $\mathbf{x}_n$ onto the observation subspace Y_n. For the gaussian case, this projection happens to be the conditional mean $\hat{\mathbf{x}}_{n/n} = E[\mathbf{x}_n|Y_n]$. Let us define also the predicted estimate $\hat{\mathbf{x}}_{n/n-1}$ based on the observations $Y_{n-1} = \{\mathbf{y}_0, \mathbf{y}_1, \dots, \mathbf{y}_{n-1}\}$. Again, for the gaussian case we have $\hat{\mathbf{x}}_{n/n-1} = E[\mathbf{x}_n|Y_{n-1}]$. To cover both the gaussian and nongaussian, but linear, cases, we will use the following notation for the estimates, estimation errors, and mean-square error covariances:

$$\boxed{\begin{aligned} \hat{\mathbf{x}}_{n/n-1} &= \mathrm{Proj}[\mathbf{x}_n|Y_{n-1}] \\ \mathbf{e}_{n/n-1} &= \mathbf{x}_n - \hat{\mathbf{x}}_{n/n-1} \\ P_{n/n-1} &= E[\mathbf{e}_{n/n-1}\mathbf{e}_{n/n-1}^T] \end{aligned}} \quad \text{and} \quad \boxed{\begin{aligned} \hat{\mathbf{x}}_{n/n} &= \mathrm{Proj}[\mathbf{x}_n|Y_n] \\ \mathbf{e}_{n/n} &= \mathbf{x}_n - \hat{\mathbf{x}}_{n/n} \\ P_{n/n} &= E[\mathbf{e}_{n/n}\mathbf{e}_{n/n}^T] \end{aligned}} \tag{13.2.1}$$

We will first state the estimation algorithm, and then prove it. The Kalman filtering algorithm for the model (13.1.1)–(13.1.2) is as follows:

```
Initialize in time by:  $\hat{\mathbf{x}}_{0/-1} = \bar{\mathbf{x}}_0$,  $P_{0/-1} = \Sigma_0$
At time $n$, $\hat{\mathbf{x}}_{n/n-1}$, $P_{n/n-1}$, $\mathbf{y}_n$ are available,
    $D_n = C_n P_{n/n-1} C_n^T + R_n$                                innovations covariance
    $G_n = P_{n/n-1} C_n^T D_n^{-1}$                                  Kalman gain for filtering
    $K_n = A_n G_n = A_n P_{n/n-1} C_n^T D_n^{-1}$                    Kalman gain for prediction
    $\hat{\mathbf{y}}_{n/n-1} = C_n \hat{\mathbf{x}}_{n/n-1}$                                predicted measurement
    $\boldsymbol{\varepsilon}_n = \mathbf{y}_n - \hat{\mathbf{y}}_{n/n-1} = \mathbf{y}_n - C_n \hat{\mathbf{x}}_{n/n-1}$             innovations sequence
Measurement update / correction:
    $\hat{\mathbf{x}}_{n/n} = \hat{\mathbf{x}}_{n/n-1} + G_n \boldsymbol{\varepsilon}_n$                              filtered estimate
    $P_{n/n} = P_{n/n-1} - G_n D_n G_n^T$                               estimation error
Time update / prediction:
    $\hat{\mathbf{x}}_{n+1/n} = A_n \hat{\mathbf{x}}_{n/n} = A_n \hat{\mathbf{x}}_{n/n-1} + K_n \boldsymbol{\varepsilon}_n$         predicted estimate
    $P_{n+1/n} = A_n P_{n/n} A_n^T + Q_n$                               prediction error
Go to time $n+1$
```

(13.2.2)

The quantity D_n represents the innovations covariance matrix, that is, $D_n = E[\boldsymbol{\varepsilon}_n \boldsymbol{\varepsilon}_n^T]$. The innovations sequence $\{\boldsymbol{\varepsilon}_0, \boldsymbol{\varepsilon}_1, \dots, \boldsymbol{\varepsilon}_n\}$, constructed recursively by the algorithm, represents the Gram-Schmidt orthogonalization of the observations $\{\mathbf{y}_0, \mathbf{y}_1, \dots, \mathbf{y}_n\}$ in the sense that the innovations form an orthogonal basis for the observation subspace Y_n, that is, Y_n is the linear span of either set:

$$Y_n = \{\mathbf{y}_0, \mathbf{y}_1, \dots, \mathbf{y}_n\} = \{\boldsymbol{\varepsilon}_0, \boldsymbol{\varepsilon}_1, \dots, \boldsymbol{\varepsilon}_n\}$$

The orthogonality property of $\boldsymbol{\varepsilon}_n$ is expressed by:

$$\boxed{E[\boldsymbol{\varepsilon}_n \boldsymbol{\varepsilon}_i^T] = D_n \delta_{ni}} \tag{13.2.3}$$

There are some alternative ways of writing the above equations. For example, the equation for $P_{n/n}$ may be written in the equivalent ways:

$$\boxed{\begin{aligned} &1.\ \ P_{n/n} = P_{n/n-1} - G_n D_n G_n^T = (I - G_n C_n) P_{n/n-1} \\ &2.\ \ P_{n/n} = P_{n/n-1} - P_{n/n-1} C_n^T D_n^{-1} C_n P_{n/n-1} = \text{standard form} \\ &3.\ \ P_{n/n} = (I - G_n C_n) P_{n/n-1} (I - G_n C_n)^T + G_n R_n G_n^T = \text{Joseph form} \\ &4.\ \ P_{n/n} = \big[P_{n/n-1}^{-1} + C_n^T R_n^{-1} C_n\big]^{-1} = \text{information form} \end{aligned}} \tag{13.2.4}$$

with $D_n = C_n P_{n/n-1} C_n^T + R_n$ and $G_n = P_{n/n-1} C_n^T D_n^{-1}$. Similarly, we can write the Kalman gain G_n in its information form:

$$\boxed{G_n = P_{n/n-1} C_n^T D_n^{-1} = P_{n/n} C_n^T R_n^{-1}} \tag{13.2.5}$$

It follows from the information forms that the filtered estimate may be re-expressed as:

$$\begin{aligned} \hat{\mathbf{x}}_{n/n} &= \hat{\mathbf{x}}_{n/n-1} + G_n \boldsymbol{\varepsilon}_n = \hat{\mathbf{x}}_{n/n-1} + P_{n/n} C_n^T R_n^{-1} (\mathbf{y}_n - C_n \hat{\mathbf{x}}_{n/n-1}) \\ &= P_{n/n} [P_{n/n}^{-1} - C_n^T R_n^{-1} C_n] \hat{\mathbf{x}}_{n/n-1} + P_{n/n} C_n^T R_n^{-1} \mathbf{y}_n \\ &= P_{n/n} P_{n/n-1}^{-1} \hat{\mathbf{x}}_{n/n-1} + P_{n/n} C_n^T R_n^{-1} \mathbf{y}_n \end{aligned}$$

from which we obtain the information form of the updating equation:

$$\boxed{P_{n/n}^{-1} \hat{\mathbf{x}}_{n/n} = P_{n/n-1}^{-1} \hat{\mathbf{x}}_{n/n-1} + C_n^T R_n^{-1} \mathbf{y}_n} \tag{13.2.6}$$

In the relations that involve R_n^{-1}, one must assume that R_n has full rank. The difference equation for the predicted estimate may be written directly in terms of the current observation $\mathbf{y}_n$ and the *closed-loop* state matrix $F_n = A_n - K_n C_n$, as follows:

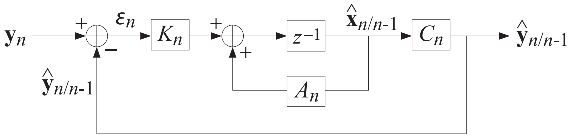
$$\begin{aligned} \hat{\mathbf{x}}_{n+1/n} &= A_n \hat{\mathbf{x}}_{n/n-1} + K_n \boldsymbol{\varepsilon}_n = A_n \hat{\mathbf{x}}_{n/n-1} + K_n (\mathbf{y}_n - C_n \hat{\mathbf{x}}_{n/n-1}) \\ &= (A_n - K_n C_n) \hat{\mathbf{x}}_{n/n-1} + K_n \mathbf{y}_n \end{aligned}$$

that is,

$$\boxed{\hat{\mathbf{x}}_{n+1/n} = (A_n - K_n C_n) \hat{\mathbf{x}}_{n/n-1} + K_n \mathbf{y}_n} \tag{13.2.7}$$

A block diagram realization is depicted in Fig. 13.2.1. The error covariance update equations,

$$\begin{aligned} P_{n/n} &= P_{n/n-1} - P_{n/n-1} C_n^T D_n^{-1} C_n P_{n/n-1} \\ P_{n+1/n} &= A_n P_{n/n} A_n^T + Q_n \end{aligned}$$



$\mathbf{y}_n$
+
ε_n
K_n
+
z^{-1}
$\hat{\mathbf{x}}_{n/n-1}$
C_n
$\hat{\mathbf{y}}_{n/n-1}$
−
+
A_n
$\hat{\mathbf{y}}_{n/n-1}$

Fig. 13.2.1 Kalman filter realization.

may be combined into a single equation known as the discrete-time *Riccati difference* equation, to be initialized at $P_{0/-1} = \Sigma_0$:

$$\boxed{P_{n+1/n} = A_n\Big[P_{n/n-1} - P_{n/n-1}C_n^T(C_nP_{n/n-1}C_n^T + R_n)^{-1}C_nP_{n/n-1}\Big]A_n^T + Q_n} \tag{13.2.8}$$

which can also be written in the "information" forms (if R_n^{-1} exists):

$$\boxed{\begin{aligned} P_{n+1/n} &= A_n[P_{n/n-1}^{-1} + C_n^TR_n^{-1}C_n)]^{-1}A_n^T + Q_n \\ P_{n+1/n} &= A_n[I + P_{n/n-1}C_n^TR_n^{-1}C_n)]^{-1}P_{n/n-1}A_n^T + Q_n \end{aligned}} \tag{13.2.9}$$

and in the Joseph-like forms:

$$\boxed{\begin{aligned} P_{n+1/n} &= A_nP_{n/n-1}A_n^T + Q_n - K_nD_nK_n^T\,, \quad K_n = A_nP_{n/n-1}C_n^TD_n^{-1} \\ P_{n+1/n} &= (A_n - K_nC_n)P_{n/n-1}(A_n - K_nC_n)^T + K_nR_nK_n^T + Q_n \end{aligned}} \tag{13.2.10}$$

Similarly, the closed-loop transition matrix can be written as,

$$\boxed{F_n = A_n - K_nC_n = A_n[I + P_{n/n-1}C_n^TR_n^{-1}C_n]^{-1}} \tag{13.2.11}$$

We note also that since, $\hat{\mathbf{x}}_{n+1/n} = A_n\hat{\mathbf{x}}_{n/n}$ and $\hat{\mathbf{x}}_{n/n-1} = A_{n-1}\hat{\mathbf{x}}_{n-1/n-1}$, the difference equations for the predicted and filtered estimates would read as follows in terms of the Kalman gains K_n and G_n, respectively,

$$\boxed{\begin{array}{ll} \hat{\mathbf{y}}_{n/n-1} = C_n\hat{\mathbf{x}}_{n/n-1} = C_nA_{n-1}\hat{\mathbf{x}}_{n-1/n-1} & \text{predicted measurement} \\ \boldsymbol{\varepsilon}_n = \mathbf{y}_n - \hat{\mathbf{y}}_{n/n-1} & \text{innovations sequence} \\ \hat{\mathbf{x}}_{n+1/n} = A_n\hat{\mathbf{x}}_{n/n-1} + K_n\boldsymbol{\varepsilon}_n & \text{predicted estimate} \\ \hat{\mathbf{x}}_{n/n} = A_{n-1}\hat{\mathbf{x}}_{n-1/n-1} + G_n\boldsymbol{\varepsilon}_n & \text{filtered estimate} \end{array}} \tag{13.2.12}$$

13.3 Derivation

To derive Eq. (13.2.2), we recall from Chap. 1 that the optimum linear estimate of a zero-mean random vector $\mathbf{x}$ based on a zero-mean random vector $\mathbf{y}$, and the corresponding

orthogonality condition and error covariance, are given by:

$$\begin{aligned} \hat{\mathbf{x}} &= R_{xy}R_{yy}^{-1}\mathbf{y} = E[\mathbf{x}\mathbf{y}^T]E[\mathbf{y}\mathbf{y}^T]^{-1}\mathbf{y} \\ \mathbf{e} &= \mathbf{x} - \hat{\mathbf{x}}\,, \quad R_{ey} = E[\mathbf{e}\mathbf{y}^T] = 0 \\ R_{ee} &= E[\mathbf{e}\mathbf{e}^T] = R_{xx} - R_{xy}R_{yy}^{-1}R_{yx} \end{aligned} \tag{13.3.1}$$

When the vectors have non-zero means, say $\bar{\mathbf{x}}, \bar{\mathbf{y}}$, the same results apply with the replacement $\mathbf{x} \to \mathbf{x} - \bar{\mathbf{x}}$, $\mathbf{y} \to \mathbf{y} - \bar{\mathbf{y}}$, and $\hat{\mathbf{x}} \to \hat{\mathbf{x}} - \bar{\mathbf{x}}$ in (13.3.1). Under this replacement, the correlation matrices are replaced by the corresponding covariances, e.g.,

$$R_{xy} = E[\mathbf{x}\mathbf{y}^T] \to E[(\mathbf{x} - \bar{\mathbf{x}})(\mathbf{y} - \bar{\mathbf{y}})^T] = \Sigma_{xy}\,, \quad \text{etc.}$$

Hence, the optimum estimate is now:

$$\begin{aligned} \hat{\mathbf{x}} &= \bar{\mathbf{x}} + \Sigma_{xy}\Sigma_{yy}^{-1}(\mathbf{y} - \bar{\mathbf{y}}) \\ \mathbf{e} &= \mathbf{x} - \hat{\mathbf{x}}\,, \quad R_{ey} = \Sigma_{ey} = E[\mathbf{e}\mathbf{y}^T] = 0 \\ \Sigma_{ee} &= \Sigma_{xx} - \Sigma_{xy}\Sigma_{yy}^{-1}\Sigma_{yx} \end{aligned} \tag{13.3.2}$$

We note that the estimate is unbiased, that is, $E[\hat{\mathbf{x}}] = \bar{\mathbf{x}}$, and therefore, the estimation error has zero mean, $E[\mathbf{e}] = 0$, its covariance matrix will be $\Sigma_{ee} = R_{ee} = E[\mathbf{e}\mathbf{e}^T]$, and the orthogonality condition may be written as $E[\mathbf{e}(\mathbf{y} - \bar{\mathbf{y}})^T] = E[\mathbf{e}\mathbf{y}^T] = 0$.

Let us apply now this result to the state model (13.1.1) at $n = 0$. This will also clarify the manner in which the algorithm is to be initialized. As part of the model, we assume that the initial state $\mathbf{x}_0$ has a known mean $\bar{\mathbf{x}}_0$ and covariance matrix Σ_0. According to (13.3.2), the optimum estimate $\hat{\mathbf{x}}_{0/0}$ of $\mathbf{x}_0$ based on the observation $\mathbf{y}_0$ will be given by:

$$\begin{aligned} \hat{\mathbf{x}}_{0/0} &= \bar{\mathbf{x}}_0 + \Sigma_{x_0y_0}\Sigma_{y_0y_0}^{-1}(\mathbf{y}_0 - \bar{\mathbf{y}}_0) \\ \mathbf{e}_{0/0} &= \mathbf{x}_0 - \hat{\mathbf{x}}_{0/0}\,, \quad E[\mathbf{e}_{0/0}(\mathbf{y}_0 - \bar{\mathbf{y}}_0)^T] = 0 \\ \Sigma_{e_0e_0} &= E[\mathbf{e}_{0/0}\mathbf{e}_{0/0}^T] = \Sigma_{x_0x_0} - \Sigma_{x_0y_0}\Sigma_{y_0y_0}^{-1}\Sigma_{y_0x_0} \end{aligned} \tag{13.3.3}$$

with $E[\hat{\mathbf{x}}_{0/0}] = \bar{\mathbf{x}}_0$ and $E[\mathbf{e}_{0/0}] = 0$. Let us define $\boldsymbol{\varepsilon}_0 = \mathbf{y}_0 - \bar{\mathbf{y}}_0$ and $G_0 = \Sigma_{x_0y_0}\Sigma_{y_0y_0}^{-1}$. From the measurement model $\mathbf{y}_0 = C_0\mathbf{x}_0 + \mathbf{v}_0$, we have $\bar{\mathbf{y}}_0 = C_0\bar{\mathbf{x}}_0$, which gives:

$$\boldsymbol{\varepsilon}_0 = \mathbf{y}_0 - \bar{\mathbf{y}}_0 = C_0(\mathbf{x}_0 - \bar{\mathbf{x}}_0) + \mathbf{v}_0$$

Clearly, $E[\boldsymbol{\varepsilon}_0] = 0$. Since $\mathbf{v}_0$ is uncorrelated with $\mathbf{x}_0$, we will have:

$$E[\boldsymbol{\varepsilon}_0\boldsymbol{\varepsilon}_0^T] = C_0E[(\mathbf{x}_0 - \bar{\mathbf{x}}_0)(\mathbf{x}_0 - \bar{\mathbf{x}}_0)^T]C_0^T + E[\mathbf{v}_0\mathbf{v}_0^T]\,, \quad \text{or,}$$

$$D_0 = E[\boldsymbol{\varepsilon}_0\boldsymbol{\varepsilon}_0^T] = \Sigma_{y_0y_0} = C_0\Sigma_0C_0^T + R_0$$

where $\Sigma_0 = E[(\mathbf{x}_0 - \bar{\mathbf{x}}_0)(\mathbf{x}_0 - \bar{\mathbf{x}}_0)^T] = \Sigma_{x_0x_0}$. Similarly, we find:

$$\Sigma_{x_0y_0} = E[(\mathbf{x}_0 - \bar{\mathbf{x}}_0)\boldsymbol{\varepsilon}_0^T] = E[(\mathbf{x}_0 - \bar{\mathbf{x}}_0)(\mathbf{x}_0 - \bar{\mathbf{x}}_0)^T]C_0^T = \Sigma_0C_0^T$$

Thus, the Kalman gain becomes,

$$G_0 = \Sigma_{x_0y_0}\Sigma_{y_0y_0}^{-1} = \Sigma_0 C_0^T D_0^{-1}$$

With the definitions $\hat{\mathbf{x}}_{0/-1} = \bar{\mathbf{x}}_0$ and $P_{0/-1} = \Sigma_0$, we may rewrite (13.3.3) as,

$$\hat{\mathbf{x}}_{0/0} = \hat{\mathbf{x}}_{0/-1} + G_0\boldsymbol{\varepsilon}_0\,, \quad G_0 = P_{0/-1}C_0^T D_0^{-1} \tag{13.3.4}$$

The corresponding error covariance matrix will be given by (13.3.3):

$$\Sigma_{e_0e_0} = \Sigma_{x_0x_0} - \Sigma_{x_0y_0}\Sigma_{y_0y_0}^{-1}\Sigma_{y_0x_0} = \Sigma_0 - \Sigma_0 C_0^T D_0^{-1} C_0 \Sigma_0\,, \quad \text{or,}$$

$$P_{0/0} = P_{0/-1} - P_{0/-1}C_0^T D_0^{-1} C_0 P_{0/-1} = P_{0/-1} - G_0 D_0 G_0^T \tag{13.3.5}$$

Because $\mathbf{e}_{0/0}$ has zero mean, we may write the orthogonality condition in Eq. (13.3.3) as,

$$E[\mathbf{e}_{0/0}\mathbf{y}_0^T] = E[\mathbf{e}_{0/0}(\mathbf{y}_0 - \bar{\mathbf{y}}_0)^T] = E[\mathbf{e}_{0/0}\boldsymbol{\varepsilon}_0^T] = 0$$

which states that the estimation error is orthogonal to the observation $\mathbf{y}_0$, or equivalently, to the innovations vector $\boldsymbol{\varepsilon}_0$. To complete the $n = 0$ step of the algorithm, we must now determine the prediction of $\mathbf{x}_1$ based on $\mathbf{y}_0$ or $\boldsymbol{\varepsilon}_0$. We may apply (13.3.2) again,

$$\begin{aligned}
&\hat{\mathbf{x}}_{1/0} = \bar{\mathbf{x}}_1 + \Sigma_{x_1y_0}\Sigma_{y_0y_0}^{-1}(\mathbf{y}_0 - \bar{\mathbf{y}}_0) = \bar{\mathbf{x}}_1 + E[\mathbf{x}_1\boldsymbol{\varepsilon}_0]E[\boldsymbol{\varepsilon}_0\boldsymbol{\varepsilon}_0^T]^{-1}\boldsymbol{\varepsilon}_0 \\
&\mathbf{e}_{1/0} = \mathbf{x}_1 - \hat{\mathbf{x}}_{1/0}\,, \quad E[\mathbf{e}_{1/0}(\mathbf{y}_0 - \bar{\mathbf{y}}_0)^T] = E[\mathbf{e}_{1/0}\boldsymbol{\varepsilon}_0^T] = 0 \\
&\Sigma_{e_1e_1} = E[\mathbf{e}_{1/0}\mathbf{e}_{1/0}^T] = \Sigma_{x_1x_1} - \Sigma_{x_1y_0}\Sigma_{y_0y_0}^{-1}\Sigma_{y_0x_1}
\end{aligned} \tag{13.3.6}$$

From the state equation $\mathbf{x}_1 = A_0\mathbf{x}_0 + \mathbf{w}_0$ and the independence of $\mathbf{w}_0$ and $\mathbf{y}_0$, we find,

$$\begin{aligned}
\Sigma_{x_1y_0} &= \Sigma_{(A_0x_0+w_0)y_0} = A_0\Sigma_{x_0y_0} \\
K_0 &\equiv \Sigma_{x_1y_0}\Sigma_{y_0y_0}^{-1} = A_0\Sigma_{x_0y_0}\Sigma_{y_0y_0}^{-1} = A_0G_0 \\
\Sigma_{x_1x_1} &= \Sigma_{(A_0x_0+w_0)(A_0x_0+w_0)} = A_0\Sigma_{x_0x_0}A_0^T + Q_0 \\
P_{1/0} &= \Sigma_{e_1e_1} = \Sigma_{x_1x_1} - \Sigma_{x_1y_0}\Sigma_{y_0y_0}^{-1}\Sigma_{y_0x_1} \\
&= A_0\Sigma_{x_0x_0}A_0^T + Q_0 - A_0\Sigma_{x_0y_0}\Sigma_{y_0y_0}^{-1}\Sigma_{y_0x_0}A_0^T \\
&= A_0[\Sigma_{x_0x_0} - \Sigma_{x_0y_0}\Sigma_{y_0y_0}^{-1}\Sigma_{y_0x_0}]A_0^T + Q_0 = A_0P_{0/0}A_0^T + Q_0
\end{aligned}$$

Since, $\bar{\mathbf{x}}_1 = A_0\bar{\mathbf{x}}_0 = A_0\hat{\mathbf{x}}_{0/-1}$, we may rewrite the predicted estimate and its error as,

$$\begin{aligned}
\hat{\mathbf{x}}_{1/0} &= A_0\hat{\mathbf{x}}_{0/-1} + K_0\boldsymbol{\varepsilon}_0 = A_0[\hat{\mathbf{x}}_{0/-1} + G_0\boldsymbol{\varepsilon}_0] = A_0\hat{\mathbf{x}}_{0/0} \\
P_{1/0} &= A_0P_{0/0}A_0^T + Q_0
\end{aligned} \tag{13.3.7}$$

This completes all the steps at $n = 0$. We collect the results together:

$$\boxed{\begin{aligned}
&\hat{\mathbf{x}}_{0/-1} = \bar{\mathbf{x}}_0\,, \quad P_{0/-1} = \Sigma_0 \\
&D_0 = C_0P_{0/-1}C_0^T + R_0 \\
&G_0 = P_{0/-1}C_0^T D_0^{-1} \\
&K_0 = A_0G_0 = A_0P_{0/-1}C_0^T D_0^{-1} \\
&\hat{\mathbf{y}}_{0/-1} = C_0\hat{\mathbf{x}}_{0/-1} \\
&\boldsymbol{\varepsilon}_0 = \mathbf{y}_0 - \hat{\mathbf{y}}_{0/-1} = \mathbf{y}_0 - C_0\hat{\mathbf{x}}_{0/-1} \\
&\hat{\mathbf{x}}_{0/0} = \hat{\mathbf{x}}_{0/-1} + G_0\boldsymbol{\varepsilon}_0 \\
&P_{0/0} = P_{0/-1} - G_0D_0G_0^T \\
&\hat{\mathbf{x}}_{1/0} = A_0\hat{\mathbf{x}}_{0/0} = A_0\hat{\mathbf{x}}_{0/-1} + K_0\boldsymbol{\varepsilon}_0 \\
&P_{1/0} = A_0P_{0/0}A_0^T + Q_0
\end{aligned}}$$

Moving on to $n = 1$, we construct the next innovations vector $\boldsymbol{\varepsilon}_1$ by:

$$\boldsymbol{\varepsilon}_1 = \mathbf{y}_1 - \hat{\mathbf{y}}_{1/0} = \mathbf{y}_1 - C_1\hat{\mathbf{x}}_{1/0} \tag{13.3.8}$$

Since $\mathbf{y}_1 = C_1\mathbf{x}_1 + \mathbf{v}_1$, it follows that,

$$\boldsymbol{\varepsilon}_1 = C_1(\mathbf{x}_1 - \hat{\mathbf{x}}_{1/0}) + \mathbf{v}_1 = C_1\mathbf{e}_{1/0} + \mathbf{v}_1 \tag{13.3.9}$$

Because $\boldsymbol{\varepsilon}_0$ is orthogonal to $\mathbf{e}_{1/0}$ and $\mathbf{v}_1$ is independent of $\mathbf{y}_0$, we have:

$$E[\boldsymbol{\varepsilon}_1\boldsymbol{\varepsilon}_0^T] = 0 \tag{13.3.10}$$

We also have $E[\boldsymbol{\varepsilon}_1] = 0$ and the covariance matrix:

$$D_1 = E[\boldsymbol{\varepsilon}_1\boldsymbol{\varepsilon}_1^T] = C_1P_{1/0}C_1^T + R_1 \tag{13.3.11}$$

Thus, the zero-mean vectors $\{\boldsymbol{\varepsilon}_0, \boldsymbol{\varepsilon}_1\}$ form an orthogonal basis for the subspace $Y_1 = \{\mathbf{y}_0, \mathbf{y}_1\}$. The optimum estimate of $\mathbf{x}_1$ based on Y_1 is obtained by Eq. (13.3.2), but with $\mathbf{y}$ replaced by the extended basis vector $\begin{bmatrix} \boldsymbol{\varepsilon}_0 \\ \boldsymbol{\varepsilon}_1 \end{bmatrix}$ whose covariance matrix is diagonal. It follows that,

$$\begin{aligned}
\hat{\mathbf{x}}_{1/1} &= \text{Proj}[\mathbf{x}_1|Y_1] = \text{Proj}[\mathbf{x}_1|\boldsymbol{\varepsilon}_0, \boldsymbol{\varepsilon}_1] = \bar{\mathbf{x}}_1 + E\Big[\mathbf{x}_1[\boldsymbol{\varepsilon}_0^T, \boldsymbol{\varepsilon}_1^T]\Big]\begin{bmatrix} E[\boldsymbol{\varepsilon}_0\boldsymbol{\varepsilon}_0^T] & 0 \\ 0 & E[\boldsymbol{\varepsilon}_1\boldsymbol{\varepsilon}_1^T] \end{bmatrix}^{-1}\begin{bmatrix} \boldsymbol{\varepsilon}_0 \\ \boldsymbol{\varepsilon}_1 \end{bmatrix} \\
&= \bar{\mathbf{x}}_1 + E[\mathbf{x}_1\boldsymbol{\varepsilon}_0]E[\boldsymbol{\varepsilon}_0\boldsymbol{\varepsilon}_0^T]^{-1}\boldsymbol{\varepsilon}_0 + E[\mathbf{x}_1\boldsymbol{\varepsilon}_1]E[\boldsymbol{\varepsilon}_1\boldsymbol{\varepsilon}_1^T]^{-1}\boldsymbol{\varepsilon}_1
\end{aligned} \tag{13.3.12}$$

The first two terms are recognized from Eq. (13.3.6) to be the predicted estimate $\hat{\mathbf{x}}_{1/0}$. Therefore, we have,

$$\hat{\mathbf{x}}_{1/1} = \hat{\mathbf{x}}_{1/0} + E[\mathbf{x}_1\boldsymbol{\varepsilon}_1]E[\boldsymbol{\varepsilon}_1\boldsymbol{\varepsilon}_1^T]^{-1}\boldsymbol{\varepsilon}_1 = \hat{\mathbf{x}}_{1/0} + G_1\boldsymbol{\varepsilon}_1 \tag{13.3.13}$$

Since $\boldsymbol{\varepsilon}_1 \perp \boldsymbol{\varepsilon}_0$, we have $E[\hat{\mathbf{x}}_{1/0}\boldsymbol{\varepsilon}_1^T] = 0$, and using (13.3.9) we may write:

$$E[\mathbf{x}_1\boldsymbol{\varepsilon}_1] = E[(\mathbf{x}_1 - \hat{\mathbf{x}}_{1/0})\boldsymbol{\varepsilon}_1^T] = E[\mathbf{e}_{1/0}\boldsymbol{\varepsilon}_1^T] = E[\mathbf{e}_{1/0}(\mathbf{e}_{1/0}^T C_1^T + \mathbf{v}_1^T)] = P_{1/0}C_1^T$$

Thus, the Kalman gain for the filtered estimate is:

$$G_1 = E[\mathbf{x}_1\boldsymbol{\varepsilon}_1]E[\boldsymbol{\varepsilon}_1\boldsymbol{\varepsilon}_1^T]^{-1} = P_{1/0}C_1^T D_1^{-1}$$

The corresponding error covariance matrix can be obtained from Eq. (13.3.2), but perhaps a faster way is to argue as follows. Using Eq. (13.3.13), we have

$$\mathbf{e}_{1/1} = \mathbf{x}_1 - \hat{\mathbf{x}}_{1/1} = \mathbf{x}_1 - \hat{\mathbf{x}}_{1/0} - G_1\boldsymbol{\varepsilon}_1 = \mathbf{e}_{1/0} - G_1\boldsymbol{\varepsilon}_1\,, \quad \text{or,}$$

$$\mathbf{e}_{1/0} = \mathbf{e}_{1/1} + G_1\boldsymbol{\varepsilon}_1 \tag{13.3.14}$$

The orthogonality conditions for the estimate $\hat{\mathbf{x}}_{1/1}$ are $E[\mathbf{e}_{1/1}\boldsymbol{\varepsilon}_0^T] = E[\mathbf{e}_{1/1}\boldsymbol{\varepsilon}_1^T] = 0$. Thus, the two terms on the right-hand-side of (13.3.14) are orthogonal and we obtain the covariances:

$$E[\mathbf{e}_{1/0}\mathbf{e}_{1/0}^T] = E[\mathbf{e}_{1/1}\mathbf{e}_{1/1}^T] + G_1 E[\boldsymbol{\varepsilon}_1\boldsymbol{\varepsilon}_1^T]G_1^T\,, \quad \text{or,}$$

$$P_{1/0} = P_{1/1} + G_1 D_1 G_1^T\,, \quad \text{or,}$$

$$P_{1/1} = P_{1/0} - G_1 D_1 G_1^T = P_{1/0} - P_{1/0}C_1^T D_1^{-1} C_1 P_{1/0} \tag{13.3.15}$$

To complete the $n = 1$ steps, we must predict $\mathbf{x}_2$ from $Y_1 = \{\mathbf{y}_0, \mathbf{y}_1\} = \{\boldsymbol{\varepsilon}_0, \boldsymbol{\varepsilon}_1\}$. From the state equation $\mathbf{x}_2 = A_1\mathbf{x}_1 + \mathbf{w}_1$, we have:

$$\begin{aligned}
&\hat{\mathbf{x}}_{2/1} = \text{Proj}[\mathbf{x}_2|Y_1] = \text{Proj}[A_1\mathbf{x}_1 + \mathbf{w}_1|Y_1] = A_1\hat{\mathbf{x}}_{1/1} = A_1(\hat{\mathbf{x}}_{1/0} + G_1\boldsymbol{\varepsilon}_1) = A_1\hat{\mathbf{x}}_{1/0} + K_1\boldsymbol{\varepsilon}_1\\
&\mathbf{e}_{2/1} = \mathbf{x}_2 - \hat{\mathbf{x}}_{2/1} = A_1(\mathbf{x}_1 - \hat{\mathbf{x}}_{1/1}) + \mathbf{w}_1 = A_1\mathbf{e}_{1/1} + \mathbf{w}_1\\
&P_{2/1} = E[\mathbf{e}_{2/1}\mathbf{e}_{2/1}^T] = A_1 E[\mathbf{e}_{1/1}\mathbf{e}_{1/1}^T]A_1^T + Q_1 = A_1 P_{1/1} A_1^T + Q_1
\end{aligned}$$

where we defined $K_1 = A_1 G_1$ and used the fact that $\mathbf{w}_1$ is independent of $\mathbf{x}_1$ and $\hat{\mathbf{x}}_{1/1}$, since the latter depends only on $\bar{\mathbf{x}}_0, \mathbf{y}_0, \mathbf{y}_1$. We collect the results for $n = 1$ together:

$$\boxed{\begin{aligned}
D_1 &= C_1 P_{1/0} C_1^T + R_1\\
G_1 &= P_{1/0} C_1^T D_1^{-1}\\
K_1 &= A_1 G_1 = A_1 P_{1/0} C_1^T D_1^{-1}\\
\hat{\mathbf{y}}_{1/0} &= C_1 \hat{\mathbf{x}}_{1/0}\\
\boldsymbol{\varepsilon}_1 &= \mathbf{y}_1 - \hat{\mathbf{y}}_{1/0} = \mathbf{y}_1 - C_1\hat{\mathbf{x}}_{1/0}\\
\hat{\mathbf{x}}_{1/1} &= \hat{\mathbf{x}}_{1/0} + G_1\boldsymbol{\varepsilon}_1\\
P_{1/1} &= P_{1/0} - G_1 D_1 G_1^T\\
\hat{\mathbf{x}}_{2/1} &= A_1\hat{\mathbf{x}}_{1/1} = A_1\hat{\mathbf{x}}_{1/0} + K_1\boldsymbol{\varepsilon}_1\\
P_{2/1} &= A_1 P_{1/1} A_1^T + Q_1
\end{aligned}}$$

At the nth time instant, we assume that we have already constructed the orthogonalized basis of zero-mean innovations up to time $n-1$, i.e.,

$$Y_{n-1} = \{\mathbf{y}_0, \mathbf{y}_1, \dots, \mathbf{y}_{n-1}\} = \{\boldsymbol{\varepsilon}_0, \boldsymbol{\varepsilon}_1, \dots, \boldsymbol{\varepsilon}_{n-1}\}$$

$$E[\boldsymbol{\varepsilon}_i\boldsymbol{\varepsilon}_j^T] = D_i\delta_{ij}\,, \quad 0 \le i, j \le n-1$$

Then, the optimal prediction of $\mathbf{x}_n$ based on Y_{n-1} will be:

$$\hat{\mathbf{x}}_{n/n-1} = \text{Proj}[\mathbf{x}_n|Y_{n-1}] = \bar{\mathbf{x}}_n + \sum_{i=0}^{n-1} E[\mathbf{x}_n\boldsymbol{\varepsilon}_i^T]E[\boldsymbol{\varepsilon}_i\boldsymbol{\varepsilon}_i^T]^{-1}\boldsymbol{\varepsilon}_i \tag{13.3.16}$$

Defining $\boldsymbol{\varepsilon}_n = \mathbf{y}_n - \hat{\mathbf{y}}_{n/n-1} = \mathbf{y}_n - C_n\hat{\mathbf{x}}_{n/n-1}$, we obtain,

$$\boldsymbol{\varepsilon}_n = \mathbf{y}_n - C_n\hat{\mathbf{x}}_{n/n-1} = C_n(\mathbf{x}_n - \hat{\mathbf{x}}_{n/n-1}) + \mathbf{v}_n = C_n\mathbf{e}_{n/n-1} + \mathbf{v}_n \tag{13.3.17}$$

Therefore, $E[\boldsymbol{\varepsilon}_n] = 0$, and since $\mathbf{v}_n \perp \mathbf{e}_{n/n-1}$ (because $\mathbf{e}_{n/n-1}$ depends only on $\mathbf{x}_0, \dots, \mathbf{x}_n$ and $\mathbf{y}_0, \dots, \mathbf{y}_{n-1}$), we have:

$$D_n = E[\boldsymbol{\varepsilon}_n\boldsymbol{\varepsilon}_n^T] = C_n P_{n/n-1} C_n^T + R_n \tag{13.3.18}$$

From the optimality of $\hat{\mathbf{x}}_{n/n-1}$, we have the orthogonality property $\mathbf{e}_{n/n-1} \perp \boldsymbol{\varepsilon}_i$, or, $E[\mathbf{e}_{n/n-1}\boldsymbol{\varepsilon}_i^T] = 0$, for $i = 0, 1, \dots, n-1$, and since also $\mathbf{v}_n \perp \boldsymbol{\varepsilon}_i$, we conclude that the constructed $\boldsymbol{\varepsilon}_n$ will be orthogonal to all the previous $\boldsymbol{\varepsilon}_i$, i.e.,

$$E[\boldsymbol{\varepsilon}_n\boldsymbol{\varepsilon}_i^T] = 0\,, \quad i = 0, 1, \dots, n-1$$

Thus, we may enlarge the orthogonalized basis of the observation subspace to time n:

$$Y_n = \{\mathbf{y}_0, \mathbf{y}_1, \dots, \mathbf{y}_{n-1}, \mathbf{y}_n\} = \{\boldsymbol{\varepsilon}_0, \boldsymbol{\varepsilon}_1, \dots, \boldsymbol{\varepsilon}_{n-1}, \boldsymbol{\varepsilon}_n\}$$

$$E[\boldsymbol{\varepsilon}_i\boldsymbol{\varepsilon}_j^T] = D_i\delta_{ij}\,, \quad 0 \le i, j \le n$$

We note also that the definition $\hat{\mathbf{y}}_{n/n-1} = C_n\hat{\mathbf{x}}_{n/n-1}$ is equivalent to the conventional Gram-Schmidt construction process defined in Chap. 1, that is, starting with,

$$\hat{\mathbf{y}}_{n/n-1} = \text{Proj}[\mathbf{y}_n|Y_{n-1}] = \bar{\mathbf{y}}_n + \sum_{i=0}^{n-1} E[\mathbf{y}_n\boldsymbol{\varepsilon}_i^T]E[\boldsymbol{\varepsilon}_i\boldsymbol{\varepsilon}_i^T]^{-1}\boldsymbol{\varepsilon}_i$$

$$\boldsymbol{\varepsilon}_n = \mathbf{y}_n - \hat{\mathbf{y}}_{n/n-1}$$

then, we may justify the relationship, $\hat{\mathbf{y}}_{n/n-1} = C_n\hat{\mathbf{x}}_{n/n-1}$. Indeed, since, $\mathbf{y}_n = C_n\mathbf{x}_n + \mathbf{v}_n$, we have, $\bar{\mathbf{y}}_n = C_n\bar{\mathbf{x}}_n$, and using Eq. (13.3.16), we obtain:

$$\begin{aligned}
\hat{\mathbf{y}}_{n/n-1} &= \bar{\mathbf{y}}_n + \sum_{i=0}^{n-1} E[\mathbf{y}_n\boldsymbol{\varepsilon}_i^T]E[\boldsymbol{\varepsilon}_i\boldsymbol{\varepsilon}_i^T]^{-1}\boldsymbol{\varepsilon}_i\\
&= C_n\bar{\mathbf{x}}_n + \sum_{i=0}^{n-1} E[(C_n\mathbf{x}_n + \mathbf{v}_n)\boldsymbol{\varepsilon}_i^T]E[\boldsymbol{\varepsilon}_i\boldsymbol{\varepsilon}_i^T]^{-1}\boldsymbol{\varepsilon}_i\\
&= C_n\left[\bar{\mathbf{x}}_n + \sum_{i=0}^{n-1} E[\mathbf{x}_n\boldsymbol{\varepsilon}_i^T]E[\boldsymbol{\varepsilon}_i\boldsymbol{\varepsilon}_i^T]^{-1}\boldsymbol{\varepsilon}_i\right] = C_n\hat{\mathbf{x}}_{n/n-1}
\end{aligned}$$

Next, we consider the updated estimate of $\mathbf{x}_n$ based on Y_n and given in terms of the innovations sequence:

$$\hat{\mathbf{x}}_{n/n} = \text{Proj}[\mathbf{x}_n|Y_n] = \bar{\mathbf{x}}_n + \sum_{i=0}^{n} E[\mathbf{x}_n\boldsymbol{\varepsilon}_i^T]E[\boldsymbol{\varepsilon}_i\boldsymbol{\varepsilon}_i^T]^{-1}\boldsymbol{\varepsilon}_i \tag{13.3.19}$$

It follows that, for $n \geq 1$,

$$\hat{\mathbf{x}}_{n/n} = \hat{\mathbf{x}}_{n/n-1} + E[\mathbf{x}_n\boldsymbol{\varepsilon}_n^T]E[\boldsymbol{\varepsilon}_n\boldsymbol{\varepsilon}_n^T]^{-1}\boldsymbol{\varepsilon}_n = \hat{\mathbf{x}}_{n/n-1} + G_n\boldsymbol{\varepsilon}_n \tag{13.3.20}$$

Because $\boldsymbol{\varepsilon}_n \perp \boldsymbol{\varepsilon}_i$, $i = 0, 1, \dots, n-1$, we have $E[\hat{\mathbf{x}}_{n/n-1}\boldsymbol{\varepsilon}_n^T] = 0$, which implies,

$$\begin{aligned} E[\mathbf{x}_n\boldsymbol{\varepsilon}_n^T] &= E[(\mathbf{x}_n - \hat{\mathbf{x}}_{n/n-1})\boldsymbol{\varepsilon}_n^T] = E[\mathbf{e}_{n/n-1}\boldsymbol{\varepsilon}_n^T] = E[\mathbf{e}_{n/n-1}(C_n\mathbf{e}_{n/n-1} + \mathbf{v}_n)^T] \\ &= E[\mathbf{e}_{n/n-1}\mathbf{e}_{n/n-1}^T]C_n^T = P_{n/n-1}C_n^T \end{aligned}$$

Thus, the Kalman gain will be:

$$G_n = E[\mathbf{x}_n\boldsymbol{\varepsilon}_n^T]E[\boldsymbol{\varepsilon}_n\boldsymbol{\varepsilon}_n^T]^{-1} = P_{n/n-1}C_n^T D_n^{-1} = P_{n/n-1}C_n^T[C_nP_{n/n-1}C_n^T + R_n]^{-1} \tag{13.3.21}$$

The estimation errors $\mathbf{e}_{n/n}$ and $\mathbf{e}_{n/n-1}$ are related by,

$$\mathbf{e}_{n/n} = \mathbf{x}_n - \hat{\mathbf{x}}_{n/n} = \mathbf{x}_n - \hat{\mathbf{x}}_{n/n-1} - G_n\boldsymbol{\varepsilon}_n = \mathbf{e}_{n/n-1} - G_n\boldsymbol{\varepsilon}_n\,, \quad \text{or,}$$

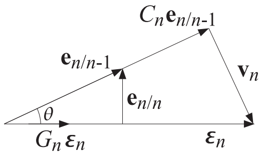
$$\mathbf{e}_{n/n-1} = \mathbf{e}_{n/n} + G_n\boldsymbol{\varepsilon}_n \tag{13.3.22}$$

and since one of the orthogonality conditions for $\hat{\mathbf{x}}_{n/n}$ is $E[\mathbf{e}_{n/n}\boldsymbol{\varepsilon}_n^T] = 0$, the two terms in the right-hand side will be orthogonal, which leads to the covariance relation:

$$P_{n/n-1} = P_{n/n} + G_nD_nG_n^T\,, \quad \text{or,}$$

$$P_{n/n} = P_{n/n-1} - G_nD_nG_n^T = P_{n/n-1} - P_{n/n-1}C_n^TD_n^{-1}C_nP_{n/n-1} \tag{13.3.23}$$

A nice geometrical interpretation of Eqs. (13.3.17) and (13.3.22) was given by Kronhamn [867] and is depicted below (see also Chap. 11):



The similarity of the two orthogonal triangles leads to Eq. (13.3.21). Indeed, for the scalar case, the lengths of the triangle sides are given by the square roots of the covariances, e.g., $\sqrt{E[\varepsilon_n^2]} = \sqrt{D_n}$. Then, the Pythagorean theorem and the similarity of the triangles give,

$$\begin{aligned} &E[\varepsilon_n^2] = C_n^2E[e_{n/n-1}^2] + E[v_n^2] \quad\Rightarrow\quad D_n = C_nP_{n/n-1}C_n + R_n \\ &E[e_{n/n-1}^2] = E[e_{n/n}^2] + G_n^2E[\varepsilon_n^2] \quad\Rightarrow\quad P_{n/n-1} = P_{n/n} + G_nD_nG_n \\ &\cos\theta = \frac{G_n\sqrt{D_n}}{\sqrt{P_{n/n-1}}} = \frac{C_n\sqrt{P_{n/n-1}}}{\sqrt{D_n}} \quad\Rightarrow\quad G_n = \frac{P_{n/n-1}C_n}{D_n} \\ &\sin\theta = \frac{\sqrt{P_{n/n}}}{\sqrt{P_{n/n-1}}} = \frac{\sqrt{R_n}}{\sqrt{D_n}} \quad\Rightarrow\quad P_{n/n-1}D_n^{-1} = P_{n/n}R_n^{-1} \quad\Rightarrow\quad \text{Eq. (13.2.5)} \end{aligned}$$

Finally, we determine the next predicted estimate, which may be obtained by using the state equation $\mathbf{x}_{n+1} = A_n\mathbf{x}_n + \mathbf{w}_n$, and noting that $E[\mathbf{x}_{n+1}\boldsymbol{\varepsilon}_i^T] = E[(A_n\mathbf{x}_n + \mathbf{w}_n)\boldsymbol{\varepsilon}_i^T] = A_nE[\mathbf{x}_n\boldsymbol{\varepsilon}_i^T]$, for $0 \leq i \leq n$. Then, using $\bar{\mathbf{x}}_{n+1} = A_n\bar{\mathbf{x}}_n$, we find,

$$\begin{aligned} \hat{\mathbf{x}}_{n+1/n} &= \bar{\mathbf{x}}_{n+1} + \sum_{i=0}^{n} E[\mathbf{x}_{n+1}\boldsymbol{\varepsilon}_i^T]E[\boldsymbol{\varepsilon}_i\boldsymbol{\varepsilon}_i^T]^{-1}\boldsymbol{\varepsilon}_i \\ &= A_n\left[\bar{\mathbf{x}}_n + \sum_{i=0}^{n} E[\mathbf{x}_n\boldsymbol{\varepsilon}_i^T]E[\boldsymbol{\varepsilon}_i\boldsymbol{\varepsilon}_i^T]^{-1}\boldsymbol{\varepsilon}_i\right] = A_n\hat{\mathbf{x}}_{n/n} = A_n[\hat{\mathbf{x}}_{n/n-1} + G_n\boldsymbol{\varepsilon}_n] \\ &= A_n\hat{\mathbf{x}}_{n/n-1} + K_n\boldsymbol{\varepsilon}_n = (A_n - K_nC_n)\hat{\mathbf{x}}_{n/n-1} + K_n\mathbf{y}_n \end{aligned}$$

where we defined $K_n = A_nG_n$. The error covariance is obtained by noting that,

$$\mathbf{e}_{n+1/n} = \mathbf{x}_{n+1} - \hat{\mathbf{x}}_{n+1/n} = A_n\mathbf{e}_{n/n} + \mathbf{w}_n$$

and because $\mathbf{w}_n$ is orthogonal to $\mathbf{e}_{n/n}$, this leads to

$$P_{n+1/n} = E[\mathbf{e}_{n+1/n}\mathbf{e}_{n+1/n}^T] = A_nE[\mathbf{e}_{n/n}\mathbf{e}_{n/n}^T]A_n^T + E[\mathbf{w}_n\mathbf{w}_n^T] = A_nP_{n/n}A_n^T + Q_n$$

This completes the operations at the nth time step. The various equivalent expressions in Eqs. (13.2.4) and (13.2.5) are straightforward to derive. The Joseph form is useful because it guarantees the numerical positive-definiteness of the error covariance matrix. The information form is a consequence of the matrix inversion lemma. It can be shown directly as follows. To simplify the notation, we write the covariance update as,

$$\hat{P} = P - GDG^T = P - PC^TD^{-1}CP\,, \quad D = R + CPC^T$$

Multiply from the right by P^{-1} and from the left by $\hat{P}^{-1}$ to get:

$$P^{-1} = \hat{P}^{-1} - \hat{P}^{-1}PC^TD^{-1}C \tag{13.3.24}$$

Next, multiply from the right by PC^T to get:

$$\begin{aligned} C^T &= \hat{P}^{-1}PC^T - \hat{P}^{-1}PC^TD^{-1}CPC^T = \hat{P}^{-1}PC^T(I - D^{-1}CPC^T) \\ &= \hat{P}^{-1}PC^TD^{-1}(D - CPC^T) = \hat{P}^{-1}PC^TD^{-1}R \end{aligned}$$

which gives (assuming that R^{-1} exists):

$$C^TR^{-1} = \hat{P}^{-1}PC^TD^{-1}$$

Inserting this result into Eq. (13.3.24), we obtain

$$P^{-1} = \hat{P}^{-1} - \hat{P}^{-1}PC^TD^{-1}C = \hat{P}^{-1} - C^TR^{-1}C \quad\Rightarrow\quad \hat{P}^{-1} = P^{-1} + C^TR^{-1}C$$

and also obtain,

$$\hat{P}C^TR^{-1} = PC^TD^{-1} = G$$

Since the information form works with the inverse covariances, to complete the operations at each time step, we need to develop a recursion for the inverse $P_{n+1/n}^{-1}$ in terms of $P_{n/n}^{-1}$. Denoting $P_{n+1/n}$ by P_{next}, we have

$$P_{\text{next}} = A\hat{P}A^T + Q$$

If we assume that A^{-1} and Q^{-1} exist, then the application of the matrix inversion lemma to this equation allows us to rewrite it in terms of the matrix inverses:

$$P_{\text{next}}^{-1} = A^{-T}\hat{P}^{-1}A^{-1} - A^{-T}\hat{P}^{-1}A^{-1}\big[Q^{-1} + A^{-T}\hat{P}^{-1}A^{-1}\big]^{-1}A^{-T}\hat{P}^{-1}A^{-1}$$

To summarize, the information form of the Kalman filter is as follows:

$$\begin{aligned}
&P_{n/n}^{-1} = P_{n/n-1}^{-1} + C_n^T R_n^{-1} C_n \\
&P_{n/n}^{-1}\hat{\mathbf{x}}_{n/n} = P_{n/n-1}^{-1}\hat{\mathbf{x}}_{n/n-1} + C_n^T R_n^{-1}\mathbf{y}_n \\
&P_{n+1/n}^{-1} = A_n^{-T}P_{n/n}^{-1}A_n^{-1} - A_n^{-T}P_{n/n}^{-1}A_n^{-1}\big[Q_n^{-1} + A_n^{-T}P_{n/n}^{-1}A_n^{-1}\big]^{-1}A_n^{-T}P_{n/n}^{-1}A_n^{-1}
\end{aligned} \tag{13.3.25}$$

13.4 Forecasting and Missing Observations

The problem of forecasting ahead from the current time sample n and the problem of missing observations are similar. Suppose one has at hand the estimate $\hat{\mathbf{x}}_{n/n}$ based on $Y_n = \{\mathbf{y}_0, \mathbf{y}_1, \dots, \mathbf{y}_n\}$. Then the last part of the Kalman filtering algorithm (13.2.2) produces the prediction of $\mathbf{x}_{n+1}$ based on Y_n,

$$\hat{\mathbf{x}}_{n+1/n} = A_n\hat{\mathbf{x}}_{n/n}$$

This prediction can be continued to future times. For example, since $\mathbf{x}_{n+2} = A_{n+1}\mathbf{x}_{n+1} + \mathbf{w}_{n+1}$ and $\mathbf{w}_{n+1}$ is independent of Y_n, we have:

$$\begin{aligned}
\hat{\mathbf{x}}_{n+2/n} &= \text{Proj}\big[\mathbf{x}_{n+2}|Y_n\big] \\
&= \text{Proj}\big[A_{n+1}\mathbf{x}_{n+1} + \mathbf{w}_{n+1}|Y_n\big] \\
&= A_{n+1}\hat{\mathbf{x}}_{n+1/n} = A_{n+1}A_n\hat{\mathbf{x}}_{n/n} = \Phi_{n+2,n}\hat{\mathbf{x}}_{n/n}
\end{aligned}$$

and so on. Thus, the prediction of $\mathbf{x}_{n+p}$ based on Y_n is for $p \ge 1$,

$$\hat{\mathbf{x}}_{n+p/n} = \Phi_{n+p,n}\hat{\mathbf{x}}_{n/n} \tag{13.4.1}$$

The corresponding error covariance is found by applying (13.1.8), that is,

$$\mathbf{x}_{n+p} = \Phi_{n+p,n}\,\mathbf{x}_n + \sum_{k=n+1}^{n+p}\Phi_{n+p,k}\mathbf{w}_{k-1}\,, \quad p \ge 1 \tag{13.4.2}$$

which in conjunction with (13.4.1), gives for the forecast error $\mathbf{e}_{n+p/n} = \mathbf{x}_n - \hat{\mathbf{x}}_{n+p/n}$,

$$\mathbf{e}_{n+p/n} = \Phi_{n+p,n}\,\mathbf{e}_{n/n} + \sum_{k=n+1}^{n+p}\Phi_{n+p,k}\mathbf{w}_{k-1} \tag{13.4.3}$$

which implies for its covariance:

$$P_{n+p/n} = \Phi_{n+p,n}P_{n/n}\Phi_{n+p,n}^T + \sum_{k=n+1}^{n+p}\Phi_{n+p,k}Q_{k-1}\Phi_{n+p,k}^T \tag{13.4.4}$$

Eqs. (13.4.1) and (13.4.4) apply also in the case when a group of observations, say, $\{\mathbf{y}_{n+1}, \mathbf{y}_{n+2}, \dots, \mathbf{y}_{n+p-1}\}$, are missing. In such case, one simply predicts ahead from time n using the observation set Y_n. Once the observation $\mathbf{y}_{n+p}$ becomes available, one may resume the usual algorithm using the initial values $\hat{\mathbf{x}}_{n+p/n}$ and $P_{n+p/n}$.

This procedure is equivalent to setting, in the algorithm (13.2.2), $G_{n+i} = 0$ and $\boldsymbol{\varepsilon}_{n+i} = 0$ over the period of the missing observations, $i = 1, 2, \dots, p-1$, that is, ignoring the measurement updates, but not the time updates.

In some presentations of the Kalman filtering algorithm, it is assumed that the observations are available from $n \ge 1$, i.e., $Y_n = \{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n\} = \{\boldsymbol{\varepsilon}_1, \boldsymbol{\varepsilon}_2, \dots, \boldsymbol{\varepsilon}_n\}$, and the algorithm is initialized at $\hat{\mathbf{x}}_{0/0} = \bar{\mathbf{x}}_0$ with $P_{0/0} = \Sigma_0$. We may view this as a case of a missing observation $\mathbf{y}_0$, and therefore, from the above rule, we may set $\boldsymbol{\varepsilon}_0 = 0$ and $G_0 = 0$, which leads to $\hat{\mathbf{x}}_{0/0} = \hat{\mathbf{x}}_{0/-1} = \bar{\mathbf{x}}_0$ and $P_{0/0} = P_{0/-1} = \Sigma_0$. The algorithm may be stated then somewhat differently, but equivalently, to Eq. (13.2.2):

Initialize at $n = 0$ by: $\hat{\mathbf{x}}_{0/0} = \bar{\mathbf{x}}_0$, $P_{0/0} = \Sigma_0$

At time $n \ge 1$, $\hat{\mathbf{x}}_{n-1/n-1}$, $P_{n-1/n-1}$, $\mathbf{y}_n$ are available,

$$\begin{array}{ll}
\hat{\mathbf{x}}_{n/n-1} = A_{n-1}\hat{\mathbf{x}}_{n-1/n-1} & \text{predicted estimate} \\
P_{n/n-1} = A_{n-1}P_{n-1/n-1}A_{n-1}^T + Q_{n-1} & \text{prediction error} \\
D_n = C_nP_{n/n-1}C_n^T + R_n & \text{innovations covariance} \\
G_n = P_{n/n-1}C_n^TD_n^{-1} & \text{Kalman gain for filtering} \\
\hat{\mathbf{y}}_{n/n-1} = C_n\hat{\mathbf{x}}_{n/n-1} & \text{predicted measurement} \\
\boldsymbol{\varepsilon}_n = \mathbf{y}_n - \hat{\mathbf{y}}_{n/n-1} = \mathbf{y}_n - C_n\hat{\mathbf{x}}_{n/n-1} & \text{innovations sequence} \\
\hat{\mathbf{x}}_{n/n} = \hat{\mathbf{x}}_{n/n-1} + G_n\boldsymbol{\varepsilon}_n & \text{filtered estimate} \\
P_{n/n} = P_{n/n-1} - G_nD_nG_n^T & \text{mean-square error}
\end{array}$$

Go to time $n + 1$

13.5 Kalman Filter with Deterministic Inputs

A state/measurement model that has a deterministic input $\mathbf{u}_n$ in addition the noise input $\mathbf{w}_n$ can be formulated by,

$$\boxed{\begin{aligned}\mathbf{x}_{n+1} &= A_n\mathbf{x}_n + B_n\mathbf{u}_n + \mathbf{w}_n \\ \mathbf{y}_n &= C_n\mathbf{x}_n + \mathbf{v}_n\end{aligned}} \quad \begin{array}{l}\text{(state model)} \\ \text{(measurement model)}\end{array} \tag{13.5.1}$$

As we mentioned earlier, this requires a minor modification of the algorithm (13.2.2), namely, replacing the time-update equation by that in Eq. (13.5.3) below. Using linear superposition, we may think of this model as two models, one driven by the white noise inputs, and the other by the deterministic input, that is,

$$\begin{aligned} \mathbf{x}_{n+1}^{(1)} &= A_n\mathbf{x}_n^{(1)} + \mathbf{w}_n \\ \mathbf{y}_n^{(1)} &= C_n\mathbf{x}_n^{(1)} + \mathbf{v}_n \end{aligned} \quad \text{and} \quad \begin{aligned} \mathbf{x}_{n+1}^{(2)} &= A_n\mathbf{x}_n^{(2)} + B_n\mathbf{u}_n \\ \mathbf{y}_n^{(2)} &= C_n\mathbf{x}_n^{(2)} \end{aligned} \tag{13.5.2}$$

If we adjust the initial conditions of the two systems to match that of (13.5.1), that is, $\mathbf{x}_0 = \mathbf{x}_0^{(1)} + \mathbf{x}_0^{(2)}$, then the solution of the system (13.5.1) will be the sum:

$$\mathbf{x}_n = \mathbf{x}_n^{(1)} + \mathbf{x}_n^{(2)}\,, \quad \mathbf{y}_n = \mathbf{y}_n^{(1)} + \mathbf{y}_n^{(2)}$$

System (2) is purely deterministic, and therefore, we have the estimates,

$$\begin{aligned} \hat{\mathbf{x}}_{n/n}^{(2)} &= \text{Proj}[\mathbf{x}_n^{(2)}|Y_n] = \mathbf{x}_n^{(2)} \\ \hat{\mathbf{x}}_{n/n-1}^{(2)} &= \text{Proj}[\mathbf{x}_n^{(2)}|Y_{n-1}] = \mathbf{x}_n^{(2)} \end{aligned}$$

and similarly, $\hat{\mathbf{y}}_{n/n-1}^{(2)} = \mathbf{y}_n^{(2)}$. For system (1), we may apply the Kalman filtering algorithm of Eq. (13.2.2). We note that

$$\boldsymbol{\varepsilon}_n = \mathbf{y}_n - \hat{\mathbf{y}}_{n/n-1} = \mathbf{y}_n^{(1)} + \mathbf{y}_n^{(2)} - \hat{\mathbf{y}}_{n/n-1}^{(1)} - \hat{\mathbf{y}}_{n/n-1}^{(2)} = \mathbf{y}_n^{(1)} - \hat{\mathbf{y}}_{n/n-1}^{(1)}$$

so that $D_n = D_n^{(1)}$. Similarly, we find,

$$\begin{aligned} \mathbf{e}_{n/n-1} &= \mathbf{x}_n - \hat{\mathbf{x}}_{n/n-1} = \mathbf{x}_n^{(1)} - \hat{\mathbf{x}}_{n/n-1}^{(1)} \quad &\Rightarrow \quad P_{n/n-1} = P_{n/n-1}^{(1)} \\ \mathbf{e}_{n/n} &= \mathbf{x}_n - \hat{\mathbf{x}}_{n/n} = \mathbf{x}_n^{(1)} - \hat{\mathbf{x}}_{n/n}^{(1)} \quad &\Rightarrow \quad P_{n/n} = P_{n/n}^{(1)} \end{aligned}$$

and similarly, $G_n = G_n^{(1)}$ and $K_n = K_n^{(1)}$. The measurement update equation remains the same, that is,

$$\hat{\mathbf{x}}_{n/n} = \hat{\mathbf{x}}_{n/n}^{(1)} + \mathbf{x}_n^{(2)} = \hat{\mathbf{x}}_{n/n-1}^{(1)} + G_n\boldsymbol{\varepsilon}_n + \mathbf{x}_n^{(2)} = \hat{\mathbf{x}}_{n/n-1} + G_n\boldsymbol{\varepsilon}_n$$

The only step of the algorithm (13.2.2) that changes is the time update equation:

$$\begin{aligned} \hat{\mathbf{x}}_{n+1/n} &= \hat{\mathbf{x}}_{n+1/n}^{(1)} + \hat{\mathbf{x}}_{n+1/n}^{(2)} = \hat{\mathbf{x}}_{n+1/n}^{(1)} + \mathbf{x}_{n+1}^{(2)} \\ &= \big[A_n\hat{\mathbf{x}}_{n/n-1}^{(1)} + K_n\boldsymbol{\varepsilon}_n\big] + \big[A_n\mathbf{x}_n^{(2)} + B_n\mathbf{u}_n\big] \\ &= A_n\big(\hat{\mathbf{x}}_{n/n-1}^{(1)} + \mathbf{x}_n^{(2)}\big) + B_n\mathbf{u}_n + K_n\boldsymbol{\varepsilon}_n \\ &= A_n\hat{\mathbf{x}}_{n/n-1} + B_n\mathbf{u}_n + K_n\boldsymbol{\varepsilon}_n = A_n\hat{\mathbf{x}}_{n/n} + B_n\mathbf{u}_n\,, \quad \text{or,} \end{aligned}$$

$$\boxed{\hat{\mathbf{x}}_{n+1/n} = A_n\hat{\mathbf{x}}_{n/n-1} + B_n\mathbf{u}_n + K_n\boldsymbol{\varepsilon}_n = A_n\hat{\mathbf{x}}_{n/n} + B_n\mathbf{u}_n} \tag{13.5.3}$$

13.6 Time-Invariant Models

In many applications, the model parameters $\{A_n, C_n, Q_n, R_n\}$ are constants in time, that is, $\{A, C, Q, R\}$, and the model takes the form:

$$\boxed{\begin{aligned} \mathbf{x}_{n+1} &= A\mathbf{x}_n + \mathbf{w}_n \\ \mathbf{y}_n &= C\mathbf{x}_n + \mathbf{v}_n \end{aligned}} \quad \begin{aligned} &\text{(state model)} \\ &\text{(measurement model)} \end{aligned} \tag{13.6.1}$$

The signals $\mathbf{w}_n, \mathbf{v}_n$ are again assumed to be mutually-independent, zero-mean, white-noise signals with known covariance matrices:

$$\boxed{\begin{aligned} E[\mathbf{w}_n\mathbf{w}_i^T] &= Q\delta_{ni} \\ E[\mathbf{v}_n\mathbf{v}_i^T] &= R\delta_{ni} \\ E[\mathbf{w}_n\mathbf{v}_i^T] &= 0 \end{aligned}} \tag{13.6.2}$$

The model is iterated starting at $n = 0$. The initial state vector $\mathbf{x}_0$ is assumed to be random and independent of $\mathbf{w}_n, \mathbf{v}_n$, but with a known mean $\bar{\mathbf{x}}_0 = E[\mathbf{x}_0]$ and covariance matrix $\Sigma_0 = E[(\mathbf{x}_0 - \bar{\mathbf{x}}_0)(\mathbf{x}_0 - \bar{\mathbf{x}}_0)^T]$. The Kalman filtering algorithm (13.2.2) then takes the form:

$$\boxed{\begin{array}{ll} \text{Initialize in time by: } \hat{\mathbf{x}}_{0/-1} = \bar{\mathbf{x}}_0,\ P_{0/-1} = \Sigma_0 & \\ \text{At time } n,\ \hat{\mathbf{x}}_{n/n-1},\ P_{n/n-1},\ \mathbf{y}_n \text{ are available,} & \\ \quad D_n = CP_{n/n-1}C^T + R & \text{innovations covariance} \\ \quad G_n = P_{n/n-1}C^TD_n^{-1} & \text{Kalman gain for filtering} \\ \quad K_n = AG_n = AP_{n/n-1}C^TD_n^{-1} & \text{Kalman gain for prediction} \\ \quad \hat{\mathbf{y}}_{n/n-1} = C\hat{\mathbf{x}}_{n/n-1} & \text{predicted measurement} \\ \quad \boldsymbol{\varepsilon}_n = \mathbf{y}_n - \hat{\mathbf{y}}_{n/n-1} = \mathbf{y}_n - C\hat{\mathbf{x}}_{n/n-1} & \text{innovations sequence} \\ \text{Measurement update / correction:} & \\ \quad \hat{\mathbf{x}}_{n/n} = \hat{\mathbf{x}}_{n/n-1} + G_n\boldsymbol{\varepsilon}_n & \text{filtered estimate} \\ \quad P_{n/n} = P_{n/n-1} - G_nD_nG_n^T & \text{estimaton error} \\ \text{Time update / prediction:} & \\ \quad \hat{\mathbf{x}}_{n+1/n} = A\hat{\mathbf{x}}_{n/n} = A\hat{\mathbf{x}}_{n/n-1} + K_n\boldsymbol{\varepsilon}_n & \text{predicted estimate} \\ \quad P_{n+1/n} = AP_{n/n}A^T + Q & \text{prediction error} \\ \text{Go to time } n+1 & \end{array}} \tag{13.6.3}$$

Note also that Eqs. (13.2.12) become now,

$$\boxed{\begin{array}{ll} \hat{\mathbf{y}}_{n/n-1} = C\hat{\mathbf{x}}_{n/n-1} = CA\hat{\mathbf{x}}_{n-1/n-1} & \text{predicted measurement} \\ \boldsymbol{\varepsilon}_n = \mathbf{y}_n - \hat{\mathbf{y}}_{n/n-1} & \text{innovations sequence} \\ \hat{\mathbf{x}}_{n+1/n} = A\hat{\mathbf{x}}_{n/n-1} + K\boldsymbol{\varepsilon}_n & \text{predicted estimate} \\ \hat{\mathbf{x}}_{n/n} = A\hat{\mathbf{x}}_{n-1/n-1} + G\boldsymbol{\varepsilon}_n & \text{filtered estimate} \end{array}} \tag{13.6.4}$$

The MATLAB function, `kfilt.m`, implements the Kalman filtering algorithm of Eq. (13.6.3). It has usage:

```
[L,X,P,Xf,Pf] = kfilt(A,C,Q,R,Y,x0,S0);     % Kalman filtering
```

Its inputs are the state-space model parameters $\{A, C, Q, R\}$, the initial values $\bar{\mathbf{x}}_0$, Σ_0, and the observations $\mathbf{y}_n$, $0 \le n \le N$, arranged into an $r\times(N+1)$ matrix:

$$Y = \big[\mathbf{y}_0, \mathbf{y}_1, \dots, \mathbf{y}_n, \dots, \mathbf{y}_N\big]$$

The outputs are the predicted and filtered estimates arranged into $p\times(N+1)$ matrices:

$$\begin{aligned} X &= \big[\hat{\mathbf{x}}_{0/-1},\, \hat{\mathbf{x}}_{1/0},\, \dots,\, \hat{\mathbf{x}}_{n/n-1},\, \dots,\, \hat{\mathbf{x}}_{N/N-1}\big] \\ X_f &= \big[\hat{\mathbf{x}}_{0/0},\, \hat{\mathbf{x}}_{1/1},\, \dots,\, \hat{\mathbf{x}}_{n/n},\, \dots,\, \hat{\mathbf{x}}_{N/N}\big] \end{aligned}$$

whose error covariance matrices are arranged into the $p\times p\times(N+1)$ three-dimensional arrays P, P_f, such that (in MATLAB notation):

$$\texttt{P(:,:,n+1)} = P_{n/n-1}\,, \quad \texttt{Pf(:,:,n+1)} = P_{n/n}\,, \quad 0 \le n \le N$$

The output L is the value of the negative-log-likelihood function calculated from Eq. (13.12.2).

Under certain conditions of stabilizability and detectability, the Kalman filter parameters $\{D_n, P_{n/n-1}, G_n, K_n\}$ converge to steady-state values $\{D, P, G, K\}$ such that P is unique and positive-semidefinite symmetric and the converged closed-loop state matrix $F = A - KC$ is stable, i.e., its eigenvalues are strictly inside the unit circle. The steady-state values are all given in terms of P, as follows:

$$\boxed{\begin{aligned} D &= CPC^T + R \\ G &= PC^TD^{-1} = [I + PC^TR^{-1}C]^{-1}PC^TR^{-1} \\ K &= AG = A[I + PC^TR^{-1}C]^{-1}PC^TR^{-1} \\ F &= A - KC = A[I + PC^TR^{-1}C]^{-1} \end{aligned}} \qquad (13.6.5)$$

and P is determined as the unique positive-semidefinite symmetric solution of the so-called *discrete algebraic Riccati equation* (DARE), written in two alternative ways:

$$\boxed{\begin{aligned} P &= APA^T - APC^T(CPC^T + R)^{-1}CPA^T + Q \\ P &= A[I + PC^TR^{-1}C]^{-1}PA^T + Q \end{aligned}} \quad \text{(DARE)} \qquad (13.6.6)$$

The required conditions are that the pair $[C, A]$ be completely detectable and the pair $[A, Q^{1/2}]$, completely stabilizable,[†] where $Q^{1/2}$ denotes a square root of the positive semidefinite matrix Q. Refs. [863,865] include a literature overview of various conditions for this and related weaker results. The convergence speed of the time-varying quantities to their steady-state values is determined essentially by the magnitude *square* of largest eigenvalue of the closed-loop matrix $F = A - KC$ (see for example [871,872]). If we define the eigenvalue radius $\rho = \max_i |\lambda_i|$, where λ_i are the eigenvalues of F, then a measure of the effective time constant is:

$$n_{\text{eff}} = \frac{\ln\epsilon}{\ln\rho^2} \qquad (13.6.7)$$

[†] For definitions of complete stabilizability and detectability see [863], which is available online.

where ϵ is a small user-defined quantity, such as $\epsilon = 10^{-2}$ for the 40-dB time constant, or $\epsilon = 10^{-3}$ for the 60-dB time constant.

The MATLAB function, `dare()`, in the control systems toolbox allows the calculation of the solution P and Kalman gain K, with usage:

```
[P,L,KT] = dare(A', C', Q, R);
```

where the output `P` is the required solution, `KT` is the transposed of the gain K, and `L` is the vector of eigenvalues of the closed-loop matrix $F = A - KC$. The syntax requires that the input matrices A, C be entered in transposed form.

Example 13.6.1: *Benchmark Example.* Solution methods of the DARE are reviewed in [877]. The following two-dimensional model is a benchmark example from the collection [878]:

$$A = \begin{bmatrix} 4 & -4.5 \\ 3 & -3.5 \end{bmatrix}, \quad C = [1,\, -1]\,, \quad Q = \begin{bmatrix} 9 & 6 \\ 6 & 4 \end{bmatrix} = \begin{bmatrix} 3 \\ 2 \end{bmatrix}[3,\, 2]\,, \quad R = 1$$

The MATLAB call,

```
[P,L,K_tr] = dare(A', C', Q, R);
```

returns the values:

$$P = \begin{bmatrix} 14.5623 & 9.7082 \\ 9.7082 & 6.4721 \end{bmatrix}, \quad K = K_{\text{tr}}^T = \begin{bmatrix} 1.8541 \\ 1.2361 \end{bmatrix}, \quad L = \begin{bmatrix} 0.3820 \\ -0.5000 \end{bmatrix}$$

These agree with the exact solutions:

$$P = \frac{1+\sqrt{5}}{2}\begin{bmatrix} 9 & 6 \\ 6 & 4 \end{bmatrix}, \quad K = \frac{\sqrt{5}-1}{2}\begin{bmatrix} 3 \\ 2 \end{bmatrix}, \quad L = \begin{bmatrix} (3-\sqrt{5})/2 \\ -0.5 \end{bmatrix}$$

This example does satisfy the stabilizability and detectability requirements for convergence, even though the model itself is uncontrollable and unobservable. Indeed, using the square root factor $\mathbf{q} = [3, 2]^T$ for Q where $Q = \mathbf{q}\mathbf{q}^T$, we see that the controllability and observability matrices are rank defective:

$$[\mathbf{q},\, A\mathbf{q}] = \begin{bmatrix} 3 & 3 \\ 2 & 2 \end{bmatrix}, \quad \begin{bmatrix} C \\ CA \end{bmatrix} = \begin{bmatrix} 1 & -1 \\ 1 & -1 \end{bmatrix}$$

The largest eigenvalue of the matrix F is $\lambda_1 = -0.5$, which leads to an estimated 40-dB time constant of $n_{\text{eff}} = \log(0.01)/\log\big((0.5)^2\big) = 3.3$. The time-varying prediction-error matrix $P_{n/n-1}$ can be given in closed form. Using the methods of [871,872], we find:

$$P_{n/n-1} = P + F^nM_nF^{Tn}\,, \quad n \ge 0 \qquad (13.6.8)$$

where, F^n is the nth power of F and can be expressed in terms of its eigenvalues, as follows,

$$F^n = \begin{bmatrix} 3\lambda_2^n - 2\lambda_1^n & 3\lambda_1^n - 3\lambda_2^n \\ 2\lambda_2^n - 2\lambda_1^n & 3\lambda_1^n - 2\lambda_2^n \end{bmatrix}, \quad \lambda_1 = -0.5\,, \quad \lambda_2 = \frac{3-\sqrt{5}}{2}$$

This is obtained from the eigenvalue decomposition:

$$F = V\Lambda V^{-1} = \begin{bmatrix} 1 & 1.5 \\ 1 & 1 \end{bmatrix}\begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}\begin{bmatrix} -2 & 3 \\ 2 & -2 \end{bmatrix} \quad\Rightarrow\quad F^n = V\Lambda^n V^{-1}$$

The matrix M_n is given by:

$$M_n = \frac{1}{ad - bc + (a+b+c+d)c_n}\begin{bmatrix} c_n + d & c_n - b \\ c_n - c & c_n + a \end{bmatrix}, \quad c_n = \frac{1}{\sqrt{5}}(1 - \lambda_2^{2n})$$

and the numbers a, b, c, d are related to an arbitrary initial value $P_{0/-1}$ via the definition:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} = (P_{0/-1} - P)^{-1} \quad\Rightarrow\quad P_{0/-1} - P = \frac{1}{ad - bc}\begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

provided that the indicated matrix inverse exists. We note that at $n = 0$, $M_0 = P_{0/-1} - P$ and the above solution for $P_{n/n-1}$ correctly accounts for the initial condition. It can be verified that Eq. (13.6.8) is the solution to the difference equation with the prescribed initial value:

$$P_{n+1/n} = A\big[P_{n/n-1} - P_{n/n-1}C^T(CP_{n/n-1}C^T + R)^{-1}CP_{n/n-1}\big]A^T + Q\,, \quad \text{or,}$$

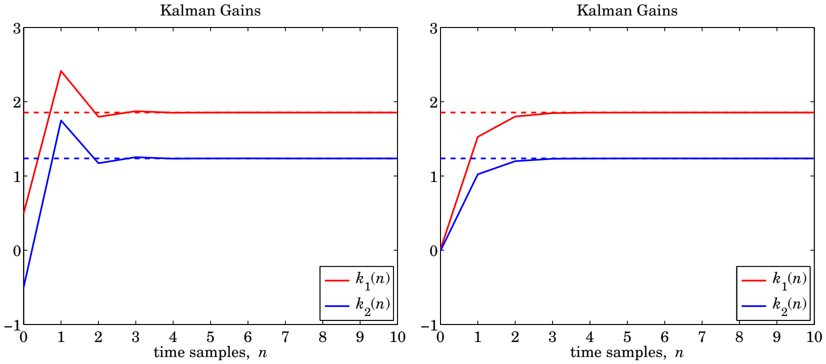
$$P_{n+1/n} = A\big[I + P_{n/n-1}C^T R^{-1}C\big]^{-1}P_{n/n-1}A^T + Q$$

Since $|\lambda_2| < |\lambda_1|$, it is evident from the above solution that the convergence time-constant is determined by $|\lambda_1|^2$. As $P_{n/n-1} \to P$, so does the Kalman gain:

$$K_n = AP_{n/n-1}C^T(CP_{n/n-1}C^T + R)^{-1} \longrightarrow K = APC^T(CPC^T + R)^{-1}$$

The figure below plots the two components of the gain vector $K_n = [k_1(n), k_2(n)]^T$ versus time n, for the two choices of initial conditions:

$$P_{0/-1} = \begin{bmatrix} 100 & 0 \\ 0 & 100 \end{bmatrix}, \quad P_{0/-1} = \begin{bmatrix} 1/100 & 0 \\ 0 & 1/100 \end{bmatrix}$$



We note that we did not initialize to $P_{0/-1} = 0$ because P is rank defective and the initial matrix $M_0 = -P$ would not be invertible. □

Example 13.6.2: *Local Level Model.* Consider the local-level model of Example 13.1.1 (see also Problem 11.13),

$$x_{n+1} = x_n + w_n$$

$$y_n = x_n + v_n$$

with $Q = \sigma_w^2$ and $R = \sigma_v^2$. The Kalman filtering algorithm (13.6.3) has the parameters $A = 1$, $C = 1$, and satisfies the Riccati difference and algebraic equations:

$$P_{n+1/n} = \frac{P_{n/n-1}R}{P_{n/n-1} + R} + Q \quad\Rightarrow\quad P = \frac{PR}{P+R} + Q \quad\Rightarrow\quad \frac{P^2}{P+R} = Q$$

and has time-varying and steady gains:

$$K_n = G_n = \frac{P_{n/n-1}}{P_{n/n-1} + R} \quad\Rightarrow\quad K = G = \frac{P}{P+R}$$

and closed-loop transition matrix:

$$F_n = 1 - K_n = \frac{R}{P_{n/n-1} + R} \quad\Rightarrow\quad F = 1 - K = \frac{R}{P+R}$$

The positive solution of the algebraic Riccati equation is:

$$P = \frac{Q}{2} + \sqrt{QR + \frac{Q^2}{4}}$$

The convergence properties to the steady values depend on the closed-loop matrix (here scalar) F. Again, using the methods of [871,872], we find the exact solutions:

$$P_{n/n-1} = P + \frac{(P_0 - P)F^{2n}}{1 + (P_0 - P)S(1 - F^{2n})}\,, \quad n \geq 0\,, \quad S = \frac{P+R}{P(P+2R)}$$

where P_0 is an arbitrary positive initial value for $P_{0/-1}$. Since $P_{n/n-1} \to P$ as $n \to \infty$, it follows that also $F_n \to F$ and $K_n \to K$, so that the Kalman filtering equations read,

$$\hat{x}_{n+1/n} = \hat{x}_{n/n} = \hat{x}_{n/n-1} + K_n(y_n - \hat{x}_{n/n}) = F_n\hat{x}_{n/n-1} + (1 - F_n)y_n$$

$$\hat{x}_{n+1/n} = \hat{x}_{n/n} = \hat{x}_{n/n-1} + K(y_n - \hat{x}_{n/n}) = F\hat{x}_{n/n-1} + (1 - F)y_n$$

where the second one is the steady-state version, which is recognized as the exponential smoother with parameter $\lambda = F$. We note that because $P > 0$, we have $0 < F < 1$. □

13.7 Steady-State Kalman Filters

As soon as the Kalman filter gains have converged to their asymptotic values, the Kalman filter can be operated as a time-invariant filter with the following input/output equations for the predicted estimate:

$$\boxed{\begin{aligned} \hat{\mathbf{x}}_{n+1/n} &= A\hat{\mathbf{x}}_{n/n-1} + K(\mathbf{y}_n - C\hat{\mathbf{x}}_{n/n-1}) \\ \hat{\mathbf{x}}_{n+1/n} &= (A - KC)\hat{\mathbf{x}}_{n/n-1} + K\mathbf{y}_n \end{aligned}} \quad \text{(steady-state Kalman filter)} \qquad (13.7.1)$$

or, in its prediction-correction form, where $K = AG$,

$$\boxed{\begin{aligned} \hat{\mathbf{x}}_{n/n} &= \hat{\mathbf{x}}_{n/n-1} + G(\mathbf{y}_n - C\hat{\mathbf{x}}_{n/n-1}) \\ \hat{\mathbf{x}}_{n+1/n} &= A\hat{\mathbf{x}}_{n/n} \end{aligned}} \quad \text{(steady-state Kalman filter)} \tag{13.7.2}$$

or, in its filtered form, using Eq. (13.6.4),

$$\boxed{\begin{aligned} \hat{\mathbf{x}}_{n/n} &= A\hat{\mathbf{x}}_{n-1/n-1} + G(\mathbf{y}_n - CA\hat{\mathbf{x}}_{n-1/n-1}) \\ \hat{\mathbf{x}}_{n/n} &= (A - GCA)\hat{\mathbf{x}}_{n-1/n-1} + G\mathbf{y}_n \end{aligned}} \quad \text{(steady-state Kalman filter)} \tag{13.7.3}$$

Since these depend only on the gains K, G, they may be viewed as state-estimators, or *observers*, independently of the Kalman filter context.

In cases when one does not know the state-model noise parameters Q, R, non-optimal values for the gains K, G may be used (as long as the closed-loop state-transition matrices $F = A - KC$ and $A - GCA$ are stable). Such non-optimal examples include the single and double exponential moving average filters and Holt's exponential smoothing discussed in Chap. 6, as well the general α-β and α-β-γ filters.

The corresponding transfer function matrices from the input $\mathbf{y}_n$ to the prediction $\hat{\mathbf{x}}_{n/n-1}$ and to the filtered estimate $\hat{\mathbf{x}}_{n/n}$ are found by taking z-transforms of Eqs. (13.7.1) and (13.7.2). Denoting the identity matrix by I, we have:

$$\begin{aligned} H_p(z) &= (zI - A + KC)^{-1}K \\ H_f(z) &= z(zI - A + GCA)^{-1}G \end{aligned} \tag{13.7.4}$$

Example 13.7.1: *α-β Tracking Filters.* The kinematic state-space models considered in Example 13.1.3 for a moving object subject to random accelerations were of the form:

$$\begin{aligned} \begin{bmatrix} x_{n+1} \\ \dot{x}_{n+1} \end{bmatrix} &= \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_n \\ \dot{x}_n \end{bmatrix} + \mathbf{w}_n \\ y_n &= [1, 0] \begin{bmatrix} x_n \\ \dot{x}_n \end{bmatrix} + v_n \end{aligned} \tag{13.7.5}$$

with measurement noise variance $R = \sigma_v^2$ and two possible choices for the noise term $\mathbf{w}_n$,

$$\mathbf{w}_n = \begin{bmatrix} 0 \\ w_n \end{bmatrix}, \quad \mathbf{w}_n = \begin{bmatrix} T^2/2 \\ T \end{bmatrix} a_n$$

where w_n represents a random velocity and a_n a random acceleration. The corresponding covariance matrices $Q = E[\mathbf{w}_n \mathbf{w}_n^T]$ are,

$$Q = \begin{bmatrix} 0 & 0 \\ 0 & \sigma_w^2 \end{bmatrix}, \quad Q = \begin{bmatrix} T^4/4 & T^3/2 \\ T^3/2 & T^2 \end{bmatrix} \sigma_a^2$$

An α–β tracking filter is an observer for the model (13.7.5) written in its prediction-correction form of Eq. (13.7.2) with a gain vector defined in terms of the α, β parameters:

$$G = \begin{bmatrix} \alpha \\ \beta/T \end{bmatrix}$$

Eqs. (13.7.2) then read:

$$\begin{aligned} \begin{bmatrix} \hat{x}_{n/n} \\ \hat{\dot{x}}_{n/n} \end{bmatrix} &= \begin{bmatrix} \hat{x}_{n/n-1} \\ \hat{\dot{x}}_{n/n-1} \end{bmatrix} + \begin{bmatrix} \alpha \\ \beta/T \end{bmatrix} (y_n - \hat{x}_{n/n-1}) \\ \begin{bmatrix} \hat{x}_{n+1/n} \\ \hat{\dot{x}}_{n+1/n} \end{bmatrix} &= \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{x}_{n/n} \\ \hat{\dot{x}}_{n/n} \end{bmatrix} \end{aligned} \tag{13.7.6}$$

where we used $\hat{\mathbf{y}}_{n/n-1} = C\hat{\mathbf{x}}_{n/n-1} = [1, 0] \begin{bmatrix} \hat{x}_{n/n-1} \\ \hat{\dot{x}}_{n/n-1} \end{bmatrix} = \hat{x}_{n/n-1}$. Explicitly, we write,

$$\boxed{\begin{aligned} \hat{x}_{n/n} &= \hat{x}_{n/n-1} + \alpha(y_n - \hat{x}_{n/n-1}) \\ \hat{\dot{x}}_{n/n} &= \hat{\dot{x}}_{n/n-1} + \frac{\beta}{T}(y_n - \hat{x}_{n/n-1}) \\ \hat{x}_{n+1/n} &= \hat{x}_{n/n} + T\hat{\dot{x}}_{n/n} \\ \hat{\dot{x}}_{n+1/n} &= \hat{\dot{x}}_{n/n} \end{aligned}} \quad (\alpha\text{–}\beta \text{ tracking filter})$$

These are essentially equivalent to Holt's exponential smoothing method discussed in Sec. 6.12. The corresponding prediction and filtering transfer functions of Eq. (13.7.4) are easily found to be:

$$\begin{aligned} H_p(z) &= \frac{1}{z^2 + (\alpha + \beta - 2)z + 1 - \alpha} \begin{bmatrix} (\alpha + \beta)z - \alpha \\ \beta(z - 1)/T \end{bmatrix} \\ H_f(z) &= \frac{1}{z^2 + (\alpha + \beta - 2)z + 1 - \alpha} \begin{bmatrix} z(\beta - \alpha + \alpha z) \\ \beta z(z - 1)/T \end{bmatrix} \end{aligned} \tag{13.7.7}$$

The particular choices $\alpha = 1 - \lambda^2$ and $\beta = (1 - \lambda)^2$ result in the double-exponential smoothing transfer functions for the local level and local slope of Eq. (6.8.5):

$$H_f(z) = \frac{1}{(1 - \lambda z^{-1})^2} \begin{bmatrix} (1 - \lambda)(1 + \lambda - 2\lambda z^{-1}) \\ (1 - \lambda)^2(1 - z^{-1})/T \end{bmatrix} \tag{13.7.8}$$

The noise-reduction ratios for the position and velocity components of $\mathbf{H}_f(z)$ are easily found to be [870,874]:

$$\mathcal{R}_x = \frac{2\alpha^2 + 2\beta - 3\alpha\beta}{\alpha(4 - 2\alpha - \beta}, \quad \mathcal{R}_{\dot{x}} = \frac{2\beta^2/T^2}{\alpha(4 - 2\alpha - \beta}$$

Example 13.7.2: *α–β Tracking Filters as Kalman Filters.* Optimum values of the α, β parameters can be obtained if one thinks of the α–β tracking filter as the steady-state Kalman filter of the model (13.7.5). We start with the case defined by the parameters,

$$A = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix}, \quad C = [1, 0], \quad \mathbf{w}_n = \begin{bmatrix} 0 \\ w_n \end{bmatrix}, \quad Q = \begin{bmatrix} 0 & 0 \\ 0 & \sigma_w^2 \end{bmatrix}, \quad R = \sigma_v^2$$

Let P denote the solution of the DARE, $P = A(P - GDG^T)A^T + Q$, where the gain G is:

$$P = \begin{bmatrix} P_{11} & P_{12} \\ P_{12} & P_{22} \end{bmatrix}, \quad D = CPC^T + R = P_{11} + R, \quad G = PC^TD^{-1} = \frac{1}{P_{11} + R} \begin{bmatrix} P_{11} \\ P_{12} \end{bmatrix}$$

and we set $P_{21} = P_{12}$. If G is to be identified with the gain of the α–β tracking filter, we must have:

$$G = \frac{1}{P_{11}+R}\begin{bmatrix} P_{11} \\ P_{12}\end{bmatrix} = \begin{bmatrix} \alpha \\ \beta/T\end{bmatrix} \quad \Rightarrow \quad \frac{P_{11}}{P_{11}+R} = \alpha\,, \quad \frac{P_{12}}{P_{11}+R} = \frac{\beta}{T}$$

which may be solved for P_{11}, P_{12}:

$$P_{11} = R\,\frac{\alpha}{1-\alpha}\,, \quad P_{12} = \frac{R}{T}\,\frac{\beta}{1-\alpha}\,, \quad D = \frac{R}{1-\alpha} \tag{13.7.9}$$

The three parameters α, β, P_{22} fix the matrix P completely. The DARE provides three equations from which these three parameters can be determined in terms of the model statistics σ_w^2, σ_v^2. To this end, let us define the so-called *tracking index* [869], as the dimensionless ratio (note that σ_w has units of velocity, and σ_v, units of length):

$$\boxed{\lambda^2 = \frac{\sigma_w^2 T^2}{\sigma_v^2}} \quad \text{(tracking index)} \tag{13.7.10}$$

Using Eqs. (13.7.9), we obtain

$$P - GDG^T = \begin{bmatrix} R\alpha & R\beta/T \\ R\beta/T & P_{22} - \dfrac{\beta^2/T^2}{1-\alpha}\end{bmatrix}$$

Then, the DARE, $P = A(P - GDG^T)A^T + Q$, reads explicitly,

$$\begin{bmatrix} P_{11} & P_{12} \\ P_{12} & P_{22}\end{bmatrix} = \begin{bmatrix} 1 & T \\ 0 & 1\end{bmatrix}\begin{bmatrix} R\alpha & R\beta/T \\ R\beta/T & P_{22} - \dfrac{\beta^2/T^2}{1-\alpha}\end{bmatrix}\begin{bmatrix} 1 & 0 \\ T & 1\end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & \sigma_w^2\end{bmatrix} \tag{13.7.11}$$

Forming the difference of the two sides, we obtain:

$$A(P-GDG^T)A^T + Q - P = \begin{bmatrix} P_{22}T^2 - \dfrac{R\big((\alpha+\beta)^2 - 2\beta\big)}{1-a} & P_{22}T - \dfrac{R\beta(\alpha+\beta)}{T(1-\alpha)} \\ P_{22}T - \dfrac{R\beta(\alpha+\beta)}{T(1-\alpha)} & \sigma_w^2 - \dfrac{R\beta^2}{T^2(1-\alpha)}\end{bmatrix}$$

Equating the off-diagonal matrix elements to zero provides an expression for P_{22}:

$$P_{22} = \frac{R\beta(\alpha+\beta)}{T^2(1-\alpha)}$$

Then, setting the diagonal elements to zero, gives the two equations for α, β:

$$\beta = \frac{\alpha^2}{2-\alpha}\,, \quad \frac{\beta^2}{1-\alpha} = \frac{\sigma_w^2 T^2}{\sigma_v^2} = \lambda^2 \tag{13.7.12}$$

The first of these was arrived at by [870] using different methods. The system (13.7.12) can be solved explicitly in terms of λ^2 as follows [873]:

$$r = \sqrt{\frac{1}{2} + \sqrt{\frac{1}{4} + \frac{4}{\lambda^2}}}\,, \quad \alpha = \frac{2}{r+1}\,, \quad \beta = \frac{2}{r(r+1)} \tag{13.7.13}$$

Next, consider the alternative kinematic model defined by the parameters [868,869]:

$$A = \begin{bmatrix} 1 & T \\ 0 & 1\end{bmatrix}, \quad C = [1, 0]\,, \quad Q = \begin{bmatrix} T^4/4 & T^3/2 \\ T^3/2 & T^2\end{bmatrix}\sigma_a^2\,, \quad R = \sigma_v^2$$

A similar calculation leads to the DARE solution for the covariance matrix:

$$P_{11} = R\,\frac{\alpha}{1-\alpha}\,, \quad P_{12} = \frac{R}{T}\,\frac{\beta}{1-\alpha}\,, \quad P_{22} = \frac{R}{2T^2}\,\frac{\beta(2\alpha+\beta)}{1-\alpha}\,, \quad D = P_{11} + R = \frac{R}{1-\alpha}$$

with α, β satisfying the conditions:

$$\frac{2\beta - \alpha\beta - \alpha^2}{1-\alpha} = \frac{\lambda^2}{4}\,, \quad \frac{\beta^2}{1-\alpha} = \lambda^2 \tag{13.7.14}$$

where now the tracking index is defined as the dimensionless ratio:

$$\lambda^2 = \frac{\sigma_a^2 T^4}{\sigma_v^2} \tag{13.7.15}$$

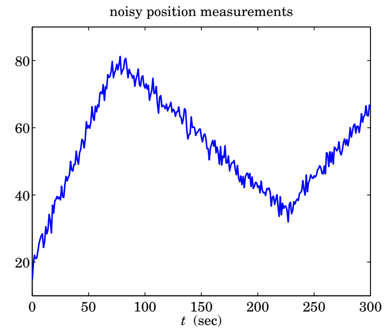
The solution of the system (13.7.14) is found to be [868]:

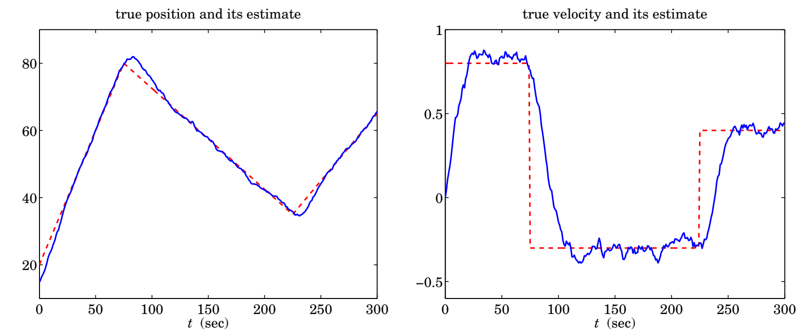
$$r = \sqrt{1 + \frac{8}{\lambda}}\,, \quad \alpha = \frac{4r}{(r+1)^2}\,, \quad \beta = \frac{8}{(r+1)^2} \tag{13.7.16}$$

It is easily verified that these satisfy the Kalata relationship [869]:

$$\beta = 2(2-\alpha) - 4\sqrt{1-\alpha} \tag{13.7.17}$$

For both models, the optimum solutions for α, β given in Eqs. (13.7.13) and (13.7.16) lead to a stable closed-loop matrix $F = A - KC$, that is, its eigenvalues lie inside the unit circle. These eigenvalues are the two roots of the denominator polynomial of the transfer functions (13.7.7), that is, the roots of $z^2 + (\alpha + \beta - 2)z + 1 - \alpha = 0$. The graphs below show a simulation.





The following parameter values were chosen $\sigma_a = 0.02$, $\sigma_v = 2$, $T = 1$, which lead to a tracking index (13.7.15) of $\lambda = 0.01$, and the value of the parameter $r = 28.3019$ in Eq. (13.7.16), which gives the $\alpha\beta$-parameters $\alpha = 0.1319$ and $\beta = 0.0093$. The algorithm (13.7.6) was iterated with an initial value $\hat{\mathbf{x}}_{0/-1} = [y_0, 0]^T$.

The following MATLAB code segment shows the generation of the input signal y_n, the computation of α, β, and the filtering operation:

```
t0 = 0;   t1 = 75;   t2 = 225; t3 = 300;   % turning times
b0 = 0.8; b1 = -0.3; b2 = 0.4;             % segment slopes

m0 = 20;
m1 = m0 + b0 * (t1-t0);                   % segment turning points
m2 = m1 + b1 * (t2-t1);

t = (t0:t3); T=1;

s = (m0+b0*t).*upulse(t,t1) + (m1+b1*(t-t1)).*upulse(t-t1,t2-t1) +...
    (m2+b2*(t-t2)).*upulse(t-t2,t3-t2+1);

sdot = b0*upulse(t,t1) + b1*upulse(t-t1,t2-t1) + b2*upulse(t-t2,t3-t2+1);

seed = 1000; randn('state',seed);
sv = 2; v = sv * randn(1,length(t));

y = s + v;                                % noisy position measurements

sa = 0.02; lambda = sa*T^2/sv; r = sqrt(1+8/lambda);

a = 4*r/(r+1)^2; b = 8/(r+1)^2;                 % alpha-beta parameters

A = [1, T; 0, 1]; C=[1,0]; G = [a; b/T];

xp = [y(1); 0];                               % initial prediction

for n=1:length(t),
   x(:,n) = xp + G*(y(n) - C*xp);
   xp = A*x(:,n);
end

figure; plot(t,y,'b-');                       % noisy positions
figure; plot(t,s,'r--', t,x(1,:),'b-');       % true & estimated position
figure; plot(t,sdot,'r--', t,x(2,:),'b-');    % true & estimated velocity
```

Example 13.7.3: *Transients of α-β Tracking Kalman Filters.* Here, we look at a simulation of the random-acceleration model of Eq. (13.1.15) and of the time-varying Kalman filter as it converges to steady-state. The model is defined by

$$\begin{bmatrix} x_{n+1} \\ \dot{x}_{n+1} \end{bmatrix} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_n \\ \dot{x}_n \end{bmatrix} + \begin{bmatrix} T^2/2 \\ T \end{bmatrix} a_n\,, \quad y_n = [1,0] \begin{bmatrix} x_n \\ \dot{x}_n \end{bmatrix} + v_n$$

with model matrices:

$$A = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix}, \quad C = [1,0]\,, \quad Q = \begin{bmatrix} T^4/4 & T^3/2 \\ T^3/2 & T^2 \end{bmatrix} \sigma_a^2\,, \quad R = \sigma_v^2$$

The corresponding Kalman filter is defined by Eq. (13.6.3). If we denote the elements of the time-varying Kalman gain G_n by

$$G_n = \begin{bmatrix} \alpha_n \\ \beta_n/T \end{bmatrix}$$

then, we expect α_n, β_n to eventually converge to the steady-state values given in (13.7.16). The Kalman filtering algorithm reads explicitly,

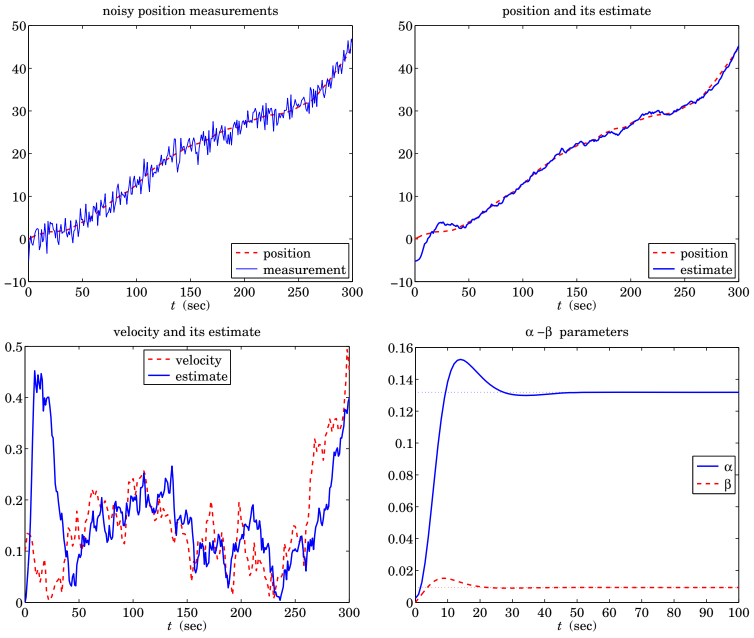
$$\begin{bmatrix} \hat{x}_{n/n} \\ \hat{\dot{x}}_{n/n} \end{bmatrix} = \begin{bmatrix} \hat{x}_{n/n-1} \\ \hat{\dot{x}}_{n/n-1} \end{bmatrix} + \begin{bmatrix} \alpha_n \\ \beta_n/T \end{bmatrix} (y_n - \hat{x}_{n/n-1})\,, \quad \begin{bmatrix} \hat{x}_{n+1/n} \\ \hat{\dot{x}}_{n+1/n} \end{bmatrix} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{x}_{n/n} \\ \hat{\dot{x}}_{n/n} \end{bmatrix}$$

where

$$D_n = CP_{n/n-1}C^T + R\,, \quad G_n = P_{n/n-1}C^T/D_n$$

$$P_{n/n} = P_{n/n-1} - G_n D_n G_n^T\,, \quad P_{n+1/n} = AP_{n/n}A^T + Q$$

The figures below show a simulation with the same parameter values as in the previous example, $\sigma_a = 0.02$, $\sigma_v = 2$, and $T = 1$.



The upper-left graph shows the noisy measurement y_n plotted together with the position x_n to be estimated. The upper-right graph plots the estimate $\hat{x}_{n/n}$ together with x_n. The lower-left graph shows the velocity and its estimate, $\dot{x}_n$ and $\hat{\dot{x}}_{n/n}$. The lower-right graph shows α_n, β_n as they converge to their steady values $\alpha = 0.1319$, $\beta = 0.0093$, which were calculated from (13.7.16):

$$\lambda = \frac{\sigma_a T^2}{\sigma_v} = 0.01\,, \quad r = \sqrt{1 + \frac{8}{\lambda}} = 28.3019\,, \quad \alpha = \frac{4r}{(r+1)^2}\,, \quad \beta = \frac{8}{(r+1)^2}$$

The model was simulated by generating two independent, zero-mean, gaussian, length-300 acceleration and measurement noise signals a_n, v_n. The initial state vector was chosen at zero position, but with a finite velocity,

$$\mathbf{x}_0 = \begin{bmatrix} x_0 \\ \dot{x}_0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0.1 \end{bmatrix}$$

The Kalman filter was initialized to the following predicted state vector and covariance:

$$\hat{\mathbf{x}}_{0/-1} = \begin{bmatrix} \hat{x}_{0/-1} \\ \hat{\dot{x}}_{0/-1} \end{bmatrix} = \begin{bmatrix} y_0 \\ 0 \end{bmatrix}, \quad P_{0/-1} = \begin{bmatrix} 0.01 & 0 \\ 0 & 0.01 \end{bmatrix}$$

The following MATLAB code illustrates the generation of the graphs:

```
N = 301; T = 1; Tmax = (N-1)*T; t = 0:T:Tmax;

seed = 1000; randn(’state’,seed);

sv = 2; sa = 0.02; lambda = sa*T^2/sv; r = sqrt(1+8/lambda);
a = 4*r/(r+1)^2; b = 8/(r+1)^2;

v = sv * randn(1,length(t));                % measurement noise
w = [T^2/2; T] * sa * randn(1,length(t));   % state noise

R = sv^2; Q = [T^4/4, T^3/2; T^3/2, T^2]*sa^2;

A = [1,T; 0,1]; C = [1,0];

x0 = [0; 0.1]; x(:,1) = x0;         % initial state

for n=1:N-1                         % generate states and measurements
   x(:,n+1) = A*x(:,n) + w(n);
   y(n) = C*x(:,n) + v(n);
end
   y(N) = C*x(:,N) + v(N);

xp = [y(1);0]; P0 = diag([1,1]/100); P = P0;    % initialize Kalman filter

for n=1:length(t),                              % run Kalman filter
   D = C*P*C’+R;
   G(:,n) = P*C’/D;                             % G = Kalman gain
   X(:,n) = xp + G(:,n)*(y(n) - C*xp);          % X(:,n) = filtered state
   Pf = P - G(:,n)*D*G(:,n)’;                   % error covariance of X
   xp = A*X(:,n);                               % xp = predicted state
   P = A*Pf*A’ + Q;                             % error covariance of xp
end
```

```
figure; plot(t,x(1,:),’r--’, t,y,’b-’);
figure; plot(t,x(1,:),’r--’, t,X(1,:),’b-’);
figure; plot(t,x(2,:),’r--’, t,X(2,:),’b-’);
figure; plot(t,G(1,:),’b-’, t,G(2,:),’r--’);
```

The eigenvalues of the asymptotic closed-loop state matrix $F = A - KC$, which are the roots of the polynomial, $z^2 + (\alpha + \beta_2)z + (1 - \alpha)$, can be expressed directly in terms of the parameter r as follows:

$$\lambda_1 = \frac{r^2 - 3 - 2j\sqrt{r^2 - 2}}{(r+1)^2}\,, \quad \lambda_2 = \frac{r^2 - 3 + 2j\sqrt{r^2 - 2}}{(r+1)^2}$$

The eigenvalues are complex conjugates whenever $r^2 > 2$, or equivalently, when the tracking index is $\lambda < 8$, which is usually the case in practice. For $\lambda \ge 8$, or $1 < r^2 \le 2$, they are real-valued. For the complex case, the eigenvalues have magnitude:

$$|\lambda_1| = |\lambda_2| = \frac{r-1}{r+1}$$

One can then determine an estimate of the convergence time-constant of the Kalman filter,

$$n_{\text{eff}} = \frac{\ln \epsilon}{2 \ln\left(\dfrac{r-1}{r+1}\right)}$$

For the present example, we find $n_{\text{eff}} = 49$ samples for the 60-dB time constant ($\epsilon = 10^{-3}$), which is evident from the above plot of α_n, β_n. Using the methods of [871,872] one may also construct closed-form solutions for the time-varying covariance $P_{n/n-1}$ and gain G_n. First, we determine the eigenvector decomposition of F:

$$F = A - KC = \frac{1}{(r+1)^2} \begin{bmatrix} r^2 - 2r - 7 & T(r+1)^2 \\ -8T & (r+1)^2 \end{bmatrix} = V\Lambda V^{-1}$$

$$V = \begin{bmatrix} v_1 & v_2 \\ 1 & 1 \end{bmatrix}, \quad \Lambda = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}, \quad V^{-1} = \frac{1}{v_1 - v_2} \begin{bmatrix} 1 & -v_2 \\ -1 & v_1 \end{bmatrix}$$

$$v_1 = \frac{T}{4}\left[r + 2 - j\sqrt{r^2 - 2}\right], \quad v_2 = \frac{T}{4}\left[r + 2 + j\sqrt{r^2 - 2}\right]$$

Then, the nth power of F is given by:

$$F = V\Lambda V^{-1} = \frac{1}{v_1 - v_2} \begin{bmatrix} v_1\lambda_1 - v_2\lambda_2 & v_1 v_2(\lambda_2 - \lambda_1) \\ \lambda_1 - \lambda_2 & v_1\lambda_2 - v_2\lambda_1 \end{bmatrix}$$

$$F^n = V\Lambda^n V^{-1} = \frac{1}{v_1 - v_2} \begin{bmatrix} v_1\lambda_1^n - v_2\lambda_2^n & v_1 v_2(\lambda_2^n - \lambda_1^n) \\ \lambda_1^n - \lambda_2^n & v_1\lambda_2^n - v_2\lambda_1^n \end{bmatrix}$$

The converged steady-state value of $P_{n/n-1}$, which is the solution of the DARE, may also be expressed in terms of the parameter r, as follows:

$$P = \frac{4rR}{(r+1)^2} \begin{bmatrix} 1 & \dfrac{2}{Tr} \\ \dfrac{2}{Tr} & \dfrac{8}{T^2 r(r+1)} \end{bmatrix}$$

Given an initial 2×2 matrix $P_{0/-1}$, we construct the matrix $E_0 = P_{0/-1} - P$. Then, the exact solution of the Riccati difference equation is given by:

$$P_{n/n-1} = P + F^n E_0 \big[I + S_n E_0\big]^{-1} F^{Tn}$$

where I is the 2×2 identity matrix and S_n is defined as follows, for $n \ge 0$,

$$S_n = S - F^{Tn} S F^n\,, \quad S = \frac{r^2-1}{8rR}\begin{bmatrix} 1 & -T/2 \\ -T/2 & T^2(r^2+1)/8 \end{bmatrix}$$

These expressions demonstrate why the convergence time-constant depends on the square of the maximum eigenvalue of F. □

Example 13.7.4: *Local Trend Model.* Holt's exponential smoothing model is an effective way of tracking the local level a_n and local slope b_n of a signal and represents a generalization of the double exponential moving average (DEMA) model. Its state-space form was considered briefly in Sec. 6.13. The following time-invariant linear trend state-space model has steady-state Kalman filtering equations that are equivalent to Holt's method,

$$\begin{bmatrix} a_{n+1} \\ b_{n+1} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} a_n \\ b_n \end{bmatrix} + \begin{bmatrix} w_n \\ u_n \end{bmatrix}\,, \quad y_n = [1, 0]\begin{bmatrix} a_n \\ b_n \end{bmatrix} + v_n \tag{13.7.18}$$

so that its state-model matrices are,

$$A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}\,, \quad C = [1, 0]$$

where a_n, b_n represent the local level and local slope, and w_n, u_n, v_n are zero-mean, mutually uncorrelated, white-noise signals with variances $Q_a = \sigma_w^2$, $Q_b = \sigma_u^2$, $R = \sigma_v^2$. Denote the state vector and its filtered and predicted estimates by,

$$\mathbf{x}_n = \begin{bmatrix} a_n \\ b_n \end{bmatrix}\,, \quad \hat{\mathbf{x}}_{n/n} = \begin{bmatrix} \hat{a}_{n/n} \\ \hat{b}_{n/n} \end{bmatrix}\,, \quad \hat{\mathbf{x}}_{n+1/n} = \begin{bmatrix} \hat{a}_{n+1/n} \\ \hat{b}_{n+1/n} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} \hat{a}_{n/n} \\ \hat{b}_{n/n} \end{bmatrix}$$

so that,

$$\hat{a}_{n+1/n} = \hat{a}_{n/n} + \hat{b}_{n/n}\,, \quad \hat{b}_{n+1/n} = \hat{b}_{n/n}$$

Then, the predicted measurement can be expressed in two ways as follows,

$$\hat{y}_{n/n-1} = C\hat{\mathbf{x}}_{n/n-1} = [1, 0]\begin{bmatrix} \hat{a}_{n/n-1} \\ \hat{b}_{n/n-1} \end{bmatrix} = \hat{a}_{n/n-1} = \hat{a}_{n-1/n-1} + \hat{b}_{n-1/n-1}$$

Denote the two-dimensional steady-state Kalman gains G and $K = AG$ by,

$$G = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}\,, \quad K = AG = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \alpha + \beta \\ \beta \end{bmatrix}$$

Then, the steady-state Kalman filtering equations Eq. (13.7.1) and (13.7.3) take the form,

$$\begin{aligned} \hat{\mathbf{x}}_{n/n} &= A\hat{\mathbf{x}}_{n-1/n-1} + G(y_n - \hat{y}_{n/n-1}) \\ \hat{\mathbf{x}}_{n+1/n} &= A\hat{\mathbf{x}}_{n/n-1} + K(y_n - \hat{y}_{n/n-1}) \end{aligned}$$

which are precisely Holt's exponential smoothing formulas,

$$\begin{aligned} \begin{bmatrix} \hat{a}_{n/n} \\ \hat{b}_{n/n} \end{bmatrix} &= \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} \hat{a}_{n-1/n-1} \\ \hat{b}_{n-1/n-1} \end{bmatrix} + \begin{bmatrix} \alpha \\ \beta \end{bmatrix}(y_n - \hat{a}_{n-1/n-1} - \hat{b}_{n-1/n-1}) \\ \begin{bmatrix} \hat{a}_{n+1/n} \\ \hat{b}_{n+1/n} \end{bmatrix} &= \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} \hat{a}_{n/n-1} \\ \hat{b}_{n/n-1} \end{bmatrix} + \begin{bmatrix} \alpha + \beta \\ \beta \end{bmatrix}(y_n - \hat{a}_{n/n-1}) \end{aligned}$$

The corresponding model parameters Q_a, Q_b and the error covariance matrix P can be reconstructed in terms of R and the gains α, β, as follows,

$$Q = \begin{bmatrix} Q_a & 0 \\ 0 & Q_b \end{bmatrix} = \frac{R}{1-\alpha}\begin{bmatrix} \alpha^2 + \alpha\beta - 2\beta & 0 \\ 0 & \beta^2 \end{bmatrix}\,, \quad P = \frac{R}{1-\alpha}\begin{bmatrix} \alpha & \beta \\ \beta & \beta(\alpha+\beta) \end{bmatrix}$$

One can easily verify, according to Eq. (13.6.5), that,

$$D = CPC^T + R = \frac{R}{1-\alpha}\,, \quad G = PC^T D^{-1} = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$$

and that P satisfies the algebraic Riccati equation (13.6.6), that is,

$$P = APA^T - APC^T(CPC^T + R)^{-1}CPA^T + Q$$

Assuming that α, β are positive and that $\alpha < 1$, the positivity of Q_a requires the condition, $\alpha^2 + \alpha\beta > 2\beta$, which also implies that P is positive definite since its determinant is,

$$\det P = R^2\,\frac{(\alpha^2 + \alpha\beta - \beta)\beta}{(1-\alpha)^2} = R^2\,\frac{(\alpha^2 + \alpha\beta - 2\beta + \beta)\beta}{(1-\alpha)^2} > 0$$

Thus, Holt's method admits a Kalman filtering interpretation. □

13.8 Continuous-Time Kalman Filter

The continuous-time Kalman filter, known as the Kalman-Bucy filter [853], is based on the state-space model:

$$\boxed{\begin{aligned} \dot{\mathbf{x}}(t) &= A(t)\mathbf{x}(t) + \mathbf{w}(t) \\ \mathbf{y}(t) &= C(t)\mathbf{x}(t) + \mathbf{v}(t) \end{aligned}} \tag{13.8.1}$$

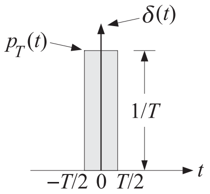
where $\mathbf{w}(t), \mathbf{v}(t)$ are mutually uncorrelated white-noise signals with covariances:

$$\boxed{\begin{aligned} E[\mathbf{w}(t)\mathbf{w}(\tau)^T] &= Q(t)\delta(t-\tau) \\ E[\mathbf{v}(t)\mathbf{v}(\tau)^T] &= R(t)\delta(t-\tau) \\ E[\mathbf{w}(t)\mathbf{v}(\tau)^T] &= 0 \end{aligned}} \tag{13.8.2}$$

We assume also that $\mathbf{w}(t), \mathbf{v}(t)$ are uncorrelated with the initial state vector $\mathbf{x}(0)$. More precisely, one should write the stochastic differential equation for the state in the form: $d\mathbf{x}(t)= A(t)\mathbf{x}(t)dt + \mathbf{w}(t)dt$ and view the quantity $d\mathbf{b}(t)= \mathbf{w}(t)dt$ as a vector-valued Brownian motion, with covariance $E[d\mathbf{b}(t)d\mathbf{b}(t)^T]= Q(t)dt$. However, for the above linear model, the formal manipulations using $\mathbf{w}(t)$ lead to equivalent results.

The continuous-time Kalman filter can be obtained from the discrete one in the limit as the sampling interval T tends to zero. One of the issues that arises is how to define the white-noise sequences $\mathbf{w}_n, \mathbf{v}_n$ of the discrete-time model of Eq. (13.1.1) in terms of the sampled values of the continuous-time signals $\mathbf{w}(t), \mathbf{v}(t)$. The covariance matrix at a single time instant is not well-defined for white noise signals. Indeed, setting $t = \tau = t_n = nT$ in Eq. (13.8.2) would give an infinite value for the covariance $E[\mathbf{w}(t_n)\mathbf{w}(t_n)^T]$.

Since the delta function $\delta(t)$ can be thought of as the limit as $T \to 0$ of a square pulse function $p_T(t)$ of width T and height $1/T$ shown below, we may replace $\delta(t-\tau)$ by $p_T(t-\tau)$ in the right-hand-side of Eq. (13.8.2).



This leads to the approximate but finite values:

$$E[\mathbf{w}(t_n)\mathbf{w}(t_n)^T]= \frac{Q(t_n)}{T}, \quad E[\mathbf{v}(t_n)\mathbf{v}(t_n)^T]= \frac{R(t_n)}{T} \tag{13.8.3}$$

The same conclusion can be reached from the Brownian motion point of view, which would give formally,

$$E[\mathbf{w}(t)\mathbf{w}(t)^T]= E\left[\frac{d\mathbf{b}(t)}{dt}\,\frac{d\mathbf{b}(t)^T}{dt}\right] = \frac{E[d\mathbf{b}(t)\,d\mathbf{b}(t)^T]}{dt^2} = \frac{Q(t)dt}{dt^2} = \frac{Q(t)}{dt}$$

and identifying dt by the sampling time T. We may apply now Eq. (13.8.3) to the sampled version of the measurement equation:

$$\mathbf{y}(t_n)= C(t_n)\mathbf{x}(t_n)+\mathbf{v}(t_n)$$

Thus, the discrete-time measurement noise signal $\mathbf{v}_n$ is identified as $\mathbf{v}(t_n)$, with covariance matrix:

$$E[\mathbf{v}_n\mathbf{v}_n^T]= R_n = \frac{R(t_n)}{T} \tag{13.8.4}$$

To identify $\mathbf{w}_n$, we consider the discretized version of the state equation:

$$\dot{\mathbf{x}}(t_n)\approx \frac{\mathbf{x}(t_{n+1})-\mathbf{x}(t_n)}{T} = A(t_n)\mathbf{x}(t_n)+\mathbf{w}(t_n)$$

which gives,

$$\mathbf{x}(t_{n+1})= [I + TA(t_n)]\mathbf{x}(t_n)+T\mathbf{w}(t_n)$$

and we may identify the discrete-time model quantities:

$$A_n = I + TA(t_n), \quad \mathbf{w}_n = T\mathbf{w}(t_n)$$

with noise covariance matrix $E[\mathbf{w}_n\mathbf{w}_n^T]= T^2E[\mathbf{w}(t_n)\mathbf{w}(t_n)^T]$, or using (13.8.3),

$$E[\mathbf{w}_n\mathbf{w}_n^T]= T^2 \cdot \frac{Q(t_n)}{T} = TQ(t_n)\equiv Q_n \tag{13.8.5}$$

To summarize, for small T, the discrete-time and continuous-time signals and model parameters are related by

$$\begin{gathered}\mathbf{x}_n = \mathbf{x}(t_n), \quad \mathbf{w}_n = T\mathbf{w}(t_n), \quad \mathbf{y}_n = \mathbf{y}(t_n), \quad \mathbf{v}_n = \mathbf{v}(t_n) \\ A_n = I + TA(t_n), \quad C_n = C(t_n), \quad Q_n = TQ(t_n), \quad R_n = \frac{R(t_n)}{T}\end{gathered} \tag{13.8.6}$$

Next, consider the limit of the discrete-time Kalman filter. Since $T \to 0$, there will be no distinction between $P_{n/n}$ and $P_{n/n-1}$, and we may set $P(t_n)\approx P_{n/n} \approx P_{n/n-1}$. Using (13.8.6), the innovations covariance and the Kalman gains become approximately (to lowest order in T):

$$\begin{aligned}D_n &= C_nP_{n/n-1}C_n^T + R_n = C(t_n)P(t_n)C(t_n)+\frac{R(t_n)}{T} \approx \frac{R(t_n)}{T} \\ G_n &= P_{n/n-1}C_n^TD_n^{-1} = P(t_n)C(t_n)^TR(t_n)^{-1}T \equiv K(t_n)T \\ K_n &= A_nP_{n/n-1}C_n^TD_n^{-1} = [I + TA(t_n)]P(t_n)C(t_n)^TR(t_n)^{-1}T \approx K(t_n)T\end{aligned}$$

where we set $K(t_n)= P(t_n)C(t_n)^TR(t_n)^{-1}$. Setting $\hat{\mathbf{x}}(t_n)= \hat{\mathbf{x}}_{n/n-1}$ and hence $\hat{\mathbf{x}}(t_{n+1})= \hat{\mathbf{x}}_{n+1/n}$, the Kalman filtering equation $\hat{\mathbf{x}}_{n+1/n} = A_n\hat{\mathbf{x}}_{n/n-1} + K_n\boldsymbol{\varepsilon}_n$ becomes:

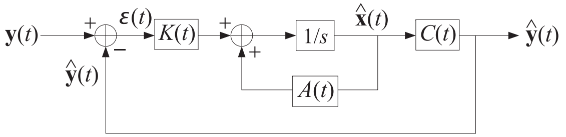
$$\hat{\mathbf{x}}(t_{n+1})= [I + TA(t_n)]\hat{\mathbf{x}}(t_n)+TK(t_n)\boldsymbol{\varepsilon}(t_n), \quad \text{or,}$$

$$\frac{\hat{\mathbf{x}}(t_{n+1})-\hat{\mathbf{x}}(t_n)}{T} = A(t_n)\hat{\mathbf{x}}(t_n)+K(t_n)\boldsymbol{\varepsilon}(t_n)= A(t_n)\hat{\mathbf{x}}(t_n)+K(t_n)\big[\mathbf{y}(t_n)-C(t_n)\hat{\mathbf{x}}(t_n)\big]$$

which becomes the differential equation in the limit $T \to 0$:

$$\dot{\hat{\mathbf{x}}}(t)= A(t)\hat{\mathbf{x}}(t)+K(t)\boldsymbol{\varepsilon}(t)= A(t)\hat{\mathbf{x}}(t)+K(t)\big[\mathbf{y}(t)-C(t)\hat{\mathbf{x}}(t)\big]$$

with a realization depicted below.



Finally, we consider the limiting form of the Riccati difference equation:

$$P_{n+1/n} = A_n[P_{n/n-1} - G_nD_nG_n^T]A_n^T + Q_n$$

Using Eqs. (13.8.6) and noting that $P_{n+1/n} = P(t_{n+1})$, we obtain:

$$P(t_{n+1})= [I + TA(t_n)]\big[P(t_n)-K(t_n)TR(t_n)T^{-1}TK(t_n)^T\big]\big[I + TA(t_n)^T\big] + TQ(t_n)$$

which may be written to order T as follows:

$$\frac{P(t_{n+1})-P(t_n)}{T} = A(t_n)P(t_n)+P(t_n)A(t_n)^T-K(t_n)R(t_n)K(t_n)^T+Q(t_n)$$

which becomes the differential equation in the limit $T\to 0$,

$$\dot{P}(t)=A(t)P(t)+P(t)A(t)^T-K(t)R(t)K(t)^T+Q(t)$$

Substituting $K(t)=P(t)C(t)^TR(t)^{-1}$, we obtain the *Riccati differential equation*:

$$\dot{P}(t)=A(t)P(t)+P(t)A(t)^T-P(t)C(t)^TR(t)^{-1}C(t)P(t)+Q(t)$$

To summarize, the continuous-time Kalman filter for the model (13.8.1) is given by:

$$\boxed{\begin{aligned} K(t) &= P(t)C(t)^TR(t)^{-1}\\ \dot{\hat{\mathbf{x}}}(t) &= A(t)\hat{\mathbf{x}}(t)+K(t)\big[\mathbf{y}(t)-C(t)\hat{\mathbf{x}}(t)\big]\\ \dot{P}(t) &= A(t)P(t)+P(t)A(t)^T-P(t)C(t)^TR(t)^{-1}C(t)P(t)+Q(t)\end{aligned}} \qquad (13.8.7)$$

where $P(t)$ represents the covariance $E[\mathbf{e}(t)\mathbf{e}(t)^T]$ of the estimation error $\mathbf{e}(t)=\mathbf{x}(t)-\hat{\mathbf{x}}(t)$, and the initial conditions are taken to be:

$$\hat{\mathbf{x}}(0)=E[\mathbf{x}(0)]\,,\quad P(0)=E\big[(\mathbf{x}(0)-\hat{\mathbf{x}}(0))(\mathbf{x}(0)-\hat{\mathbf{x}}(0))^T\big]$$

For time-invariant models, i.e. with time-independent model parameters $\{A,C,Q,R\}$, and under the same type of complete stabilizability and detectability assumptions as in the discrete-time case, the Riccati solution $P(t)$ tends to the unique positive-semidefinite symmetric solution P of the *continuous-time algebraic Riccati equation* (CARE):

$$\boxed{AP+PA^T-PC^TR^{-1}CP+Q=0}\quad \text{(CARE)} \qquad (13.8.8)$$

and the Kalman gain tends to the corresponding steady gain $K(t)\to K\equiv PC^TR^{-1}$, resulting in a strictly stable closed-loop state matrix $F=A-KC$, i.e., with eigenvalues in the left-hand s-plane.

Example 13.8.1: *Local Level Model.* The continuous-time version of the local-level model of Example 13.6.2 is defined by the one-dimensional model:

$$\begin{aligned} &\dot{x}(t)=w(t)\,, && E[w(t)w(\tau)]=Q\delta(t-\tau)\\ &y(t)=x(t)+v(t)\,, && E[v(t)v(\tau)]=R\delta(t-\tau)\end{aligned}$$

It represents a Wiener (Brownian) process $x(t)$ observed in noise. The Kalman filtering algorithm (13.8.7) has parameters $A=0$, $C=1$, and satisfies the Riccati differential and algebraic equations:

$$\dot{P}(t)=Q-\frac{P^2(t)}{R}\,,\quad Q-\frac{P^2}{R}=0\quad\Rightarrow\quad P=\sqrt{QR}$$

and has time-varying and steady gains:

$$K(t)=\frac{P(t)}{R}\quad\Rightarrow\quad K=\frac{P}{R}=\sqrt{\frac{Q}{R}}$$

and closed-loop transition matrices:

$$F(t)=-K(t)\quad\Rightarrow\quad F=-K=-\sqrt{\frac{Q}{R}}$$

and time-varying and steady Kalman filtering equations:

$$\begin{aligned}\dot{\hat{x}}(t) &= -K(t)\hat{x}(t)+K(t)y(t)\\ \dot{\hat{x}}(t) &= -K\hat{x}(t)+Ky(t)\end{aligned}$$

with the latter representing a continuous-time version of the exponential smoother. The convergence properties to the steady values depend on F. Using the methods of [871,872], we find the exact solution for $P(t)$, and hence $K(t)=P(t)/R$:

$$P(t)=P+\frac{2P(P_0-P)e^{2Ft}}{P_0+P-(P_0-P)e^{2Ft}}\,,\quad t\geq 0$$

where P_0 is an arbitrary positive initial value for $P(0)$, and $e^{2Ft}=e^{-2Kt}$, which decays to zero exponentially with a time constant determined by $2F$, a result analogous to the discrete-time case. □

13.9 Equivalence of Kalman and Wiener Filtering

We saw in Chap. 11 that for the case of a simple scalar state-space model the steady-state Kalman filter was equivalent to the corresponding Wiener filter, and that the innovations signal model of the observation signal was embedded in the Kalman filter. Similar results can be derived in the multichannel case.

Consider the problem of estimating a p-dimensional vector-valued signal $\mathbf{x}_n$ from an r-dimensional signal of observations $\mathbf{y}_n$. In the stationary case, the solution of this problem depends on the following $p\times r$ cross-correlation and $r\times r$ autocorrelation functions and corresponding z-transform spectral densities:

$$\begin{aligned} R_{xy}(k) &= E[\mathbf{x}_n\mathbf{y}_{n-k}^T]\,, && S_{xy}(z)=\sum_{k=-\infty}^{\infty}R_{xy}(k)z^{-k}\\ R_{yy}(k) &= E[\mathbf{y}_n\mathbf{y}_{n-k}^T]\,, && S_{yy}(z)=\sum_{k=-\infty}^{\infty}R_{yy}(k)z^{-k}\end{aligned} \qquad (13.9.1)$$

The desired causal estimate of $\mathbf{x}_n$ is given by the convolutional equation:

$$\hat{\mathbf{x}}_n=\sum_{k=0}^{\infty}h_k\mathbf{y}_{n-k}\,,\quad H(z)=\sum_{k=0}^{\infty}h_kz^{-k} \qquad (13.9.2)$$

where h_k is the optimum $p\times r$ causal impulse response matrix to be determined. The optimality conditions are equivalent to the orthogonality between the estimation error $\mathbf{e}_n = \mathbf{x}_n - \hat{\mathbf{x}}_n$ and the observations $\mathbf{y}_{n-k}$, $k \ge 0$, that make up the estimate:

$$R_{ey}(k) = E[\mathbf{e}_n\mathbf{y}_{n-k}^T] = 0\,, \quad k \ge 0 \tag{13.9.3}$$

These are equivalent to the matrix-valued Wiener-Hopf equations for h_k:

$$\sum_{m=0}^{\infty} h_m R_{yy}(k-m) = R_{xy}(k)\,, \quad k \ge 0 \tag{13.9.4}$$

The solution can be constructed in the z-domain with the help of a causal and causally invertible signal model $B(z)$ for the observations $\mathbf{y}_n$, driven by an r-dimensional white-noise sequence $\boldsymbol{\varepsilon}_n$ of (time-independent) covariance $E[\boldsymbol{\varepsilon}_n\boldsymbol{\varepsilon}_{n-k}^T] = D\delta(k)$, that is,

$$\mathbf{y}_n = \sum_{k=0}^{\infty} b_k \boldsymbol{\varepsilon}_{n-k}\,, \quad B(z) = \sum_{k=0}^{\infty} b_k z^{-k} \tag{13.9.5}$$

where b_k is the causal $r\times r$ impulse response matrix of the model. The model implies the spectral factorization of observations spectral density $S_{yy}(z)$:

$$\boxed{S_{yy}(z) = B(z)DB^T(z^{-1})} \tag{13.9.6}$$

We will construct the optimal filter $H(z)$ using the gapped-function technique of Chap. 11. To this end, we note that for any k,

$$\begin{aligned} R_{ey}(k) &= E[\mathbf{e}_n\mathbf{y}_{n-k}^T] = E[(\mathbf{x}_n - \hat{\mathbf{x}}_n)\mathbf{y}_{n-k}^T] = E\left[\left(\mathbf{x}_n - \sum_{m=0}^{\infty} h_m\mathbf{y}_{n-m}\right)\mathbf{y}_{n-k}^T\right] \\ &= R_{xy}(k) - \sum_{m=0}^{\infty} h_m R_{yy}(k-m) \end{aligned}$$

and in the z-domain:

$$S_{ey}(z) = S_{xy}(z) - H(z)S_{yy}(z) = S_{xy}(z) - H(z)B(z)DB^T(z^{-1})$$

The orthogonality conditions (13.9.3) require that $S_{ey}(z)$ be a strictly left-sided, or anticausal z-transform, and so will be the z-transform obtained by multiplying both sides by the matrix inverse of $B^T(z^{-1})$ because we assumed the $B(z)$ and $B^{-1}(z)$ are causal, the therefore, $B(z^{-1})$ and $B^{-1}(z^{-1})$ will be anticausal. Thus,

$$S_{ey}(z)B^{-T}(z^{-1}) = S_{xy}(z)B^{-T}(z^{-1}) - H(z)B(z)D = \text{strictly anticausal}$$

and therefore, its causal part will be zero:

$$\Big[S_{xy}(z)B(z^{-1})^{-T} - H(z)B(z)D\Big]_+ = \Big[S_{xy}(z)B^{-T}(z^{-1})\Big]_+ - [H(z)B(z)D]_+ = 0$$

Removing the causal instruction from the second term because it is already causal, we may solve for the optimum Wiener filter for estimating $\mathbf{x}_n$ from $\mathbf{y}_n$:

$$\boxed{H(z) = \Big[S_{xy}(z)B^{-T}(z^{-1})\Big]_+ D^{-1}B^{-1}(z)} \quad \text{(multichannel Wiener filter)} \tag{13.9.7}$$

This generalizes the results of Chap. 11 to vector-valued signals. Similarly, we may obtain for the minimized value of the estimation error covariance:

$$E[\mathbf{e}_n\mathbf{e}_n^T] = R_{ee}(0) = \oint_{\text{u.c.}} [S_{xx}(z) - H(z)S_{yx}(z)]\,\frac{dz}{2\pi jz} \tag{13.9.8}$$

The results may be applied to the one-step-ahead prediction problem by replacing the signal $\mathbf{x}_n$ by the signal $\mathbf{x}_1(n) = \mathbf{x}_{n+1}$. Noting that $\mathbf{X}_1(z) = z\mathbf{X}(z)$, we have:

$$H_1(z) = \Big[S_{x_1y}(z)B^{-T}(z^{-1})\Big]_+ D^{-1}B^{-1}(z)$$

and since $S_{x_1y}(z) = zS_{xy}(z)$, we find:

$$\boxed{H_1(z) = \Big[zS_{xy}(z)B^{-T}(z^{-1})\Big]_+ D^{-1}B^{-1}(z)} \quad \text{(prediction filter)} \tag{13.9.9}$$

and for the covariance of the error $\mathbf{e}_{n+1/n} = \mathbf{x}_n - \hat{\mathbf{x}}_1(n) = \mathbf{x}_n - \hat{\mathbf{x}}_{n+1/n}$,

$$E[\mathbf{e}_{n+1/n}\mathbf{e}_{n+1/n}^T] = \oint \Big[S_{xx}(z) - H_1(z)S_{yx}(z)z^{-1}\Big]\,\frac{dz}{2\pi jz} \tag{13.9.10}$$

where in the first term we used $S_{x_1x_1}(z) = zS_{xx}(z)z^{-1} = S_{xx}(z)$, and in the second term, $S_{yx_1}(z) = S_{yx}(z)z^{-1}$.

Next, we show that Eqs. (13.9.9) and (13.9.10) agree with the results obtained from the steady-state Kalman filter. In particular, we expect the contour integral in Eq. (13.9.10) to be equal to the steady-state solution P of the DARE. We recall from Sec. 13.6 that the steady-state Kalman filter parameters are, where $D = CPC^T + R$:

$$K = APC^TD^{-1} = FPC^TR^{-1}\,, \quad F = A - KC = A - APC^TD^{-1}C \tag{13.9.11}$$

where P is the unique positive-semidefinite symmetric solution of the DARE:

$$P = APA^T - APC^T(CPC^T + R)^{-1}CPA^T + Q \tag{13.9.12}$$

which can also be written as

$$Q = P - FPA^T \tag{13.9.13}$$

The state-space model for the Kalman filter can be written in the z-domain as follows:

$$\begin{aligned} \mathbf{x}_{n+1} &= A\mathbf{x}_n + \mathbf{w}_n \\ \mathbf{y}_n &= C\mathbf{x}_n + \mathbf{v}_n \end{aligned} \quad \Rightarrow \quad \begin{aligned} \mathbf{X}(z) &= (zI - A)^{-1}\mathbf{W}(z) \\ \mathbf{Y}(z) &= C(zI - A)^{-1}\mathbf{W}(z) + \mathbf{V}(z) \end{aligned}$$

from which we obtain the spectral densities:

$$\begin{aligned} S_{xx}(z) &= (zI - A)^{-1}Q(z^{-1}I - A^T)^{-1} \\ S_{xy}(z) &= S_{xx}(z)C^T = (zI - A)^{-1}Q(z^{-1}I - A^T)^{-1}C^T \\ S_{yy}(z) &= CS_{xx}(z)C^T + R = C(zI - A)^{-1}Q(z^{-1}I - A^T)^{-1}C^T + R \end{aligned} \tag{13.9.14}$$

The steady-state Kalman prediction filter is given by

$$\hat{\mathbf{x}}_{n+1/n} = A\hat{\mathbf{x}}_{n/n-1} + K\boldsymbol{\varepsilon}_n = A\hat{\mathbf{x}}_{n/n-1} + K(\mathbf{y}_n - C\hat{\mathbf{x}}_{n/n-1}) = F\hat{\mathbf{x}}_{n/n-1} + K\mathbf{y}_n$$

which may be rewritten in terms of $\hat{\mathbf{x}}_1(n) = \hat{\mathbf{x}}_{n+1/n}$, or, $\hat{\mathbf{x}}_1(n-1) = \hat{\mathbf{x}}_{n/n-1}$,

$$\hat{\mathbf{x}}_1(n) = A\hat{\mathbf{x}}_1(n-1) + K\boldsymbol{\varepsilon}_n = F\hat{\mathbf{x}}_1(n-1) + K\mathbf{y}_n \tag{13.9.15}$$

and in the z-domain, noting that $\hat{\mathbf{y}}_{n/n-1} = C\hat{\mathbf{x}}_{n/n-1} = C\hat{\mathbf{x}}_1(n-1)$,

$$\begin{aligned}
\hat{\mathbf{X}}_1(z) &= (I - z^{-1}A)^{-1}K\boldsymbol{\mathcal{E}}(z) = (I - z^{-1}F)^{-1}K\mathbf{Y}(z) \\
\hat{\mathbf{Y}}(z) &= z^{-1}C\hat{\mathbf{X}}_1(z) = C(zI - A)^{-1}K\boldsymbol{\mathcal{E}}(z) = C(zI - F)^{-1}K\mathbf{Y}(z)
\end{aligned} \tag{13.9.16}$$

From the first of these, we obtain the prediction filter transfer function $H_1(z)$ relating the observations $\mathbf{y}_n$ to the prediction $\hat{\mathbf{x}}_1(n) = \hat{\mathbf{x}}_{n+1/n}$, i.e., $\hat{\mathbf{X}}_1(z) = H_1(z)\mathbf{Y}(z)$:

$$H_1(z) = (I - z^{-1}F)^{-1}K \tag{13.9.17}$$

Since $\boldsymbol{\varepsilon}_n = \mathbf{y}_n - \hat{\mathbf{y}}_{n/n-1}$, or, $\boldsymbol{\mathcal{E}}(z) = \mathbf{Y}(z) - \hat{\mathbf{Y}}(z)$, the second of Eqs. (13.9.16) allows us to determine the signal model transfer function matrix $B(z)$, i.e., $\mathbf{Y}(z) = B(z)\boldsymbol{\mathcal{E}}(z)$:

$$\begin{aligned}
\mathbf{Y}(z) &= \boldsymbol{\mathcal{E}}(z) + \hat{\mathbf{Y}}(z) = \big[I + C(zI - A)^{-1}K\big]\boldsymbol{\mathcal{E}}(z) \\
\boldsymbol{\mathcal{E}}(z) &= \mathbf{Y}(z) - \hat{\mathbf{Y}}(z) = \big[I - C(zI - F)^{-1}K\big]\mathbf{Y}(z)
\end{aligned}$$

from which we obtain $B(z)$ and its inverse $B^{-1}(z)$:

$$\begin{aligned}
B(z) &= I + C(zI - A)^{-1}K \\
B^{-1}(z) &= I - C(zI - F)^{-1}K
\end{aligned} \tag{13.9.18}$$

It can easily be verified that $[I + C(zI - A)^{-1}K][I - C(zI - F)^{-1}K] = I$ by direct multiplication, using the fact that $F = A - KC$. Next, we must verify the spectral factorization of $S_{yy}(z)$ and show that Eq. (13.9.17) agrees with (13.9.9). We will make use of the following relationships:

$$\begin{aligned}
&(zI - F)^{-1}KB(z) = (zI - A)^{-1}K \\
&B(z)C(zI - F)^{-1} = C(zI - A)^{-1} \\
&(z^{-1}I - F^T)^{-1}C^TB^T(z^{-1}) = (z^{-1}I - A^T)^{-1}C^T
\end{aligned} \tag{13.9.19}$$

where the third is obtained by transposing the second and replacing z by z^{-1}. These can be shown in a straightforward way, for example,

$$\begin{aligned}
B(z)C(zI - F)^{-1} &= [I + C(zI - A)^{-1}K]C(zI - F)^{-1} \\
&= [C + C(zI - A)^{-1}KC](zI - F)^{-1} \\
&= C(zI - A)^{-1}(zI - A + KC)(zI - F)^{-1} \\
&= C(zI - A)^{-1}(zI - F)(zI - F)^{-1} = C(zI - A)^{-1}
\end{aligned}$$

We will also need the following relationship and its transposed/reflected version:

$$\begin{aligned}
(zI - F)^{-1}Q(z^{-1}I - A^T)^{-1} &= (I - z^{-1}F)^{-1}P + PA^T(z^{-1}I - A^T)^{-1} \\
(zI - A)^{-1}Q(z^{-1}I - F^T)^{-1} &= P(I - zF^T)^{-1} + (zI - A)^{-1}AP
\end{aligned} \tag{13.9.20}$$

These are a consequence of the DARE written in the form of Eq. (13.9.13). Indeed,

$$\begin{aligned}
&(I - z^{-1}F)^{-1}P + PA^T(z^{-1}I - A^T)^{-1} = \\
&\quad = (I - z^{-1}F)^{-1}\big[P(z^{-1}I - A^T) + (I - z^{-1}F)PA^T\big](z^{-1}I - A^T)^{-1} \\
&\quad = (I - z^{-1}F)^{-1}\big[z^{-1}P - PA^T + PA^T - z^{-1}FPA^T\big](z^{-1}I - A^T)^{-1} \\
&\quad = (I - z^{-1}F)^{-1}z^{-1}(P - FPA^T)(z^{-1}I - A^T)^{-1} = (zI - F)^{-1}Q(z^{-1}I - A^T)^{-1}
\end{aligned}$$

Next, we verify the spectral factorization (13.9.6). Using (13.9.19) we have:

$$\begin{aligned}
S_{yy}(z) &= C(zI - A)^{-1}Q(z^{-1}I - A^T)^{-1}C^T + R \\
&= B(z)C(zI - F)^{-1}Q(z^{-1}I - A^T)^{-1} + R
\end{aligned}$$

Multiplying from the left by $B^{-1}(z)$ and using (13.9.18) and (13.9.20), we obtain:

$$\begin{aligned}
B^{-1}(z)S_{yy}(z) &= C(zI - F)^{-1}Q(z^{-1}I - A^T)^{-1} + B^{-1}(z)R \\
&= C\big[(I - z^{-1}F)^{-1}P + PA^T(z^{-1}I - A^T)^{-1}\big]C^T + \big[I - C(z - F)^{-1}K\big]R \\
&= C\big[(I - z^{-1}F)^{-1}P + PA^T(z^{-1}I - A^T)^{-1}\big]C^T + R - Cz^{-1}(I - z^{-1}F)^{-1}KR \\
&= C\big[(I - z^{-1}F)^{-1}P + PA^T(z^{-1}I - A^T)^{-1}\big]C^T + R - Cz^{-1}(I - z^{-1}F)^{-1}FPC^T \\
&= C(I - z^{-1}F)^{-1}(I - z^{-1}F)PC^T + CPA^T(z^{-1}I - A^T)^{-1}C^T + R \\
&= CPC^T + CPA^T(z^{-1}I - A^T)^{-1}C^T + R \\
&= CPA^T(z^{-1}I - A^T)^{-1}C^T + D = DK^T(z^{-1}I - A^T)^{-1}C^T + D \\
&= D\big[I + K^T(z^{-1}I - A^T)^{-1}C^T\big] = DB^T(z^{-1})
\end{aligned}$$

where we replaced $KR = FPC^T$ and $CPA^T = DK^T$ from Eq. (13.9.11). This verifies Eq. (13.9.6). Next, we obtain the prediction filter using the Wiener filter solution (13.9.9). Using (13.9.19), we have:

$$S_{xy}(z) = (zI - A)^{-1}Q(z^{-1}I - A^T)^{-1}C^T = (zI - A)^{-1}Q(z^{-1}I - F^T)^{-1}C^TB^T(z^{-1})$$

Multiplying by the inverse of $B^T(z)$ from the right and using (13.9.20), we obtain:

$$\begin{aligned}
zS_{xy}(z)B^{-T}(z^{-1}) &= z(zI - A)^{-1}Q(z^{-1}I - F^T)^{-1}C^T \\
&= zP(I - zF^T)^{-1}C^T + z(zI - A)^{-1}APC^T \\
&= zP(I - zF^T)^{-1}C^T + (I - z^{-1}A)^{-1}KD
\end{aligned}$$

where we replaced $APC^T = KD$. The first term is anti-causal (if it is to have a stable inverse z-transform), while the second term is causal (assuming here that A is strictly stable). Thus, we find the causal part:

$$\Big[zS_{xy}(z)B^{-T}(z^{-1})\Big]_+ = (I - z^{-1}A)^{-1}KD = (I - z^{-1}F)^{-1}KB(z)D \tag{13.9.21}$$

where we used the first of Eqs. (13.9.19). It follows that the Wiener prediction filter is:

$$H_1(z) = \Big[zS_{xy}(z)B^{-T}(z^{-1})\Big]_+ D^{-1}B^{-1}(z) = (I - z^{-1}F)^{-1}K \tag{13.9.22}$$

and agrees with (13.9.17). Finally, we consider the prediction error covariance given by (13.9.10). Noting that $S_{yx}(z) = CS_{xx}(z)$, the integrand of (13.9.10) becomes:

$$\begin{aligned}
S_{xx}(z) - H_1(z)S_{yx}(z)z^{-1} &= [I - z^{-1}H_1(z)C]S_{xx}(z) = [I - (zI - F)^{-1}KC]S_{xx}(z)\\
&= (zI - F)^{-1}(zI - F - KC)S_{xx}(z) = (zI - F)^{-1}(zI - A)S_{xx}(z)\\
&= (zI - F)^{-1}(zI - A)(zI - A)^{-1}Q(z^{-1}I - A^T)^{-1} = (zI - F)^{-1}Q(z^{-1}I - A^T)^{-1}\\
&= (I - z^{-1}F)^{-1}P + PA^T(z^{-1}I - A^T)^{-1} = z(zI - F)^{-1}P + zPA^T(I - zA^T)^{-1}
\end{aligned}$$

and the contour integral (13.9.10) becomes:

$$\oint_{\text{u.c.}}\Big[S_{xx}(z) - H_1(z)S_{yx}(z)z^{-1}\Big]\frac{dz}{2\pi jz} = \oint_{\text{u.c.}}\Big[(zI - F)^{-1}P + PA^T(I - zA^T)^{-1}\Big]\frac{dz}{2\pi j}$$

The poles of the second term lie outside the unit circle and do not contribute to the integral. The poles of the first term are the eigenvalues of the matrix F, which all lie inside the unit circle. It is not hard to see (e.g., using the eigenvalue decomposition of F) that the first term integrates into:

$$\oint_{\text{u.c.}}\Big[(zI - F)^{-1}P\Big]\frac{dz}{2\pi j} = P$$

Thus, as expected the Wiener and Kalman expressions for $E[\mathbf{e}_{n+1/n}\mathbf{e}_{n+1/n}^T]$ agree with each other.

13.10 Fixed-Interval Smoothing

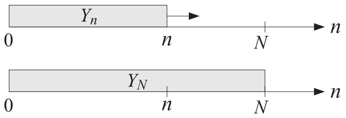
The Kalman filtering algorithm proceeds recursively in time using an ever increasing observations subspace:

$$Y_n = \{\mathbf{y}_0, \mathbf{y}_1, \dots, \mathbf{y}_n\} = \{\boldsymbol{\varepsilon}_0, \boldsymbol{\varepsilon}_1, \dots, \boldsymbol{\varepsilon}_n\}\,, \quad n = 0, 1, 2, \dots$$

with the current estimate $\hat{\mathbf{x}}_{n/n}$ based on Y_n. In the fixed-interval Kalman smoothing problem, the observations $\mathbf{y}_n$ are available over a fixed time interval $0 \le n \le N$, so that the observation subspace is:

$$Y_N = \{\mathbf{y}_0, \mathbf{y}_1, \dots, \mathbf{y}_N\} = \{\boldsymbol{\varepsilon}_0, \boldsymbol{\varepsilon}_1, \dots, \boldsymbol{\varepsilon}_N\}$$

and the estimate of $\mathbf{x}_n$, for $0 \le n \le N$, is based on the entire subspace Y_N. The two cases are depicted below:



At each n, the subspace Y_N can be decomposed in the direct sums:

$$\begin{aligned}
Y_N &= \{\boldsymbol{\varepsilon}_0, \boldsymbol{\varepsilon}_1, \dots, \boldsymbol{\varepsilon}_n\} \oplus \{\boldsymbol{\varepsilon}_{n+1}, \dots, \boldsymbol{\varepsilon}_N\} = Y_n \oplus \{\boldsymbol{\varepsilon}_{n+1}, \dots, \boldsymbol{\varepsilon}_N\}\\
Y_N &= \{\boldsymbol{\varepsilon}_0, \boldsymbol{\varepsilon}_1, \dots, \boldsymbol{\varepsilon}_{n-1}\} \oplus \{\boldsymbol{\varepsilon}_n, \dots, \boldsymbol{\varepsilon}_N\} = Y_{n-1} \oplus \{\boldsymbol{\varepsilon}_n, \dots, \boldsymbol{\varepsilon}_N\}
\end{aligned} \tag{13.10.1}$$

and therefore, we expect the estimate of $\mathbf{x}_n$ based on Y_N to be equal to the sum of the filtered estimate $\hat{\mathbf{x}}_{n/n}$, or the predicted estimate $\hat{\mathbf{x}}_{n/n-1}$, plus a correction coming from the rest of the subspace. We will work with the latter decomposition. We will find that once the ordinary Kalman filter has been run forward from $n = 0$ to $n = N$, and we have constructed the innovations basis for Y_N and the estimates $\hat{\mathbf{x}}_{n/n-1}$, the required correction can be constructed recursively, but running backwards from N down to n.

We begin by noting that at each n within $0 \le n \le N$, the state vector $\mathbf{x}_n$ can be written in its unique orthogonal decomposition relative to the subspace Y_{n-1}:

$$\mathbf{x}_n = \hat{\mathbf{x}}_{n/n-1} + \mathbf{e}_{n/n-1} \tag{13.10.2}$$

where $\hat{\mathbf{x}}_{n/n-1}$ is the ordinary predicted estimate of $\mathbf{x}_n$ as defined in the forward Kalman algorithm (13.2.2), and $\mathbf{e}_{n/n-1}$ is the prediction error whose covariance matrix is $P_{n/n-1}$. We observe that $\hat{\mathbf{x}}_{n/n-1}$ can be expressed in terms of the innovations basis of Y_{n-1}, as in Eq. (13.3.16):

$$\hat{\mathbf{x}}_{n/n-1} = \bar{\mathbf{x}}_n + \sum_{m=0}^{n-1} E[\mathbf{x}_n\boldsymbol{\varepsilon}_m^T]D_m^{-1}\boldsymbol{\varepsilon}_m \tag{13.10.3}$$

where $D_m = E[\boldsymbol{\varepsilon}_m\boldsymbol{\varepsilon}_m^T]$. The smoothed estimate of $\mathbf{x}_n$ based on the full subspace Y_N is the projection of $\mathbf{x}_n$ onto Y_N. Denoting this estimate by $\hat{\mathbf{x}}_{n/N}$, it is given in the innovations basis for $Y_N = \{\boldsymbol{\varepsilon}_0, \boldsymbol{\varepsilon}_1, \dots, \boldsymbol{\varepsilon}_N\}$:

$$\hat{\mathbf{x}}_{n/N} = \text{Proj}[\mathbf{x}_n|Y_N] = \bar{\mathbf{x}}_n + \sum_{m=0}^{N} E[\mathbf{x}_n\boldsymbol{\varepsilon}_m^T]D_m^{-1}\boldsymbol{\varepsilon}_m \tag{13.10.4}$$

The summation may be split into two terms:

$$\begin{aligned}
\hat{\mathbf{x}}_{n/N} &= \bar{\mathbf{x}}_n + \sum_{m=0}^{n-1} E[\mathbf{x}_n\boldsymbol{\varepsilon}_m^T]D_m^{-1}\boldsymbol{\varepsilon}_m + \sum_{m=n}^{N} E[\mathbf{x}_n\boldsymbol{\varepsilon}_m^T]D_m^{-1}\boldsymbol{\varepsilon}_m\\
&= \hat{\mathbf{x}}_{n/n-1} + \sum_{m=n}^{N} E[\mathbf{x}_n\boldsymbol{\varepsilon}_m^T]D_m^{-1}\boldsymbol{\varepsilon}_m
\end{aligned}$$

The second term is recognized as the estimate of $\mathbf{e}_{n/n-1}$ based on Y_N, that is,

$$\begin{aligned}
\hat{\mathbf{e}}_{n/n-1} &= \sum_{m=0}^{N} E[\mathbf{e}_{n/n-1}\boldsymbol{\varepsilon}_m^T]D_m^{-1}\boldsymbol{\varepsilon}_m \\
&= \sum_{m=0}^{n-1} E[\mathbf{e}_{n/n-1}\boldsymbol{\varepsilon}_m^T]D_m^{-1}\boldsymbol{\varepsilon}_m + \sum_{m=n}^{N} E[\mathbf{e}_{n/n-1}\boldsymbol{\varepsilon}_m^T]D_m^{-1}\boldsymbol{\varepsilon}_m \\
&= \sum_{m=n}^{N} E[(\mathbf{x}_n - \hat{\mathbf{x}}_{n/n-1})\boldsymbol{\varepsilon}_m^T]D_m^{-1}\boldsymbol{\varepsilon}_m = \sum_{m=n}^{N} E[\mathbf{x}_n\boldsymbol{\varepsilon}_m^T]D_m^{-1}\boldsymbol{\varepsilon}_m
\end{aligned}$$

where we dropped the terms $E[\mathbf{e}_{n/n-1}\boldsymbol{\varepsilon}_m^T]= 0$ for $0 \le m \le n-1$, because of the orthogonality conditions for the estimate $\hat{\mathbf{x}}_{n/n-1}$ (i.e., the estimation error must be orthogonal to the observations that make up the estimate), and then we dropped $E[\hat{\mathbf{x}}_{n/n-1}\boldsymbol{\varepsilon}_m^T]= 0$ for $n \le m \le N$ because these $\boldsymbol{\varepsilon}_m$s are orthogonal to the $\boldsymbol{\varepsilon}_m$s making up $\hat{\mathbf{x}}_{n/n-1}$, as seen from the direct sum (13.10.1). Thus, we have:

$$\hat{\mathbf{x}}_{n/N} = \hat{\mathbf{x}}_{n/n-1} + \hat{\mathbf{e}}_{n/n-1} \tag{13.10.5}$$

$$\hat{\mathbf{e}}_{n/n-1} = \sum_{m=n}^{N} E[\mathbf{e}_{n/n-1}\boldsymbol{\varepsilon}_m^T]D_m^{-1}\boldsymbol{\varepsilon}_m \tag{13.10.6}$$

In other words, the term $\hat{\mathbf{e}}_{n/n-1}$ is the correction to the predicted estimate $\hat{\mathbf{x}}_{n/n-1}$ and represents the estimate of the prediction error $\mathbf{e}_{n/n-1}$ based on the subspace $\{\boldsymbol{\varepsilon}_n, \dots, \boldsymbol{\varepsilon}_N\}$ that lies in the future of $\hat{\mathbf{x}}_{n/n-1}$. The same result can be obtained by taking the projections of both sides of Eq. (13.10.2) onto Y_N and noting that the projection of $\hat{\mathbf{x}}_{n/n-1}$ is itself because Y_{n-1} is a subspace of Y_N. The estimation error for the smoothed estimate is equal to the estimation error for $\mathbf{e}_{n/n-1}$, indeed,

$$\mathbf{e}_{n/N} = \mathbf{x}_n - \hat{\mathbf{x}}_{n/N} = \mathbf{x}_n - \hat{\mathbf{x}}_{n/n-1} - \hat{\mathbf{e}}_{n/n-1}\,, \quad \text{or,}$$

$$\mathbf{e}_{n/N} = \mathbf{e}_{n/n-1} - \hat{\mathbf{e}}_{n/n-1} \tag{13.10.7}$$

The error covariance matrices can be obtained by writing $\mathbf{e}_{n/n-1} = \mathbf{e}_{n/N} + \hat{\mathbf{e}}_{n/n-1}$ and noting that the two terms on the right-hand-side are orthogonal because $\hat{\mathbf{e}}_{n/n-1}$ is composed of observations that appear in the $\hat{\mathbf{x}}_{n/N}$ and therefore, they must be orthogonal to the corresponding estimation error $\mathbf{e}_{n/N}$. Let,

$$P_{n/N} = E[\mathbf{e}_{n/N}\mathbf{e}_{n/N}^T]\,, \quad \hat{P}_{n/n-1} = E[\hat{\mathbf{e}}_{n/n-1}\hat{\mathbf{e}}_{n/n-1}^T] \tag{13.10.8}$$

then, the above orthogonality property implies:

$$E[\mathbf{e}_{n/n-1}\mathbf{e}_{n/n-1}^T]= E[\mathbf{e}_{n/N}\mathbf{e}_{n/N}^T]+E[\hat{\mathbf{e}}_{n/n-1}\hat{\mathbf{e}}_{n/n-1}^T]\,, \quad \text{or,}$$

$$P_{n/N} = P_{n/n-1} - \hat{P}_{n/n-1} \tag{13.10.9}$$

The term $\hat{P}_{n/n-1}$ quantifies the improvement in the estimate of $\mathbf{x}_n$ afforded by using all the data Y_N instead of only Y_{n-1}.

Next, we develop the backward recursions satisfied by $\hat{\mathbf{e}}_{n/n-1}$ and $\hat{P}_{n/n-1}$, which will allow the calculation of $\hat{\mathbf{x}}_{n/N}$ and $P_{n/N}$. We recall that $\boldsymbol{\varepsilon}_m = \mathbf{y}_m - C_m\hat{\mathbf{x}}_{m/m-1} = C_m\mathbf{x}_m + \mathbf{v}_m - C_m\hat{\mathbf{x}}_{m/m-1} = C_m\mathbf{e}_{m/m-1} + \mathbf{v}_m$. This implies $E[\mathbf{e}_{n/n-1}\boldsymbol{\varepsilon}_m^T]= E[\mathbf{e}_{n/n-1}\mathbf{e}_{m/m-1}^T]C_m^T$. And it is straightforward to show that for $m \ge n$:

$$E[\mathbf{e}_{n/n-1}\mathbf{e}_{m/m-1}^T]= P_{n/n-1}\Psi_{n,m}\,, \qquad \Psi_{n,m} = \begin{cases} F_n^T F_{n+1}^T \cdots F_{m-1}^T\,, & m > n \\ I\,, & m = n \end{cases} \tag{13.10.10}$$

where $F_n = A_n - K_nC_n$ is the closed-loop transition matrix. For example, consider the case $m = n+1$. Then, $\mathbf{e}_{n+1/n} = \mathbf{x}_{n+1} - \hat{\mathbf{x}}_{n+1/n} = A_n\mathbf{x}_n + \mathbf{w}_n - A_n\hat{\mathbf{x}}_{n/n-1} - K_n\boldsymbol{\varepsilon}_n$, or,

$$\begin{aligned}
\mathbf{e}_{n+1/n} &= A_n\mathbf{e}_{n/n-1} + \mathbf{w}_n - K_n\boldsymbol{\varepsilon}_n = A_n\mathbf{e}_{n/n-1} + \mathbf{w}_n - K_n(C_n\mathbf{e}_{n/n-1} + \mathbf{v}_n) \\
&= (A_n - K_nC_n)\mathbf{e}_{n/n-1} + \mathbf{w}_n - K_n\mathbf{v}_n = F_n\mathbf{e}_{n/n-1} + \mathbf{w}_n - K_n\mathbf{v}_n
\end{aligned}$$

and because $\mathbf{e}_{n/n-1}$ depends on $\{\mathbf{x}_0, \mathbf{w}_0, \dots, \mathbf{w}_{n-1}, \mathbf{v}_0, \dots, \mathbf{v}_{n-1}\}$, it will be orthogonal to $\mathbf{w}_n, \mathbf{v}_n$, and we find:

$$E[\mathbf{e}_{n/n-1}\mathbf{e}_{n+1/n}^T]= E[\mathbf{e}_{n/n-1}\mathbf{e}_{n/n-1}^T]F_n^T = P_{n/n-1}F_n^T \tag{13.10.11}$$

For $m = n+2$, we have similarly, $\mathbf{e}_{n+2/n+1} = F_{n+1}\mathbf{e}_{n+1/n} + \mathbf{w}_{n+1} - K_{n+1}\mathbf{v}_{n+1}$, and,

$$E[\mathbf{e}_{n/n-1}\mathbf{e}_{n+2/n+1}^T]= E[\mathbf{e}_{n/n-1}\mathbf{e}_{n+1/n}^T]F_{n+1}^T = P_{n/n-1}F_n^TF_{n+1}^T$$

and so on for $m > n$. Thus, we can write $\hat{\mathbf{e}}_{n/n-1}$ in the form:

$$\hat{\mathbf{e}}_{n/n-1} = P_{n/n-1}\sum_{m=n}^{N}\Psi_{n,m}C_m^TD_m^{-1}\boldsymbol{\varepsilon}_m \tag{13.10.12}$$

Separating out the first term and recalling the Kalman gain $G_n = P_{n/n-1}C_n^TD_n^{-1}$, we have:

$$\hat{\mathbf{e}}_{n/n-1} = G_n\boldsymbol{\varepsilon}_n + P_{n/n-1}\sum_{m=n+1}^{N}\Psi_{n,m}C_m^TD_m^{-1}\boldsymbol{\varepsilon}_m \tag{13.10.13}$$

On the other hand, we have:

$$\hat{\mathbf{e}}_{n+1/n} = P_{n+1/n}\sum_{m=n+1}^{N}\Psi_{n+1,m}C_m^TD_m^{-1}\boldsymbol{\varepsilon}_m \quad\Rightarrow\quad \sum_{m=n+1}^{N}\Psi_{n+1,m}C_m^TD_m^{-1}\boldsymbol{\varepsilon}_m = P_{n+1/n}^{-1}\hat{\mathbf{e}}_{n+1/n}$$

Noting that $\Psi_{n,m} = F_n^T\Psi_{n+1,m}$, for $m \ge n+1$, we obtain:

$$\sum_{m=n+1}^{N}\Psi_{n,m}C_m^TD_m^{-1}\boldsymbol{\varepsilon}_m = F_n^T\sum_{m=n+1}^{N}\Psi_{n+1,m}C_m^TD_m^{-1}\boldsymbol{\varepsilon}_m = F_n^TP_{n+1/n}^{-1}\hat{\mathbf{e}}_{n+1/n}$$

and using this into Eq. (13.10.13), we find:

$$\hat{\mathbf{e}}_{n/n-1} = G_n\boldsymbol{\varepsilon}_n + P_{n/n-1}F_n^TP_{n+1/n}^{-1}\hat{\mathbf{e}}_{n+1/n} \tag{13.10.14}$$

Thus, the required backward recursion for $\hat{\mathbf{e}}_{n/n-1}$ may be written as:

$$\boxed{\begin{aligned} L_n &= P_{n/n-1}F_n^TP_{n+1/n}^{-1} \\ \hat{\mathbf{e}}_{n/n-1} &= G_n\boldsymbol{\varepsilon}_n + L_n\hat{\mathbf{e}}_{n+1/n} \end{aligned}} \tag{13.10.15}$$

for $n = N, N-1, \dots, 0$. At $n = N$, Eq. (13.10.12) gives:

$$\hat{\mathbf{e}}_{N/N-1} = P_{N/N-1}\sum_{m=N}^{N}\Psi_{N,m}C_m^T D_m^{-1}\boldsymbol{\varepsilon}_m = P_{N/N-1}\Psi_{N,N}C_N^T D_N^{-1}\boldsymbol{\varepsilon}_N = G_N\boldsymbol{\varepsilon}_N$$

Therefore, the initialization of the recursion (13.10.15) at $n = N$ is:

$$\hat{\mathbf{e}}_{N+1/N} = 0$$

The covariance $\hat{P}_{n/n-1}$ satisfies a similar recursion. Since $\boldsymbol{\varepsilon}_n$ is orthogonal to all the terms of $\hat{\mathbf{e}}_{n+1/n}$, which depend on $\boldsymbol{\varepsilon}_m$, $m \geq n+1$, it follows by taking covariances of both sides of (13.10.15) that:

$$\hat{P}_{n/n-1} = G_n D_n G_n^T + L_n\hat{P}_{n+1/n}L_n^T\,, \quad n = N, N-1, \dots, 0 \tag{13.10.16}$$

and initialized with $\hat{P}_{N+1/N} = 0$.

To summarize, the smoothed estimate $\hat{\mathbf{x}}_{n/N}$ is computed by first running the ordinary Kalman filtering algorithm (13.2.2) forward in time for $n = 0, 1, \dots, N$, saving the quantities $\hat{\mathbf{x}}_{n/n-1}, \boldsymbol{\varepsilon}_n$, along with $P_{n/n-1}, G_n, D_n, F_n = A_n - K_n C_n$, and then, carrying out the following backward recursions from $n = N$ down to $n = 0$,

```
Initialize: $\hat{\mathbf{e}}_{N+1/N} = 0$, $\hat{P}_{N+1/N} = 0$
for $n = N, N-1, \dots, 0$, do:
    $L_n = P_{n/n-1}F_n^T P_{n+1/n}^{-1}$
    $\hat{\mathbf{e}}_{n/n-1} = G_n\boldsymbol{\varepsilon}_n + L_n\hat{\mathbf{e}}_{n+1/n}$
    $\hat{P}_{n/n-1} = G_n D_n G_n^T + L_n\hat{P}_{n+1/n}L_n^T$
    $\hat{\mathbf{x}}_{n/N} = \hat{\mathbf{x}}_{n/n-1} + \hat{\mathbf{e}}_{n/n-1}$
    $P_{n/N} = P_{n/n-1} - \hat{P}_{n/n-1}$
```
(13.10.17)

We note also that L_n may be written in the form:

$$L_n = P_{n/n}A_n^T P_{n+1/n}^{-1} \tag{13.10.18}$$

Indeed,

$$\begin{aligned}
L_n = P_{n/n-1}F_n^T P_{n+1/n}^{-1} &= P_{n/n-1}(A_n - K_n C_n)^T P_{n+1/n}^{-1} \\
&= P_{n/n-1}\big(A_n^T - C_n^T D_n^{-1}C_n P_{n/n-1}A_n^T\big)P_{n+1/n}^{-1} \\
&= \big(P_{n/n-1} - P_{n/n-1}C_n^T D_n^{-1}C_n P_{n/n-1}\big)A_n^T P_{n+1/n}^{-1} = P_{n/n}A_n^T P_{n+1/n}^{-1}
\end{aligned}$$

There exist a number of alternative re-formulations of the smoothing problem that can be derived from algorithm (13.10.17). The so-called Rauch-Tung-Striebel (RTS) version [883,884] is obtained by eliminating the variable $\hat{\mathbf{e}}_{n/n-1}$ in favor of $\hat{\mathbf{x}}_{n/N}$. Applying the equations for the estimate and estimation error at time $n+1$, we have:

$$\begin{aligned}
\hat{\mathbf{x}}_{n+1/N} &= \hat{\mathbf{x}}_{n+1/n} + \hat{\mathbf{e}}_{n+1/n} \\
P_{n+1/N} &= P_{n+1/n} - \hat{P}_{n+1/n}
\end{aligned}
\quad\Rightarrow\quad
\begin{aligned}
\hat{\mathbf{e}}_{n+1/n} &= \hat{\mathbf{x}}_{n+1/N} - \hat{\mathbf{x}}_{n+1/n} \\
\hat{P}_{n+1/n} &= P_{n+1/n} - P_{n+1/N}
\end{aligned}$$

and substituting these into the recursions in (13.10.17), we obtain:

$$\begin{aligned}
\hat{\mathbf{e}}_{n/n-1} &= G_n\boldsymbol{\varepsilon}_n + L_n(\hat{\mathbf{x}}_{n+1/N} - \hat{\mathbf{x}}_{n+1/n}) \\
\hat{P}_{n/n-1} &= G_n D_n G_n^T + L_n(P_{n+1/n} - P_{n+1/N})L_n^T \\
\hat{\mathbf{x}}_{n/N} &= \hat{\mathbf{x}}_{n/n-1} + \hat{\mathbf{e}}_{n/n-1} = \hat{\mathbf{x}}_{n/n-1} + G_n\boldsymbol{\varepsilon}_n + L_n(\hat{\mathbf{x}}_{n+1/N} - \hat{\mathbf{x}}_{n+1/n}) \\
P_{n/N} &= P_{n/n-1} - \hat{P}_{n/n-1} = P_{n/n-1} - G_n D_n G_n^T + L_n(P_{n+1/N} - P_{n+1/n})L_n^T
\end{aligned}$$

but from the ordinary Kalman filter, we have the filtered estimate and its covariance:

$$\begin{aligned}
\hat{\mathbf{x}}_{n/n} &= \hat{\mathbf{x}}_{n/n-1} + G_n\boldsymbol{\varepsilon}_n \\
P_{n/n} &= P_{n/n-1} - G_n D_n G_n^T
\end{aligned}$$

Hence, the above smoothed estimates can be written in the RTS form:

$$\boxed{\begin{aligned}
L_n &= P_{n/n}A_n^T P_{n+1/n}^{-1} \\
\hat{\mathbf{x}}_{n/N} &= \hat{\mathbf{x}}_{n/n} + L_n(\hat{\mathbf{x}}_{n+1/N} - \hat{\mathbf{x}}_{n+1/n}) \\
P_{n/N} &= P_{n/n} + L_n(P_{n+1/N} - P_{n+1/n})L_n^T
\end{aligned}} \quad \text{(RTS smoothing)} \tag{13.10.19}$$

This is to be iterated from $n = N$ down to $n = 0$, where the differences in the second terms are initialized to zero, e.g., at $n = N$, we have $L_N(\hat{\mathbf{x}}_{N+1/N} - \hat{\mathbf{x}}_{N+1/N}) = 0$.

A disadvantage of the algorithm (13.10.17) and of the RTS form is that the computation of L_n requires an additional matrix inversion of the quantity $P_{n+1/n}$. Such inversion is avoided in the so-called Bryson-Frazier (BF) smoothing formulation [881,882]. To derive it, we use Eq. (13.10.12) to define the quantity:

$$\mathbf{g}_n = P_{n/n-1}^{-1}\hat{\mathbf{e}}_{n/n-1} = \sum_{m=n}^{N}\Psi_{n,m}C_m^T D_m^{-1}\boldsymbol{\varepsilon}_m \tag{13.10.20}$$

It follows from Eq. (13.10.14) and $G_n = P_{n/n-1}C_n^T D_n^{-1}$, and $\mathbf{g}_{n+1} = P_{n+1/n}^{-1}\hat{\mathbf{e}}_{n+1/n}$, that

$$\mathbf{g}_n = P_{n/n-1}^{-1}\hat{\mathbf{e}}_{n/n-1} = P_{n/n-1}^{-1}\big(G_n\boldsymbol{\varepsilon}_n + P_{n/n-1}F_n^T P_{n+1/n}^{-1}\hat{\mathbf{e}}_{n+1/n}\big)\,, \quad \text{or,}$$

$$\mathbf{g}_n = C_n^T D_n^{-1}\boldsymbol{\varepsilon}_n + F_n^T\mathbf{g}_{n+1} \tag{13.10.21}$$

with initial value $\mathbf{g}_{N+1} = 0$, which follows from $\hat{\mathbf{e}}_{n+1/N} = 0$. From Eq. (13.10.20) we note that the two terms $\boldsymbol{\varepsilon}_n$ and $\mathbf{g}_{n+1}$ in the right-hand-side are orthogonal, and therefore, we obtain the following recursion for the covariance $\Gamma_n = E[\mathbf{g}_n\mathbf{g}_n^T]$:

$$\Gamma_n = C_n^T D_n^{-1}C_n + F_n^T\Gamma_{n+1}F_n \tag{13.10.22}$$

where we used $C_n^T D_n^{-1}E[\boldsymbol{\varepsilon}_n\boldsymbol{\varepsilon}_n^T]D_n^{-1}C_n = C_n^T D_n^{-1}D_n D_n^{-1}C_n = C_n^T D_n^{-1}C_n$. The recursion is to be initialized at $\Gamma_{N+1} = 0$. Noting that,

$$\hat{P}_{n/n-1} = E[\hat{\mathbf{e}}_{n/n-1}\hat{\mathbf{e}}_{n/n-1}^T] = P_{n/n-1}E[\mathbf{g}_n\mathbf{g}_n^T]P_{n/n-1} = P_{n/n-1}\Gamma_n P_{n/n-1}$$

we obtain the Bryson-Frasier smoothing algorithm:

$$\boxed{\begin{aligned}
&\text{Initialize: } \mathbf{g}_{N+1} = 0,\ \Gamma_{N+1} = 0\\
&\text{for } n = N, N-1, \dots, 0, \text{ do:}\\
&\qquad \mathbf{g}_n = C_n^T D_n^{-1}\boldsymbol{\varepsilon}_n + F_n^T \mathbf{g}_{n+1}\\
&\qquad \Gamma_n = C_n^T D_n^{-1} C_n + F_n^T \Gamma_{n+1} F_n\\
&\qquad \hat{\mathbf{x}}_{n/N} = \hat{\mathbf{x}}_{n/n-1} + P_{n/n-1}\mathbf{g}_n\\
&\qquad P_{n/N} = P_{n/n-1} - P_{n/n-1}\Gamma_n P_{n/n-1}
\end{aligned}} \quad \text{(BF smoothing)} \qquad (13.10.23)$$

The algorithm requires no new inversions—the quantity $D_n^{-1}C_n$ was computed as part of the forward Kalman algorithm. The RTS and BF algorithms have also been studied within the statistical time-series analysis literature [885,886,903,904]. Further details on smoothing algorithms may be found in [865].

The MATLAB function, `ksmooth.m`, implements the Kalman smoothing algorithm of Eq. (13.10.23). It has usage:

```
[L,Xs,Ps,V] = ksmooth(A,C,Q,R,Y,x0,S0);      % Bryson-Frazier smoothing
```

Its inputs are the state-space model parameters $\{A, C, Q, R\}$, the initial values $\bar{\mathbf{x}}_0$, Σ_0, and the observations $\mathbf{y}_n$, $0 \le n \le N$, arranged into a $r\times(N+1)$ matrix:

$$Y = \big[\mathbf{y}_0, \mathbf{y}_1, \dots, \mathbf{y}_n, \dots, \mathbf{y}_N\big]$$

The outputs are the smoothed estimates arranged into the $p\times(N+1)$ matrix:

$$X_s = \big[\hat{\mathbf{x}}_{0/N},\, \hat{\mathbf{x}}_{1/N},\, \dots,\, \hat{\mathbf{x}}_{n/N},\, \dots,\, \hat{\mathbf{x}}_{N/N}\big]$$

with corresponding error covariance matrices arranged into the $p\times p\times(N+1)$ three-dimensional array P_s, such that (in MATLAB notation):

$$\texttt{Ps(:,:,n+1)} = P_{n/N}\,, \quad 0 \le n \le N$$

The output L is the value of the negative-log-likelihood function calculated from Eq. (13.12.2). The quantity V is an optional output that stores the matrix $V_{n+1,n} = E[\mathbf{e}_{n+1/N}\mathbf{e}_{n/N}^T]$ into a $p\times p\times(N+1)$ array. This quantity is used in Sec. 13.13 in the maximum likelihood estimation of the state-space model parameters using the EM algorithm. A convenient expression for it can be derived as follows. We rewrite Eq. (13.10.7) in its orthogonal decomposition forms:

$$\begin{aligned}
\mathbf{e}_{n/n-1} &= \mathbf{e}_{n/N} + \hat{\mathbf{e}}_{n/n-1}\\
\mathbf{e}_{n+1/n} &= \mathbf{e}_{n+1/N} + \hat{\mathbf{e}}_{n+1/n}
\end{aligned} \qquad (13.10.24)$$

where $E[\mathbf{e}_{n+1/N}\hat{\mathbf{e}}_{n/n-1}^T] = E[\mathbf{e}_{n/N}\hat{\mathbf{e}}_{n+1/n}^T] = 0$, which follow from the fact that $\mathbf{e}_{n/N}, \mathbf{e}_{n+1/N}$ are estimation errors and must be orthogonal to all the observations Y, and therefore, must also be orthogonal to $\hat{\mathbf{e}}_{n/n-1}, \hat{\mathbf{e}}_{n+1/n}$ because the latter are made up from a subset of Y. Then, we find for the cross-covariance:

$$E[\mathbf{e}_{n+1/n}\mathbf{e}_{n/n-1}^T] = E[\mathbf{e}_{n+1/N}\mathbf{e}_{n/N}^T] + E[\hat{\mathbf{e}}_{n+1/n}\hat{\mathbf{e}}_{n/n-1}^T]\,, \quad \text{or,}$$

$$V_{n+1,n} = E[\mathbf{e}_{n+1/N}\mathbf{e}_{n/N}^T] = E[\mathbf{e}_{n+1/n}\mathbf{e}_{n/n-1}^T] - E[\hat{\mathbf{e}}_{n+1/n}\hat{\mathbf{e}}_{n/n-1}^T]$$

From Eq. (13.10.11) we have, $E[\mathbf{e}_{n+1/n}\mathbf{e}_{n/n-1}^T] = F_n P_{n/n-1}$, and from Eq. (13.10.20) we may replace $\hat{\mathbf{e}}_{n+1/n} = P_{n/n-1}\mathbf{g}_n$, to obtain:

$$V_{n+1,n} = E[\mathbf{e}_{n+1/N}\mathbf{e}_{n/N}^T] = F_n P_{n/n-1} - P_{n+1/n}E[\mathbf{g}_{n+1}\mathbf{g}_n^T]P_{n/n-1}$$

From the recursion (13.10.21), $\mathbf{g}_n = C_n^T D_n^{-1}\boldsymbol{\varepsilon}_n + F_n^T\mathbf{g}_{n+1}$, and $E[\mathbf{g}_{n+1}\boldsymbol{\varepsilon}_n^T] = 0$, we find:

$$E[\mathbf{g}_{n+1}\mathbf{g}_n^T] = E[\mathbf{g}_{n+1}(C_n^T D_n^{-1}\boldsymbol{\varepsilon}_n + F_n^T\mathbf{g}_{n+1})^T] = E[\mathbf{g}_{n+1}\mathbf{g}_{n+1}^T]F_n = \Gamma_{n+1}F_n$$

It follows then that:

$$\begin{aligned}
V_{n+1,n} = E[\mathbf{e}_{n+1/N}\mathbf{e}_{n/N}^T] &= F_n P_{n/n-1} - P_{n+1/n}\Gamma_{n+1}F_n P_{n/n-1}\\
&= [I - P_{n+1/n}\Gamma_{n+1}]F_n P_{n/n-1}
\end{aligned} \qquad (13.10.25)$$

13.11 Square-Root Algorithms

In its most basic form the Kalman filtering algorithm reads:

$$\begin{aligned}
&D = CPC^T + R\\
&G = PC^TD^{-1},\ K = AG\\
&P_f = P - PC^TD^{-1}CP\,, \qquad \hat{\mathbf{x}}_f = \hat{\mathbf{x}} + G(\mathbf{y} - C\hat{\mathbf{x}}) = (I - GC)\hat{\mathbf{x}} + G\mathbf{y}\\
&P_{\text{new}} = AP_fA^T + Q\,, \qquad \hat{\mathbf{x}}_{\text{new}} = A\hat{\mathbf{x}}_f = (A - KC)\hat{\mathbf{x}} + K\mathbf{y}
\end{aligned} \qquad (13.11.1)$$

where $\hat{\mathbf{x}}, \hat{\mathbf{x}}_f, \hat{\mathbf{x}}_{\text{new}}$ denote the current prediction, filtered estimate, and next prediction, $\hat{\mathbf{x}}_{n/n-1}, \hat{\mathbf{x}}_{n/n}, \hat{\mathbf{x}}_{n+1/n}$, and P, P_f, P_{new} denote the corresponding mean-square error covariance matrices $P_{n/n-1}, P_{n/n}, P_{n+1/n}$, and we dropped the time indices to simplify the notation. The matrices P, P_f, P_{new} must remain positive semi-definite during the iteration of the algorithm. Because of the subtraction required to calculate P_f, it is possible that rounding errors may destroy its positivity. The "square-root" formulations operate on the square-root factors of the covariance matrices, and thus, guarantee the positivity at each iteration step.

A positive semi-definite symmetric matrix P can always be written as the product of a *lower triangular* square-root factor S and its transpose S^T:

$$P = SS^T \qquad (13.11.2)$$

For example in MATLAB, one may use the built-in Cholesky factorization function `chol`, if P is strictly positive definite, with S constructed as:

```
S = chol(P)';              % S = lower triangular such that P = S*S'
```

If P is semi-definite with some positive and some zero eigenvalues, then one can apply the QR factorization to the square root of P obtained from its eigenvalue or SVD decomposition, with the following MATLAB construction:

```
[v,s] = eig(P);                 % v = eigenvector matrix, s = eigenvalues
P_sqrt = v*real(sqrt(s))*v';    % eigenvalues s are non-negative
[q,r] = qr(P_sqrt);             % QR-factorization, P_sqrt = q*r
S = r';                         % S = lower triangular such that P = S*S'
```

Example 13.11.1: For the following positive-definite case, we find:

$$P=\begin{bmatrix}6&5&4\\5&6&4\\4&4&3\end{bmatrix}=\begin{bmatrix}2.4495&0&0\\2.0412&1.3540&0\\1.6330&0.4924&0.3015\end{bmatrix}\begin{bmatrix}2.4495&0&0\\2.0412&1.3540&0\\1.6330&0.4924&0.3015\end{bmatrix}^T$$

This P has full rank as the product of two rank-3 matrices:

$$P=\begin{bmatrix}6&5&4\\5&6&4\\4&4&3\end{bmatrix}=\begin{bmatrix}2&-1&1\\1&-2&1\\1&-1&1\end{bmatrix}\begin{bmatrix}2&-1&1\\1&-2&1\\1&-1&1\end{bmatrix}^T$$

Similarly, for the following semi-definite P, we have:

$$P=\begin{bmatrix}5&4&3\\4&5&3\\3&3&2\end{bmatrix}=\begin{bmatrix}2.2361&0&0\\1.7889&1.3416&0\\1.3416&0.4472&0\end{bmatrix}\begin{bmatrix}2.2361&0&0\\1.7889&1.3416&0\\1.3416&0.4472&0\end{bmatrix}^T$$

This P has rank two as the product of the rank-2 matrices:

$$P=\begin{bmatrix}5&4&3\\4&5&3\\3&3&2\end{bmatrix}=\begin{bmatrix}2&1\\1&2\\1&1\end{bmatrix}\begin{bmatrix}2&1&1\\1&2&1\end{bmatrix}$$

and has eigenvalues $\{0,1,11\}$. □

The Joseph forms of the covariance updating equations promote the numerical stability of the algorithm because they consist of the sum of positive semidefinite terms (provided of course that P is already positive semidefinite), but require twice as many operations as the conventional forms:

$$\begin{aligned}P_f &= (I-GC)P(I-GC)^T+GRG^T\\ P_{\text{new}} &= AP_fA^T+Q=(A-KC)P(A-KC)^T+KRK^T+Q\end{aligned}\tag{13.11.3}$$

Let $\bar R,\bar Q$ be lower-triangular square root factors for R,Q, so that $R=\bar R\bar R^T$ and $Q=\bar Q\bar Q^T$, and substitute the factorizations $P=SS^T$ and $P_f=S_fS_f^T$ in Eq. (13.11.3):

$$S_fS_f^T=(I-GC)SS^T(I-GC)^T+G\bar R\bar R^TG=\big[(I-GC)S,G\bar R\big]\begin{bmatrix}S^T(I-GC)^T\\ \bar R^TG^T\end{bmatrix}$$

Then, S_f can be obtained from the upper-triangular factor resulting by applying the QR-factorization to the $(p+r)\times p$ matrix:

$$\begin{bmatrix}S^T(I-GC)^T\\ \bar R^TG^T\end{bmatrix}=U\begin{bmatrix}S_f^T\\0\end{bmatrix},\quad S_f^T=\text{upper-triangular}$$

where U is a $(p+r)\times(p+r)$ orthogonal matrix. The lower-triangular version of this relationship reads:

$$\big[(I-GC)S,\,G\bar R\big]=\big[S_f\,,0\big]U^T\tag{13.11.4}$$

where the S_f is lower-triangular and the dimensions of the matrices are:

$$\big[\underbrace{(I-GC)S}_{p\times p},\ \underbrace{G\bar R}_{p\times r}\big]=\big[\underbrace{S_f}_{p\times p},\ \underbrace{0}_{p\times r}\big]U^T$$

Since $U^TU=I$, we verify:

$$\big[(I-GC)S,G\bar R\big]\begin{bmatrix}S^T(I-GC)^T\\ \bar R^TG^T\end{bmatrix}=\big[S_f\,,0\big]U^TU\begin{bmatrix}S_f^T\\0\end{bmatrix}=S_fS_f^T$$

For the time-update step, we note that,

$$P_{\text{new}}=S_{\text{new}}S_{\text{new}}^T=AP_fA^T+Q=AS_fS_f^T+\bar Q\bar Q^T=\big[AS_f,\,\bar Q\big]\begin{bmatrix}S_f^TA^T\\ \bar Q^T\end{bmatrix}$$

so that we may obtain S_{new} from the lower-triangular version of the QR-algorithm applied to the $p\times(2p)$ matrix:

$$\big[AS_f,\,\bar Q\big]=\big[S_{\text{new}},\,0\big]U^T\tag{13.11.5}$$

with another $(2p)\times(2p)$ orthogonal matrix U. Combining Eqs. (13.11.4) and (13.11.5), we summarize the measurement and time updating algorithm that propagates the square-root lower-triangular factors S,S_f,S_{new}:

$$\begin{aligned}&P=SS^T\,,\quad D=CPC^T+R\,,\quad G=PC^TD^{-1}\\ &\big[(I-GC)S,\,G\bar R\big]=\big[S_f\,,0\big]U^T\\ &\big[AS_f,\,\bar Q\big]=\big[S_{\text{new}},\,0\big]U^T\end{aligned}$$

The intermediate step of computing S_f can be avoided by applying the lower-triangular version of the QR-algorithm to the larger $p\times(2p+r)$ matrix:

$$\big[(A-KC)S,\,K\bar R,\,\bar Q\big]=\big[S_{\text{new}},\,0,\,0\big]U^T\,,\quad S_{\text{new}}=\text{lower-triangular}\tag{13.11.6}$$

which is equivalent to the second Joseph form in Eq. (13.11.3):

$$(A-KC)SS^T(A-KC)^T+K\bar R\bar R^TK^T+\bar Q\bar Q^T=\big[(A-KC)S,\,K\bar R,\,\bar Q\big]\begin{bmatrix}S^T(A-KC)^T\\ \bar R^TK^T\\ \bar Q^T\end{bmatrix}$$

$$=\big[S_{\text{new}},\,0,\,0\big]U^TU\begin{bmatrix}S_{\text{new}}^T\\0\\0\end{bmatrix}=S_{\text{new}}S_{\text{new}}^T$$

Restoring the time-indices, we obtain the following square-root algorithms based on the Joseph forms. Given $S_{n/n-1}$, calculate:

$$\boxed{\begin{aligned}&P_{n/n-1}=S_{n/n-1}S_{n/n-1}^T\\ &D_n=C_nP_{n/n-1}C_n^T+R_n\,,\quad G_n=P_{n/n-1}C_n^TD_n^{-1}\\ &\big[(I-G_nC_n)S_{n/n-1},\,G_n\bar R_n\big]=\big[S_{n/n}\,,0\big]U^T\\ &\big[A_nS_{n/n},\,\bar Q_n\big]=\big[S_{n+1/n},\,0\big]U^T\end{aligned}}\tag{13.11.7}$$

or, going directly to $S_{n+1/n}$:

$$\boxed{[(A_n - K_nC_n)S_{n/n-1},\, K_n\bar{R}_n,\, \bar{Q}_n] = [S_{n+1/n},\, 0,\, 0]U^T} \tag{13.11.8}$$

Example 13.11.2: Consider a system with constant model parameters:

$$A = \begin{bmatrix} 0.5 & 0.1 \\ 0.2 & 0.4 \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \quad Q = \begin{bmatrix} 1 & 2 \\ 2 & 5 \end{bmatrix}, \quad R = \begin{bmatrix} 9 & 6 \\ 6 & 8 \end{bmatrix}$$

and lower-triangular square-root factors $\bar{Q}, \bar{R}$ such that $Q = \bar{Q}\bar{Q}^T$ and $R = \bar{R}\bar{R}^T$:

$$\bar{Q} = \begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix}, \quad \bar{R} = \begin{bmatrix} 3 & 0 \\ 2 & 2 \end{bmatrix}$$

and initial square-root factor S and covariance matrix P:

$$S = \begin{bmatrix} 1.3184 & 0 \\ 1.8820 & 1.4731 \end{bmatrix}, \quad P = SS^T = \begin{bmatrix} 1.7383 & 2.4813 \\ 2.4813 & 5.7118 \end{bmatrix}$$

Then, we calculate D and G:

$$D = CPC^T + R = \begin{bmatrix} 21.4126 & 14.1931 \\ 14.1931 & 13.7118 \end{bmatrix}, \quad G = PC^TD^{-1} = \begin{bmatrix} 0.2457 & -0.0733 \\ 0.3393 & 0.0653 \end{bmatrix}$$

and the covariance matrices P_f, P_{new} from the conventional algorithm:

$$P_f = P - GDG^T = \begin{bmatrix} 0.8836 & 0.8874 \\ 0.8874 & 2.5585 \end{bmatrix}, \quad P_{\text{new}} = AP_fA^T + Q = \begin{bmatrix} 1.3352 & 2.3859 \\ 2.3859 & 5.5867 \end{bmatrix}$$

Next, we form the matrix $X \equiv [(I-GC)S,\, G\bar{R}]$ and apply the QR-algorithm to its transpose to get the factorization:

$$X = [(I - GC)S,\, G\bar{R}] = \left[\begin{array}{cc|cc} 0.6702 & -0.2539 & 0.5903 & -0.1467 \\ 0.6730 & 0.8770 & 1.1486 & 0.1306 \end{array}\right]$$

$$= \left[\begin{array}{cc|cc} 0.9400 & 0 & 0 & 0 \\ 0.9440 & 1.2913 & 0 & 0 \end{array}\right] U^T = [S_f,\, 0]U^T \equiv LU^T$$

$$U = \begin{bmatrix} 0.7130 & -0.0000 & -0.3139 & -0.6270 \\ -0.2701 & 0.8766 & 0.1350 & -0.3747 \\ 0.6280 & 0.4304 & 0.1856 & 0.6212 \\ -0.1560 & 0.2152 & -0.9213 & 0.2839 \end{bmatrix}$$

This was obtained from the MATLAB code:

```
[U,r] = qr(X'); L = r';
```

Thus, we determine S_f, and verify that $P_f = S_fS_f^T$ is the same as above:

$$S_f = \begin{bmatrix} 0.9400 & 0 \\ 0.9440 & 1.2913 \end{bmatrix}, \quad P_f = S_fS_f^T = \begin{bmatrix} 0.8836 & 0.8874 \\ 0.8874 & 2.5585 \end{bmatrix}$$

Next, with this S_f, we calculate the matrix $X = [AS_f, \bar{Q}]$ and apply the QR-factorization to its transpose to get:

$$X = [AS_f, \bar{Q}] = \left[\begin{array}{cc|cc} 0.5644 & 0.1291 & 1 & 0 \\ 0.5656 & 0.5165 & 2 & 1 \end{array}\right]$$

$$= \left[\begin{array}{cc|cc} 1.1555 & 0 & 0 & 0 \\ 2.0648 & 1.1503 & 0 & 0 \end{array}\right] U^T = [S_{\text{new}},\, 0] \equiv LU^T$$

$$U = \begin{bmatrix} 0.4884 & -0.3851 & 0.5883 & 0.5168 \\ 0.1117 & 0.2484 & -0.5947 & 0.7564 \\ 0.8654 & 0.1852 & -0.2552 & -0.3894 \\ 0.0000 & 0.8693 & 0.4849 & 0.0957 \end{bmatrix}$$

with L, U obtained from the same MATLAB code as above. Thus, we identify,

$$S_{\text{new}} = \begin{bmatrix} 1.1555 & 0 \\ 2.0648 & 1.1503 \end{bmatrix} \quad \Rightarrow \quad P_{\text{new}} = S_{\text{new}}S_{\text{new}}^T = \begin{bmatrix} 1.3352 & 2.3859 \\ 2.3859 & 5.5867 \end{bmatrix}$$

For the direct method of Eq. (13.11.8), we calculate the gain $K = AG$,

$$K = \begin{bmatrix} 0.1568 & -0.0301 \\ 0.1849 & 0.0115 \end{bmatrix}$$

and the matrix:

$$[(A - KC)S,\, K\bar{R},\, \bar{Q}] = \left[\begin{array}{cc|cc|cc} 0.4024 & -0.0392 & 0.4100 & -0.0603 & 1 & 0 \\ 0.4033 & 0.3000 & 0.5775 & 0.0229 & 2 & 1 \end{array}\right]$$

$$= \left[\begin{array}{cc|cc|cc} 1.1555 & 0 & 0 & 0 & 0 & 0 \\ 2.0648 & 1.1503 & 0 & 0 & 0 & 0 \end{array}\right] U^T = [S_{\text{new}},\, 0, 0]U^T$$

which generates the same S_{new} as the two-step procedure. □

Using the Joseph forms in conjunction with the square root factorizations provides, in effect, double protection at the expense of increased computation. There exist a variety of other square-root algorithms [863,865,888–896], including some for the smoothing problem. One of the standard ones [889] employs the triangularization:

$$\begin{bmatrix} \bar{R} & CS \\ 0 & S \end{bmatrix} = \begin{bmatrix} \bar{D} & 0 \\ PC^T\bar{D}^{-T} & S_f \end{bmatrix} U^T \tag{13.11.9}$$

where $\bar{D}$ is a lower-triangular square root factor of $D = CPC^T + R = \bar{D}\bar{D}^T$. Its correctness can be verified by noting that $P_f = P - PC^TD^{-1}CP$ and by forming the products:

$$\begin{bmatrix} \bar{R} & CS \\ 0 & S \end{bmatrix}\begin{bmatrix} \bar{R} & CS \\ 0 & S \end{bmatrix}^T = \begin{bmatrix} D & CP \\ PC^T & P \end{bmatrix}$$

$$\begin{bmatrix} \bar{D} & 0 \\ PC^T\bar{D}^{-T} & S_f \end{bmatrix} U^TU \begin{bmatrix} \bar{D} & 0 \\ PC^T\bar{D}^{-T} & S_f \end{bmatrix}^T = \begin{bmatrix} D & CP \\ PC^T & P_f + PC^TD^{-1}CP \end{bmatrix}$$

The triangularization operation produces $\bar{D}$, S_f, and $PC^T\bar{D}^{-T}$, the latter being part of the required Kalman gain $G = PC^TD^{-1} = [PC^T\bar{D}^{-T}]\bar{D}^{-1}$, where the division by the lower-triangular matrix $\bar{D}$ is an efficient operation. Therefore, the computation of the filtered estimate can also be done efficiently:

$$\hat{\mathbf{x}}_f = \hat{\mathbf{x}} + G(\mathbf{y} - C\hat{\mathbf{x}}) = \hat{\mathbf{x}} + [PC^T\bar{D}^{-T}]\bar{D}^{-1}(\mathbf{y} - C\hat{\mathbf{x}})$$

A direct-updating version [891] is possible in this case too by the triangularization of the following matrix:

$$\begin{bmatrix} \bar{R} & CS & 0 \\ 0 & AS & \bar{Q} \end{bmatrix} = \begin{bmatrix} \bar{D} & 0 & 0 \\ APC^T\bar{D}^{-T} & S_{\text{new}} & 0 \end{bmatrix} U^T$$

Its correctness follows from $P_{\text{new}} = S_{\text{new}}S_{\text{new}}^T = APA^T - APC^TD^{-1}CPA^T + Q$ and by comparing the products:

$$\begin{bmatrix} \bar{R} & CS & 0 \\ 0 & AS & \bar{Q} \end{bmatrix}\begin{bmatrix} \bar{R}^T & 0 \\ S^TC^T & S^TA^T \\ 0 & \bar{Q}^T \end{bmatrix} = \begin{bmatrix} D & CPA^T \\ APC^T & APA^T + Q \end{bmatrix}$$

$$\begin{bmatrix} \bar{D} & 0 & 0 \\ APC^T\bar{D}^{-T} & S_{\text{new}} & 0 \end{bmatrix} U^TU \begin{bmatrix} \bar{D}^T & \bar{D}^{-1}CPA^T \\ 0 & S_{\text{new}}^T \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} D & CPA^T \\ APC^T & P_{\text{new}} + APC^TD^{-1}CPA^T \end{bmatrix}$$

This factorization allows also the efficient computation of the predicted estimate:

$$\hat{\mathbf{x}}_{\text{new}} = A\hat{\mathbf{x}} + [APC^T\bar{D}^{-T}]\bar{D}^{-1}(\mathbf{y} - C\hat{\mathbf{x}})$$

Restoring the time indices, we summarize the two-step algorithm for the computation of the square-root factors of the covariances and the estimates:

$$\boxed{\begin{aligned} &\begin{bmatrix} \bar{R}_n & C_nS_{n/n-1} \\ 0 & S_{n/n-1} \end{bmatrix} = \begin{bmatrix} \bar{D}_n & 0 \\ P_{n/n-1}C_n^T\bar{D}_n^{-T} & S_{n/n} \end{bmatrix} U^T \\ &\hat{\mathbf{x}}_{n/n} = \hat{\mathbf{x}}_{n/n-1} + [P_{n/n-1}C_n^T\bar{D}_n^{-T}]\bar{D}_n^{-1}(\mathbf{y}_n - C_n\hat{\mathbf{x}}_{n/n-1}) \\ &[A_nS_{n/n},\, \bar{Q}_n] = [S_{n+1/n},\, 0]U^T \\ &\hat{\mathbf{x}}_{n+1/n} = A_n\hat{\mathbf{x}}_{n/n} \end{aligned}} \tag{13.11.10}$$

and for the direct method:

$$\boxed{\begin{aligned} &\begin{bmatrix} \bar{R}_n & C_nS_{n/n-1} & 0 \\ 0 & A_nS_n & \bar{Q}_n \end{bmatrix} = \begin{bmatrix} \bar{D}_n & 0 & 0 \\ A_nP_{n/n-1}C_n^T\bar{D}_n^{-T} & S_{n+1/n} & 0 \end{bmatrix} U^T \\ &\hat{\mathbf{x}}_{n+1/n} = A_n\hat{\mathbf{x}}_{n/n-1} + [A_nP_{n/n-1}C_n^T\bar{D}_n^{-T}]\bar{D}_n^{-1}(\mathbf{y}_n - C_n\hat{\mathbf{x}}_{n/n-1}) \end{aligned}} \tag{13.11.11}$$

Example 13.11.3: For the model defined in Example 13.11.2 and the given starting S, we carry out the triangularization of the following matrix using the QR-factorization:

$$\begin{bmatrix} \bar{R} & CS \\ 0 & S \end{bmatrix} = \left[\begin{array}{cc|cc} 3 & 0 & 3.2004 & 1.4731 \\ 2 & 2 & 1.8820 & 1.4731 \\ \hline 0 & 0 & 1.3184 & 0 \\ 0 & 0 & 1.8820 & 1.4731 \end{array}\right]$$

$$= \left[\begin{array}{cc|cc} 4.6274 & 0 & 0 & 0 \\ 3.0672 & 2.0746 & 0 & 0 \\ \hline 0.9119 & -0.1521 & 0.9400 & 0 \\ 1.7706 & 0.1355 & 0.9440 & 1.2913 \end{array}\right] U^T = \begin{bmatrix} \bar{D} & 0 \\ PC^T\bar{D}^{-T} & S_f \end{bmatrix} U^T$$

from which we extract:

$$\bar{D} = \begin{bmatrix} 4.6274 & 0 \\ 3.0672 & 2.0746 \end{bmatrix}, \quad PC^T\bar{D}^{-T} = \begin{bmatrix} 0.9119 & -0.1521 \\ 1.7706 & 0.1355 \end{bmatrix}, \quad S_f = \begin{bmatrix} 0.9400 & 0 \\ 0.9440 & 1.2913 \end{bmatrix}$$

Using the quantities D, P computed in Example 13.11.2, we verify the factorization:

$$D = \begin{bmatrix} 21.4126 & 14.1931 \\ 14.1931 & 13.7118 \end{bmatrix} = \begin{bmatrix} 4.6274 & 0 \\ 3.0672 & 2.0746 \end{bmatrix}\begin{bmatrix} 4.6274 & 0 \\ 3.0672 & 2.0746 \end{bmatrix}^T = \bar{D}\bar{D}^T$$

Similarly, we may verify the correctness of $PC^T\bar{D}^{-T}$ and S_f. Next, we illustrate the direct method. We form the following matrix and triangularize it by applying the QR-factorization to its transpose:

$$\begin{bmatrix} \bar{R} & CS & 0 \\ 0 & AS & \bar{Q} \end{bmatrix} = \left[\begin{array}{cc|cc|cc} 3 & 0 & 3.2004 & 1.4731 & 0 & 0 \\ 2 & 2 & 1.8820 & 1.4731 & 0 & 0 \\ \hline 0 & 0 & 0.8474 & 0.1473 & 1 & 0 \\ 0 & 0 & 1.0165 & 0.5892 & 2 & 1 \end{array}\right]$$

$$= \left[\begin{array}{cc|cc|cc} 4.6274 & 0 & 0 & 0 & 0 & 0 \\ 3.0672 & 2.0746 & 0 & 0 & 0 & 0 \\ \hline 0.6330 & -0.0625 & 1.1555 & 0 & 0 & 0 \\ 0.8906 & 0.0238 & 2.0648 & 1.1503 & 0 & 0 \end{array}\right] U^T = \begin{bmatrix} \bar{D} & 0 & 0 \\ APC^T\bar{D}^{-T} & S_{\text{new}} & 0 \end{bmatrix} U^T$$

and we extract the same $\bar{D}$ as above and the same S_{new} as in Example 13.11.2. □

13.12 *Maximum Likelihood Parameter Estimation*

One issue in applying the Kalman filter is the determination of the state-space model parameters $\{A, C, Q, R\}$ and the initial values $\{\bar{\mathbf{x}}_0, \Sigma_0\}$. If the dynamics is known, as for example in α–β radar tracking, then $\{A, C\}$ are known, but not necessarily the noise covariances $\{Q, R\}$ or the initial values. If the dynamics is not known, as for example in the so-called unobserved components models for microeconomic applications, then all the parameters must be estimated.

Maximum likelihood (ML) is a commonly used method for estimating the model parameters. Assuming a time-invariant model, then given a set of $N + 1$ observations, $Y = \{\mathbf{y}_0, \mathbf{y}_1, \dots, \mathbf{y}_N\}$, one determines the parameters $\{A, C, Q, R, \bar{\mathbf{x}}_0, \Sigma_0\}$ that maximize the joint probability density $p(\mathbf{y}_0,\mathbf{y}_1,\dots,\mathbf{y}_N)$ or $p(Y)$ for short. Equivalently, one may minimize the negative of the log-likelihood function:

$$L(Y)=-\log p(Y)=\min \tag{13.12.1}$$

This problem becomes tractable under the gaussian assumption for the state-space model. The Kalman filtering algorithm generates the equivalent orthogonalized observation basis of innovations $Y=\{\mathbf{y}_0,\mathbf{y}_1,\dots,\mathbf{y}_N\}=\{\boldsymbol{\varepsilon}_0,\boldsymbol{\varepsilon}_1,\dots,\boldsymbol{\varepsilon}_N\}$, which are mutually uncorrelated and gaussian, and hence, mutually independent. Therefore, the joint density factors into the marginal densities:

$$p(\mathbf{y}_0,\mathbf{y}_1,\dots,\mathbf{y}_N)=p(\boldsymbol{\varepsilon}_0,\boldsymbol{\varepsilon}_1,\dots,\boldsymbol{\varepsilon}_N)=\prod_{n=0}^{N}p(\boldsymbol{\varepsilon}_n)=\prod_{n=0}^{N}\frac{\exp(-\boldsymbol{\varepsilon}_n^T D_n^{-1}\boldsymbol{\varepsilon}_n/2)}{(2\pi)^{r/2}(\det D_n)^{1/2}}$$

where $r=\dim(\mathbf{y}_n)$ and D_n is the covariance of $\boldsymbol{\varepsilon}_n$ generated by the Kalman filtering algorithm. Thus, the log-likelihood function can be expressed up to a constant by [897]:

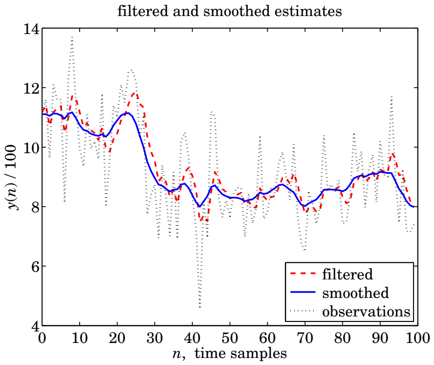
$$L(Y)=-\log p(Y)=\frac{1}{2}\sum_{n=0}^{N}\Big[\log(\det D_n)+\boldsymbol{\varepsilon}_n^T D_n^{-1}\boldsymbol{\varepsilon}_n\Big]+\text{const.} \tag{13.12.2}$$

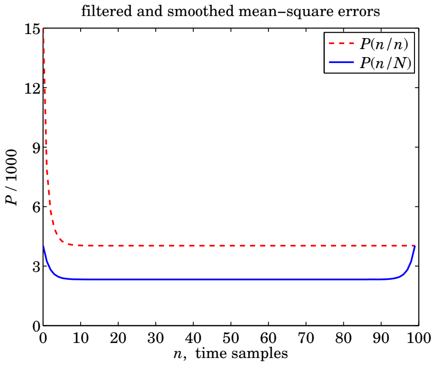
Let θ denote all or a subset of the model parameters $\{A,C,Q,R,\bar{\mathbf{x}}_0,\Sigma_0\}$ that one wishes to estimate. The dependence of $L(Y)$ on θ will be indicated explicitly by $L_\theta(Y)$.

There exist several methods [897–911] of minimizing $L_\theta(Y)$, some requiring only the evaluation of the function $L_\theta(Y)$ for various values θ, some requiring also the first, and some the second, derivatives of $L_\theta(Y)$ with respect to θ. The EM algorithm is an alternative, iterative, minimization method and we discuss it in the next section.

The MATLAB function `kfilt` can evaluate $L_\theta(Y)$ at any θ, and thus, it can be used, in conjunction with the built-in function `fminsearch` of MATLAB's optimization toolbox to perform the minimization. This function uses the Nelder-Mead simplex direct search method in which the derivatives of L_θ are not required. We illustrate the approach with two examples.

Example 13.12.1: The Nile River data set has been used as a benchmark in a variety of statistical methods [912]. It represents the annual volume of the Nile River (discharged at Aswan, in units of 10^8 m^3) from 1871 to 1970. It is depicted in the left figure below (dotted line of observations).





Following [903], we model it as a local-level (random walk in noise) model:

$$\begin{aligned}x_{n+1}&=x_n+w_n\\ y_n&=x_n+v_n\end{aligned}$$

with noise variances $Q=\sigma_w^2$ and $R=\sigma_v^2$ to be estimated, so that the parameter vector is $\theta=[\sigma_w^2,\sigma_v^2]^T$. The MATLAB code below defines the function L_θ with the help of `kfilt`, with the Kalman filter initialized to the arbitrary values $\bar{x}_0=0$ and $\Sigma_0=10^7$. It then calls `fminsearch` with the arbitrary initial values of the parameters $\theta_0=[1,1]^T$, and returns the "optimum" values:

$$\theta=\begin{bmatrix}\sigma_w^2\\ \sigma_v^2\end{bmatrix}=\begin{bmatrix}1468.5\\ 15099.7\end{bmatrix}$$

```
y = [ 1120  1160   963  1210  1160  1160   813  1230  1370  1140 ...
       995   935  1110   994  1020   960  1180   799   958  1140 ...
      1100  1210  1150  1250  1260  1220  1030  1100   774   840 ...
       874   694   940   833   701   916   692  1020  1050   969 ...
       831   726   456   824   702  1120  1100   832   764   821 ...
       768   845   864   862   698   845   744   796  1040   759 ...
       781   865   845   944   984   897   822  1010   771   676 ...
       649   846   812   742   801  1040   860   874   848   890 ...
       744   749   838  1050   918   986   797   923   975   815 ...
      1020   906   901  1170   912   746   919   718   714   740 ];

A = 1; C = 1; x0 = 0; S0 = 1e7;                   % fixed model parameters

L = @(th) kfilt(A, C, th(1), th(2), y, x0, S0);   % likelihood function

th0 = [1; 1];                                     % initialize search
th = fminsearch(L, th0);                          % Nelder-Mead search

Q = th(1); R = th(2);                             % estimated Q,R

[Lmin,x,P,xf,Pf] = kfilt(A,C,Q,R,y,x0,S0);        % run Kalman filter
[Lmin,xs,Ps] = ksmooth(A,C,Q,R,y,x0,S0);          % run Kalman smoother

t = 0:length(y)-1;
figure; plot(t,xf/100,'--',t,xs/100,'-' ,t,y/100,'k:');
figure; plot(t,Pf(:)/1e3,'--',t,Ps(:)/1e3,'-');
```

The Kalman filter and smoother are then run with the optimum parameter values for Q,R, and the resulting filtered and smoothed estimates, $\hat{x}_{n/n}$ and $\hat{x}_{n/N}$ are plotted on the left figure. The right figure plots the corresponding mean-square errors, $P_{n/n}$ and $P_{n/N}$. We note that $P_{n/N}$ is smaller than $P_{n/n}$ since it uses all the observations. It rises up at the end to agree with the filtered error, that is, $P_{n/n}=P_{n/N}$, when $n=N$. □

Example 13.12.2: Consider the α–β tracking model discussed in Example 13.7.3 and defined by the state-space model:

$$\begin{bmatrix}x_{n+1}\\ \dot{x}_{n+1}\end{bmatrix}=\begin{bmatrix}1&T\\0&1\end{bmatrix}\begin{bmatrix}x_n\\ \dot{x}_n\end{bmatrix}+\begin{bmatrix}T^2/2\\ T\end{bmatrix}a_n\,,\quad y_n=[1,0]\begin{bmatrix}x_n\\ \dot{x}_n\end{bmatrix}+v_n$$

with model matrices:

$$A = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix}, \quad C = [1,0]\,, \quad Q = \begin{bmatrix} T^4/4 & T^3/2 \\ T^3/2 & T^2 \end{bmatrix} \sigma_a^2 \equiv Q_T\,\sigma_a^2\,, \quad R = \sigma_v^2$$

The model is simulated with the following values of the parameters $T = 1$, $\sigma_a = 0.02$, $\sigma_v = 2$, and generating $N + 1$ noisy position measurements, y_n, $0 \le n \le N$, where $N = 300$, starting with the initial state-vector $\bar{\mathbf{x}}_0 = [0, 0.1]^T$. The generated observations y_n are then used to estimate the model parameters $\theta = [\sigma_a, \sigma_v]^T$ starting from the initial values $\theta_0 = [0, 0]^T$. The likelihood function is defined as in the previous example, with the Kalman filter run with the initial values:

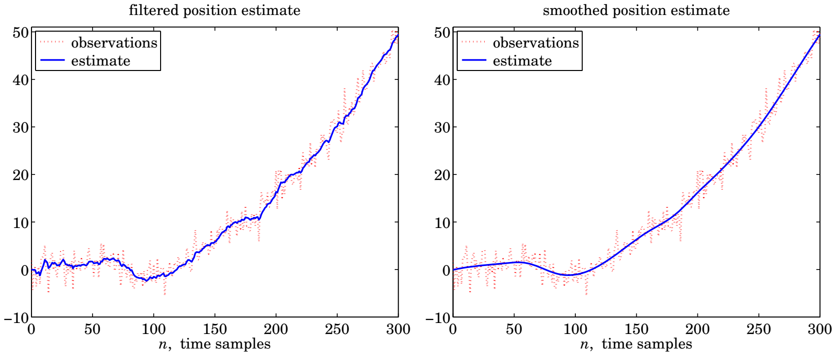
$$\bar{\mathbf{x}}_0 = \begin{bmatrix} 0 \\ 0.1 \end{bmatrix}, \quad \Sigma_0 = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}$$

The resulting estimated parameters returned from `mfinsearch`, and the corresponding estimated covariance matrices Q, R, are:

$$\theta = \begin{bmatrix} \sigma_a \\ \sigma_v \end{bmatrix} = \begin{bmatrix} -0.0199 \\ 2.0237 \end{bmatrix} \quad \Rightarrow \quad Q = Q_T\,\sigma_a^2\,, \quad R = \sigma_v^2$$

The absolute value $|\sigma_a|$ is close to the true value of 0.02. The sign does not matter since Q, R depend on the squares of σ_a, σ_v.

The Kalman filter and smoother are then run with the estimated Q, R and the resulting filtered and smoothed estimates are shown in the figure below.



The MATLAB code used to generate these graphs is as follows:

```
N = 301; T = 1; Tmax = (N-1)*T; t = 0:T:Tmax;

sa = 0.02; sv = 2;
A = [1, T; 0, 1]; C = [1, 0];                   % model parameters
QT = [T^4/4, T^3/2; T^3/2, T^2];
Q = QT*sa^2; R = sv^2;

seed = 100;  randn('state',seed);               % generate noise inputs
v = sv * randn(1,length(t));
w = [T^2/2; T] * sa * randn(1,length(t));
```

```
x0 = [0; 0.1]; S0 = 0.1 * eye(2);

x(:,1) = x0;
for n=1:N-1                                     % generate model signals
   x(:,n+1) = A*x(:,n) + w(n);
   y(n) = C*x(:,n) + v(n);
end
   y(N) = C*x(:,N) + v(N);

L = @(th) kfilt(A,C, QT * th(1)^2, th(2)^2, y,x0,S0);    % likelihood

th0 = [0, 0]';                                         % initialize search
th = fminsearch(L, th0);                               % Nelder-Mead search

Q = QT * th(1)^2; R = th(2)^2;                         % estimated Q,R

[Lmin,X,P,Xf,Pf] = kfilt(A, C, Q, R, y, x0, S0);       % run Kalman filter
[Lmin,Xs,Ps] = ksmooth(A, C, Q, R, y, x0, S0);         % run Kalman smoother

figure; plot(t,y,':', t,Xf(1,:),'-');                  % plot position only
figure; plot(t,y,':', t,Xs(1,:),'-');
```

13.13 Parameter Estimation with the EM Algorithm

The application of the Expectation-Maximization (EM) algorithm [905,906] to the estimation of the state-space model parameters has been discussed in [909–911].

We begin by developing a solution for the model parameters $\{A, C, Q, R, \bar{\mathbf{x}}_0, \Sigma_0\}$ in terms of the signals $\mathbf{x}_n, \mathbf{y}_n$ of the state-space model, then convert that into a computable iterative algorithm, and then show that it is equivalent to the EM algorithm. The approach allows us to handle also the case of a noise covariance matrix Q of less than full rank. We start with a standard time-invariant state-space model iterated over the time interval $0 \le n \le N$:

$$\begin{aligned} \mathbf{x}_{n+1} &= A\,\mathbf{x}_n + \mathbf{w}_n \\ \mathbf{y}_n &= C\,\mathbf{x}_n + \mathbf{v}_n \end{aligned} \tag{13.13.1}$$

with start-up value $\mathbf{x}_0$ with mean and covariance $\bar{\mathbf{x}}_0, \Sigma_0$. The noise covariances are:

$$\begin{gathered} E[\mathbf{w}_n \mathbf{w}_i^T] = Q\delta_{ni}\,, \quad E[\mathbf{v}_n \mathbf{v}_i^T] = R\delta_{ni}\,, \quad E[\mathbf{w}_n \mathbf{v}_i^T] = 0 \\ E[\mathbf{w}_n \mathbf{x}_0^T] = 0\,, \quad E[\mathbf{v}_n \mathbf{x}_0^T] = 0 \end{gathered} \tag{13.13.2}$$

These assumptions imply that $E[\mathbf{v}_n \mathbf{x}_n^T] = 0$, which leads to

$$E[\mathbf{v}_n \mathbf{x}_n^T] = E[(\mathbf{y}_n - C\mathbf{x}_n)\mathbf{x}_n^T] = 0 \quad \Rightarrow \quad E[\mathbf{y}_n \mathbf{x}_n^T] = CE[\mathbf{x}_n \mathbf{x}_n^T]$$

and using this result, we obtain:

$$\begin{aligned} R &= E[\mathbf{v}_n \mathbf{v}_n^T] = E[(\mathbf{y}_n - C\mathbf{x}_n)(\mathbf{y}_n - C\mathbf{x}_n)^T] = \\ &= E[\mathbf{y}_n \mathbf{y}_n^T] - CE[\mathbf{x}_n \mathbf{y}_n^T] - E[\mathbf{y}_n \mathbf{x}_n^T]C^T + CE[\mathbf{x}_n \mathbf{x}_n^T]C^T = E[\mathbf{y}_n \mathbf{y}_n^T] - CE[\mathbf{x}_n \mathbf{y}_n^T] \end{aligned}$$

Similarly, using $E[\mathbf{w}_n \mathbf{x}_n^T] = 0$, we find:

$$
\begin{aligned}
&E[\mathbf{w}_n\mathbf{x}_n^T]= E\big[(\mathbf{x}_{n+1}-A\mathbf{x}_n)\mathbf{x}_n^T\big]=0 \quad\Rightarrow\quad E[\mathbf{x}_{n+1}\mathbf{x}_n^T]= AE[\mathbf{x}_n\mathbf{x}_n^T]\\
&Q = E[\mathbf{w}_n\mathbf{w}_n^T]= E\big[(\mathbf{x}_{n+1}-A\mathbf{x}_n)(\mathbf{x}_{n+1}-A\mathbf{x}_n)^T\big] =\\
&\quad = E[\mathbf{x}_{n+1}\mathbf{x}_{n+1}^T]-AE[\mathbf{x}_n\mathbf{x}_{n+1}^T]-E[\mathbf{x}_{n+1}\mathbf{x}_n^T]A^T + AE[\mathbf{x}_n\mathbf{x}_n^T]A^T\\
&\quad = E[\mathbf{x}_{n+1}\mathbf{x}_{n+1}^T]-AE[\mathbf{x}_n\mathbf{x}_{n+1}^T]
\end{aligned}
$$

We collect the above together,

$$
\begin{aligned}
&E[\mathbf{y}_n\mathbf{x}_n^T]= CE[\mathbf{x}_n\mathbf{x}_n^T]\\
&R = E[\mathbf{y}_n\mathbf{y}_n^T]-CE[\mathbf{x}_n\mathbf{y}_n^T]-E[\mathbf{y}_n\mathbf{x}_n^T]C^T + CE[\mathbf{x}_n\mathbf{x}_n^T]C^T =\\
&\quad = E[\mathbf{y}_n\mathbf{y}_n^T]-CE[\mathbf{x}_n\mathbf{y}_n^T]\\
&E[\mathbf{x}_{n+1}\mathbf{x}_n^T]= AE[\mathbf{x}_n\mathbf{x}_n^T]\\
&Q = E[\mathbf{x}_{n+1}\mathbf{x}_{n+1}^T]-AE[\mathbf{x}_n\mathbf{x}_{n+1}^T]-E[\mathbf{x}_{n+1}\mathbf{x}_n^T]A^T + AE[\mathbf{x}_n\mathbf{x}_n^T]A^T =\\
&\quad = E[\mathbf{x}_{n+1}\mathbf{x}_{n+1}^T]-AE[\mathbf{x}_n\mathbf{x}_{n+1}^T]
\end{aligned}
\tag{13.13.3}
$$

Since these are valid for each n in the interval $0\le n\le N$, they will also be valid if we form the average sums over the same interval, for example,

$$
\sum_{n=0}^{N}E[\mathbf{y}_n\mathbf{x}_n^T]= C\sum_{n=0}^{N}E[\mathbf{x}_n\mathbf{x}_n^T] \quad\text{and}\quad \sum_{n=0}^{N-1}E[\mathbf{x}_{n+1}\mathbf{x}_n^T]= A\sum_{n=0}^{N-1}E[\mathbf{x}_n\mathbf{x}_n^T]
$$

This leads us to define the average quantities:

$$
\begin{aligned}
U_{xx} &= \frac{1}{N+1}\sum_{n=0}^{N}E[\mathbf{x}_n\mathbf{x}_n^T] &\qquad V_{xx} &= \frac{1}{N}\sum_{n=0}^{N-1}E[\mathbf{x}_n\mathbf{x}_n^T]\\
U_{yx} &= \frac{1}{N+1}\sum_{n=0}^{N}E[\mathbf{y}_n\mathbf{x}_n^T] &\qquad V_{x_1x} &= \frac{1}{N}\sum_{n=0}^{N-1}E[\mathbf{x}_{n+1}\mathbf{x}_n^T]\\
U_{yy} &= \frac{1}{N+1}\sum_{n=0}^{N}E[\mathbf{y}_n\mathbf{y}_n^T] &\qquad V_{x_1x_1} &= \frac{1}{N}\sum_{n=0}^{N-1}E[\mathbf{x}_{n+1}\mathbf{x}_{n+1}^T]
\end{aligned}
\tag{13.13.4}
$$

with $U_{xy}=U_{yx}^T$ and $V_{xx_1}=V_{x_1x}^T$. Then, the summed form of Eqs. (13.13.3) read:

$$
\begin{aligned}
&U_{yx} = CU_{xx}\\
&R = U_{yy}-CU_{xy}-U_{yx}C^T + CU_{xx}C^T = U_{yy}-CU_{xy}\\
&V_{x_1x} = AV_{xx}\\
&Q = V_{x_1x_1}-AV_{xx_1}-V_{x_1x}A^T + AV_{xx}A^T = V_{x_1x_1}-AV_{xx_1}
\end{aligned}
$$

which may be solved for the model parameters:

$$
\boxed{
\begin{aligned}
C &= U_{yx}U_{xx}^{-1}\\
R &= U_{yy}-CU_{xy}\\
A &= V_{x_1x}V_{xx}^{-1}\\
Q &= V_{x_1x_1}-AV_{xx_1}
\end{aligned}}
\tag{13.13.5}
$$

If A, C are known, then one can compute R, Q from the alternative quadratic expressions, which guarantee the (semi) positive-definiteness property of R, Q:

$$
\boxed{
\begin{aligned}
R &= U_{yy}-CU_{xy}-U_{yx}C^T + CU_{xx}C^T\\
Q &= V_{x_1x_1}-AV_{xx_1}-V_{x_1x}A^T + AV_{xx}A^T
\end{aligned}}
\tag{13.13.6}
$$

The above can be turned into an iterative estimation algorithm as follows. We start with the set of observations, $Y=\{\mathbf{y}_0,\mathbf{y}_1,\dots,\mathbf{y}_N\}$, and an initial choice for the model parameters, say,

$$
\theta^{\text{old}} = \{A^{\text{old}}, C^{\text{old}}, Q^{\text{old}}, R^{\text{old}}, \bar{\mathbf{x}}_0^{\text{old}}, \Sigma_0^{\text{old}}\}
$$

The Kalman smoother is run on the data Y using this set of model parameters, and the estimated smoothed state vectors $\hat{\mathbf{x}}_{n/N}$ and corresponding mean-square error covariances are computed. The expectation values in Eq. (13.13.4) are then replaced by the conditional ones based on Y and θ^{old}, that is,

$$
\begin{aligned}
\hat{U}_{xx} &= \frac{1}{N+1}\sum_{n=0}^{N}E[\mathbf{x}_n\mathbf{x}_n^T\,|\,Y] &\qquad \hat{V}_{xx} &= \frac{1}{N}\sum_{n=0}^{N-1}E[\mathbf{x}_n\mathbf{x}_n^T\,|\,Y]\\
\hat{U}_{yx} &= \frac{1}{N+1}\sum_{n=0}^{N}E[\mathbf{y}_n\mathbf{x}_n^T\,|\,Y] &\qquad \hat{V}_{x_1x} &= \frac{1}{N}\sum_{n=0}^{N-1}E[\mathbf{x}_{n+1}\mathbf{x}_n^T\,|\,Y]\\
\hat{U}_{yy} &= \frac{1}{N+1}\sum_{n=0}^{N}E[\mathbf{y}_n\mathbf{y}_n^T\,|\,Y] &\qquad \hat{V}_{x_1x_1} &= \frac{1}{N}\sum_{n=0}^{N-1}E[\mathbf{x}_{n+1}\mathbf{x}_{n+1}^T\,|\,Y]
\end{aligned}
\tag{13.13.7}
$$

Using the orthogonal decomposition for the smoothed states, we have:

$$
\begin{aligned}
&\mathbf{x}_n = \hat{\mathbf{x}}_{n/N}+\mathbf{e}_{n/N}\,, &&E[\hat{\mathbf{x}}_{n/N}\mathbf{e}_{n/N}^T]= 0\\
&\mathbf{x}_{n+1} = \hat{\mathbf{x}}_{n+1/N}+\mathbf{e}_{n+1/N}\,, &&E[\hat{\mathbf{x}}_{n+1/N}\mathbf{e}_{n+1/N}^T]= 0\,,\quad E[\hat{\mathbf{x}}_{n/N}\mathbf{e}_{n+1/N}^T]= 0
\end{aligned}
$$

which give:

$$
\begin{aligned}
&E[\mathbf{x}_n\mathbf{x}_n^T]= E[\hat{\mathbf{x}}_{n/N}\hat{\mathbf{x}}_{n/N}^T]+E[\mathbf{e}_{n/N}\mathbf{e}_{n/N}^T]\\
&E[\mathbf{x}_{n+1}\mathbf{x}_n^T]= E[\hat{\mathbf{x}}_{n+1/N}\hat{\mathbf{x}}_{n/N}^T]+E[\mathbf{e}_{n+1/N}\mathbf{e}_{n/N}^T]\\
&E[\mathbf{x}_{n+1}\mathbf{x}_{n+1}^T]= E[\hat{\mathbf{x}}_{n+1/N}\hat{\mathbf{x}}_{n+1/N}^T]+E[\mathbf{e}_{n+1/N}\mathbf{e}_{n+1/N}^T]
\end{aligned}
$$

Replacing these by the conditional expectations, we obtain:

$$
\begin{aligned}
E[\mathbf{x}_n\mathbf{x}_n^T\,|\,Y] &= E[\hat{\mathbf{x}}_{n/N}\hat{\mathbf{x}}_{n/N}^T\,|\,Y]+E[\mathbf{e}_{n/N}\mathbf{e}_{n/N}^T\,|\,Y] = \hat{\mathbf{x}}_{n/N}\hat{\mathbf{x}}_{n/N}^T + P_{n/N}\\
E[\mathbf{x}_{n+1}\mathbf{x}_n^T\,|\,Y] &= E[\hat{\mathbf{x}}_{n+1/N}\hat{\mathbf{x}}_{n/N}^T\,|\,Y]+E[\mathbf{e}_{n+1/N}\mathbf{e}_{n/N}^T\,|\,Y] = \hat{\mathbf{x}}_{n+1/N}\hat{\mathbf{x}}_{n/N}^T + V_{n+1,n}\\
E[\mathbf{x}_{n+1}\mathbf{x}_{n+1}^T\,|\,Y] &= E[\hat{\mathbf{x}}_{n+1/N}\hat{\mathbf{x}}_{n+1/N}^T\,|\,Y]+E[\mathbf{e}_{n+1/N}\mathbf{e}_{n+1/N}^T\,|\,Y] = \hat{\mathbf{x}}_{n+1/N}\hat{\mathbf{x}}_{n+1/N}^T + P_{n+1/N}
\end{aligned}
$$

where we set $V_{n+1,n} = E[\mathbf{e}_{n+1/N}\mathbf{e}_{n/N}^T\,|\,Y]$, given by Eq. (13.10.25). Similarly, we have:

$$
\begin{aligned}
E[\mathbf{y}_n\mathbf{x}_n^T\,|\,Y] &= \mathbf{y}_n\hat{\mathbf{x}}_{n/N}\\
E[\mathbf{y}_n\mathbf{y}_n^T\,|\,Y] &= \mathbf{y}_n\mathbf{y}_n^T
\end{aligned}
$$

Thus, we may replace Eqs. (13.13.4) by their estimated versions based on Y and θ^{old}:

$$
\boxed{\begin{aligned}
\hat{U}_{xx} &= \frac{1}{N+1}\sum_{n=0}^{N}\big[\hat{\mathbf{x}}_{n/N}\hat{\mathbf{x}}_{n/N}^T + P_{n/N}\big]\\
\hat{U}_{yx} &= \frac{1}{N+1}\sum_{n=0}^{N}\mathbf{y}_n\hat{\mathbf{x}}_{n/N}^T\\
\hat{U}_{yy} &= \frac{1}{N+1}\sum_{n=0}^{N}\mathbf{y}_n\mathbf{y}_n^T
\end{aligned}}
\quad
\boxed{\begin{aligned}
\hat{V}_{xx} &= \frac{1}{N}\sum_{n=0}^{N-1}\big[\hat{\mathbf{x}}_{n/N}\hat{\mathbf{x}}_{n/N}^T + P_{n/N}\big]\\
\hat{V}_{x_1x} &= \frac{1}{N}\sum_{n=0}^{N-1}\big[\hat{\mathbf{x}}_{n+1/N}\hat{\mathbf{x}}_{n/N}^T + V_{n+1,n}\big]\\
\hat{V}_{x_1x_1} &= \frac{1}{N}\sum_{n=0}^{N-1}\big[\hat{\mathbf{x}}_{n+1/N}\hat{\mathbf{x}}_{n+1/N}^T + P_{n+1/N}\big]
\end{aligned}}
\tag{13.13.8}
$$

Eqs. (13.13.5) can be used now to compute a new set of model parameters:

$$
\boxed{\begin{aligned}
C^{\text{new}} &= \hat{U}_{yx}\hat{U}_{xx}^{-1}\\
R^{\text{new}} &= \hat{U}_{yy} - C^{\text{new}}\hat{U}_{xy}\\
A^{\text{new}} &= \hat{V}_{x_1x}\hat{V}_{xx}^{-1}\\
Q^{\text{new}} &= \hat{V}_{x_1x_1} - A^{\text{new}}\hat{V}_{xx_1}
\end{aligned}}
\tag{13.13.9}
$$

or, if A, C are known, only Q, R are updated:

$$
\boxed{\begin{aligned}
R^{\text{new}} &= \hat{U}_{yy} - C\hat{U}_{xy} - \hat{U}_{yx}C^T + C\hat{U}_{xx}C^T\\
Q^{\text{new}} &= \hat{V}_{x_1x_1} - A\hat{V}_{xx_1} - \hat{V}_{x_1x}A^T + A\hat{V}_{xx}A^T
\end{aligned}}
\tag{13.13.10}
$$

The initial values $\bar{\mathbf{x}}_0, \Sigma_0$ are also estimated in the same way:

$$
\begin{aligned}
\bar{\mathbf{x}}_0 &= E[\mathbf{x}_0]\\
\Sigma_0 &= E[(\mathbf{x}_0-\bar{\mathbf{x}}_0)(\mathbf{x}_0-\bar{\mathbf{x}}_0)^T]
\end{aligned}
\quad\Rightarrow\quad
\begin{aligned}
\bar{\mathbf{x}}_0^{\text{new}} &= E[\mathbf{x}_0\,|\,Y] = \hat{\mathbf{x}}_{0/N}\\
\hat{\Sigma}_0^{\text{new}} &= E[(\mathbf{x}_0-\bar{\mathbf{x}}_0)(\mathbf{x}_0-\bar{\mathbf{x}}_0)^T\,|\,Y] = P_{0/N}\,,\quad \text{or,}
\end{aligned}
$$

$$
\boxed{\bar{\mathbf{x}}_0^{\text{new}} = \hat{\mathbf{x}}_{0/N}\,,\quad \hat{\Sigma}_0^{\text{new}} = P_{0/N}}
\tag{13.13.11}
$$

Thus, Eqs. (13.13.9)–(13.13.11) define a new set of model parameters:

$$
\theta^{\text{new}} = \{A^{\text{new}}, C^{\text{new}}, Q^{\text{new}}, R^{\text{new}}, \bar{\mathbf{x}}_0^{\text{new}}, \Sigma_0^{\text{new}}\}
$$

and the iteration can be repeated until convergence by monitoring the value of the likelihood function $L_\theta(Y)$. Of course, all or any subset of the model parameters θ may be iterated through this algorithm.

The outputs `Xs`, `Ps`, `V` of the MATLAB function `ksmooth` and can be used to efficiently calculate the quantities of Eq. (13.13.8). For example, we have in MATLAB notation:

$$
\begin{aligned}
\hat{U}_{xx} &= \frac{1}{N+1}\sum_{n=0}^{N}\big[\hat{\mathbf{x}}_{n/N}\hat{\mathbf{x}}_{n/N}^T + P_{n/N}\big] = \frac{1}{N+1}\big(X_s \ast X_s' + \text{sum}(P_s,3)\big)\\
\hat{U}_{yx} &= \frac{1}{N+1}\sum_{n=0}^{N}\mathbf{y}_n\hat{\mathbf{x}}_{n/N}^T = \frac{1}{N+1}(Y \ast X_s')\\
\hat{U}_{yy} &= \frac{1}{N+1}\sum_{n=0}^{N}\mathbf{y}_n\mathbf{y}_n^T = \frac{1}{N+1}(Y \ast Y')\\
\hat{V}_{xx} &= \frac{1}{N}\sum_{n=0}^{N-1}\big[\hat{\mathbf{x}}_{n/N}\hat{\mathbf{x}}_{n/N}^T + P_{n/N}\big] = \frac{1}{N}\big(X_{s0} \ast X_{s0}' + \text{sum}(P_{s0},3)\big)\\
\hat{V}_{x_1x} &= \frac{1}{N}\sum_{n=0}^{N-1}\big[\hat{\mathbf{x}}_{n+1/N}\hat{\mathbf{x}}_{n/N}^T + V_{n+1,n}\big] = \frac{1}{N}\big(X_{s1} \ast X_{s0}' + \text{sum}(V_0,3)\big)\\
\hat{V}_{x_1x_1} &= \frac{1}{N}\sum_{n=0}^{N-1}\big[\hat{\mathbf{x}}_{n+1/N}\hat{\mathbf{x}}_{n+1/N}^T + P_{n+1/N}\big] = \frac{1}{N}\big(X_{s1} \ast X_{s1}' + \text{sum}(P_{s1},3)\big)
\end{aligned}
$$

where we used the definitions:

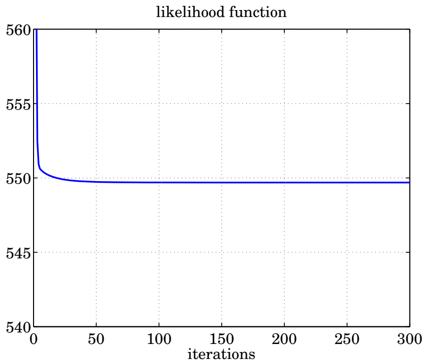
$$
\begin{aligned}
Y &= [\mathbf{y}_0, \mathbf{y}_1, \dots, \mathbf{y}_N]\\
X_s &= [\hat{\mathbf{x}}_{0/N},\, \hat{\mathbf{x}}_{1/N},\, \dots,\, \hat{\mathbf{x}}_{N-1/N},\, \hat{\mathbf{x}}_{N/N}] = [X_{s0},\, \hat{\mathbf{x}}_{N/N}] = [\hat{\mathbf{x}}_{0/N},\, X_{s1}]\\
X_{s0} &= [\hat{\mathbf{x}}_{0/N},\, \hat{\mathbf{x}}_{1/N},\, \dots,\, \hat{\mathbf{x}}_{N-1/N}]\\
X_{s1} &= [\hat{\mathbf{x}}_{1/N},\, \dots,\, \hat{\mathbf{x}}_{N-1/N},\, \hat{\mathbf{x}}_{N/N}]
\end{aligned}
\tag{13.13.12}
$$

and P_{s0}, P_{s1}, V_0 are sub-arrays of P_s and V matching the indexing of (13.13.12).

Example 13.13.1: We apply the estimation algorithm to the Nile River data of Example 13.12.1 by keeping A, C fixed, and only estimating $Q, R, \bar{x}_0, \Sigma_0$. We allow 300 iterations, with starting values chosen to be the same as those of Example 13.12.1. The following MATLAB code illustrates the implementation of the algorithm:

```
K = 300; N = length(y)-1;              % use same y as in Example 15.12.1

x0 = 0; S0 = 1e7; Q = 1; R = 1;        % initial values for the EM iteration
```

```
for i=1:K,                                        % EM iteration
   [L(i),Xs,Ps,V] = ksmooth(A,C,Q,R,y,x0,S0);    % E-step of the algorithm

   Uxx = (Xs*Xs' + sum(Ps,3))/(N+1);             % construct U,V matrices
   Uyx = (y*Xs')/(N+1);
   Uyy = (y*y')/(N+1);
   Vxx = (Xs(:,1:end-1)*Xs(:,1:end-1)' + sum(Ps(:,:,1:end-1),3))/N;
   V1x = (Xs(:,2:end)*Xs(:,1:end-1)' + sum(V(:,:,1:end-1),3))/N;
   V11 = (Xs(:,2:end)*Xs(:,2:end)' + sum(Ps(:,:,2:end),3))/N;

   R = Uyy - C*Uyx' - Uyx*C' + C*Uxx*C';         % M-step of the algorithm
   Q = V11 - A*V1x' - V1x*A' + A*Vxx*A';

   % x0 = Xs(:,1);        % uncomment to also estimate x0,S0
   % S0 = Ps(:,:,1);
end

k = 0:K-1; figure; plot(k,L);                     % plot likelihood function
```



The left graph shows the negative-log-likelihood function of Eq. (13.12.2) plotted versus iteration number for the case when only Q, R are being estimated, with $A, C, \bar{x}_0, \Sigma_0$ held fixed. The right graphs shows the case when $Q, R, \bar{x}_0, \Sigma_0$ are estimated. The estimated quantities at the last iteration are:

$$Q = 1468.5\,, \quad R = 15099.0$$

$$Q = 1294.7\,, \quad R = 15252.4\,, \quad \bar{x}_0 = 1118.4\,, \quad \Sigma_0 = 0.6$$

The results for the first case are to be compared with those of Example 13.12.1, that is, $Q = 1468.5$, $R = 15099.7$. The second case is different because four model parameters are being iterated, which lead to a smaller value of the likelihood function.

We note that in both cases, the likelihood function converges very fast initially, and then reaches a slowly-decreasing plateau. One could perhaps stop the iteration as soon the plateau is reached and use the slightly suboptimal estimates. □

The above estimation method can be extended to handle the case of a rank-defective noise covariance matrix Q. This arises for the following type of state-space model:

$$\begin{aligned} \mathbf{x}_{n+1} &= A\mathbf{x}_n + B\mathbf{w}_n \\ \mathbf{y}_n &= C\mathbf{x}_n + \mathbf{v}_n \end{aligned} \qquad (13.13.13)$$

where $\mathbf{x}_n$ is p-dimensional and B is a $p\times q$ matrix with $q < p$, assumed to have full rank (i.e., q), and $\mathbf{w}_n$ is a q-dimensional white-noise signal with non-singular $q\times q$ covariance matrix Q_w. The corresponding Q that appears in the Kalman filtering algorithm is then $Q = BQ_wB^T$, and has rank q.

Assuming a known B, the iterative algorithm can be modified to estimate Q_w. Given an initial Q_w^{old}, we calculate $Q^{\text{old}} = BQ_w^{\text{old}}B^T$, and then carry out the usual iteration step to determine Q^{new} from Eq. (13.13.9) or (13.13.10). To find Q_w^{new}, we solve the equation $BQ_w^{\text{new}}B^T = Q^{\text{new}}$ using the pseudoinverse of B, that is, $B^+ = (B^TB)^{-1}B^T$,

$$Q_w^{\text{new}} = B^+Q^{\text{new}}B^{+T} \qquad (13.13.14)$$

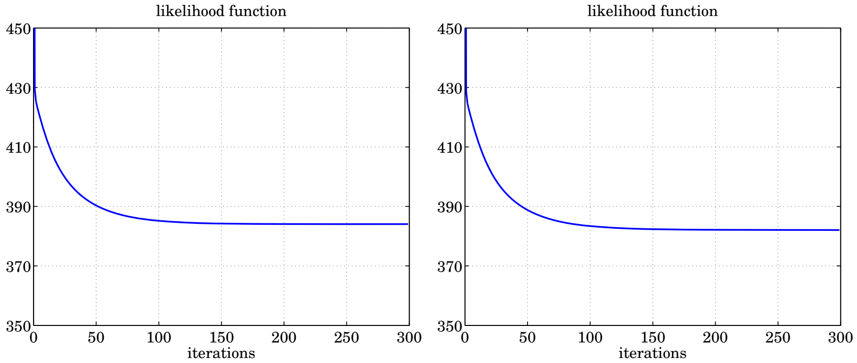
Example 13.13.2: Consider the α–β tracking model discussed in Example 13.12.2 and defined by the state-space model:

$$\begin{bmatrix} x_{n+1} \\ \dot{x}_{n+1} \end{bmatrix} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_n \\ \dot{x}_n \end{bmatrix} + \begin{bmatrix} T^2/2 \\ T \end{bmatrix} a_n\,, \quad y_n = [1, 0] \begin{bmatrix} x_n \\ \dot{x}_n \end{bmatrix} + v_n$$

with model matrices:

$$A = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix}, \quad C = [1, 0]\,, \quad B = \begin{bmatrix} T^2/2 \\ T \end{bmatrix}, \quad Q_w = \sigma_a^2\,, \quad Q = BQ_wB^T\,, \quad R = \sigma_v^2$$

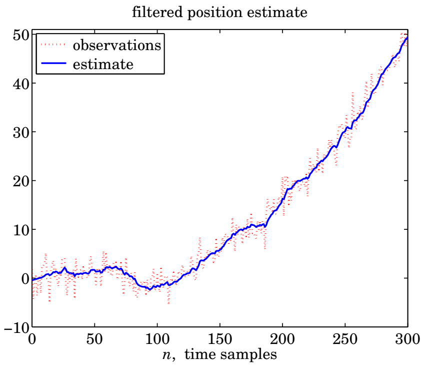
The following figure shows the likelihood as a function of iteration number for the two cases of estimating only Q_w, R (left graph), or, $Q_w, R, \bar{\mathbf{x}}_0, \Sigma_0$ (right graph), using 300 iterations. The data y_n are generated by exactly the same code (not repeated here) as in Example 13.12.2. We note again the rapid initial decrease, followed by a plateau.

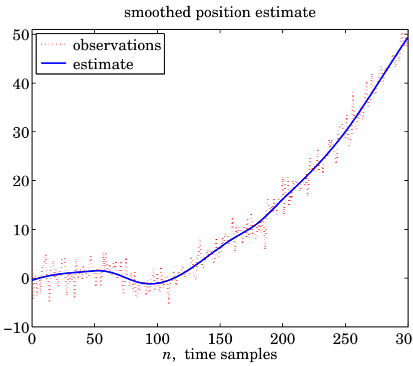


The MATLAB code used to generate the right graph is shown below.

```
K = 300; N = length(y)-1; t =0:N;        % data y generated as in Ex. 15.12.2

B = [T^2/2; T]; Binv = pinv(B);

Qw = 0.1; R = 1;                         % initial values for the iteration
x0 = [0; 0.1]; S0 = 0.1 * eye(2);


for i=1:K,
   Q = B*Qw*B’;                                  % construct Q_old
   [L(i),Xs,Ps,V] = ksmooth(A,C,Q,R,y,x0,S0);

   Uxx = (Xs*Xs’ + sum(Ps,3))/(N+1);             % compute U,V matrices
   Uyx = (y*Xs’)/(N+1);
   Uyy = (y*y’)/(N+1);
   Vxx = (Xs(:,1:end-1)*Xs(:,1:end-1)’ + sum(Ps(:,:,1:end-1),3))/N;
   V1x = (Xs(:,2:end)*Xs(:,1:end-1)’ + sum(V(:,:,1:end-1),3))/N;
   V11 = (Xs(:,2:end)*Xs(:,2:end)’ + sum(Ps(:,:,2:end),3))/N;

   R = Uyy - C*Uyx’ - Uyx*C’ + C*Uxx*C’;         % construct R_new
   Q = V11 - A*V1x’ - V1x*A’ + A*Vxx*A’;         % construct Q_new
   Qw = Binv * Q * Binv’;                        % construct Q_w_new

   x0 = Xs(:,1);              % comment out to skip estimation
   S0 = Ps(:,:,1);
end

k = 0:K-1; figure; plot(k,L);

[Lmin,X,P,Xf,Pf] = kfilt(A,C,Q,R,y,x0,S0);       % use estimated Q,R,x0,S0
[Lmin,Xs,Ps] = ksmooth(A,C,Q,R,y,x0,S0);

figure; plot(t,y,’:’, t,Xf(1,:),’-’);
figure; plot(t,y,’:’, t,Xs(1,:),’-’);
```

Once the parameters $Q_w, R, \bar{\mathbf{x}}_0, \Sigma_0$ have been estimated, the Kalman filter and smoother are run to determine the filtered and smoothed state estimates, $\hat{\mathbf{x}}_{n/n}$ and $\hat{\mathbf{x}}_{n/N}$. Their position components are shown in the figure below.





The estimated parameters in Example 13.12.2 were $Q_w = (0.0199)^2$ and $R = (2.0237)^2$,

compared with the theoretical values that were simulated $Q_w = (0.02)^2$ and $R = (2.0)^2$. By comparison, the present method gives the estimates: $Q_w = (0.0201)^2$ and $R = (2.0235)^2$ if only Q_w, R are estimated, and $Q_w = (0.0193)^2$ and $R = (2.0221)^2$, if all four parameters $Q_w, R, \bar{\mathbf{x}}_0, \Sigma_0$ are estimated. In the latter case, the estimated initial values were:

$$\bar{\mathbf{x}}_0 = \begin{bmatrix} -0.4468 \\ 0.0791 \end{bmatrix}, \quad \Sigma_0 = \begin{bmatrix} 0.00199 & -0.00015 \\ -0.00015 & 0.00003 \end{bmatrix}$$

Connection to the EM Algorithm

Consider a realization of the states and observations of the state-space model (13.13.1),

$$X = [\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_N], \quad Y = [\mathbf{y}_0, \mathbf{y}_1, \dots, \mathbf{y}_N] \tag{13.13.15}$$

Ideally, the maximum likelihood method would maximize the joint probability density $p_\theta(X,Y)$, or equivalently the log density, $\ln p_\theta(X,Y)$, with respect to the model parameters θ. However, in practice only the observations Y are available and not the states X, and one chooses to maximize instead the marginal density $p_\theta(Y)$:

$$p_\theta(Y) = \int p_\theta(X,Y)\,dX \tag{13.13.16}$$

The EM algorithm for maximizing (13.13.16) is based on the following inequality:

$$\int p(X|Y,\theta^{\text{old}})\,\ln\left[\frac{p_\theta(X,Y)}{p(X|Y,\theta^{\text{old}})}\right]dX \le \ln p_\theta(Y) \tag{13.13.17}$$

where $p(X|Y,\theta^{\text{old}})$ is the conditional density of X given Y and the parameter set θ^{old}. Eq. (13.13.17) is a consequence of Jensen’s inequality, which implies:

$$\begin{aligned}\int p(X|Y,\theta^{\text{old}})\,\ln\left[\frac{p_\theta(X,Y)}{p(X|Y,\theta^{\text{old}})}\right]dX &\le \ln\left[\int p(X|Y,\theta^{\text{old}})\,\frac{p_\theta(X,Y)}{p(X|Y,\theta^{\text{old}})}\,dX\right] \\ &= \ln\left[\int p_\theta(X,Y)\,dX\right] = \ln p_\theta(Y)\end{aligned}$$

It follows from Bayes’s rule, $p_\theta(X,Y) = p_\theta(X|Y)p_\theta(Y)$, that (13.13.17) becomes an equality at $\theta = \theta^{\text{old}}$. Eq. (13.13.17) can now be re-written in the form:

$$\mathcal{Q}(\theta,\theta^{\text{old}}) + I(\theta^{\text{old}}) \le \ln p_\theta(Y) \tag{13.13.18}$$

where we defined the following conditional expectation and entropy with respect to the conditional density $p(X|Y,\theta^{\text{old}})$:

$$\begin{aligned}\mathcal{Q}(\theta,\theta^{\text{old}}) &= \int p(X|Y,\theta^{\text{old}})\,\ln p_\theta(X,Y)\,dX \equiv E\big[\ln p_\theta(X,Y)\,|\,Y,\theta^{\text{old}}\big] \\ I(\theta^{\text{old}}) &= -\int p(X|Y,\theta^{\text{old}})\,\ln p(X|Y,\theta^{\text{old}})\,dX\end{aligned} \tag{13.13.19}$$

We note that $I(\theta^{\text{old}})$ is a positive quantity. The EM algorithm chooses that θ that maximizes the left bound in (13.13.17). More precisely, the EM algorithm consists of the repetition the the following two steps:

a. The expectation step (E-step), in which the conditional expectation $\mathcal{Q}(\theta, \theta^{\text{old}})$ is calculated.

b. The maximization step (M-step), in which $\mathcal{Q}(\theta, \theta^{\text{old}})$ is maximized resulting into a new θ:

$$\theta^{\text{new}} = \arg\max_{\theta} \mathcal{Q}(\theta, \theta^{\text{old}})$$

The convergence properties of the EM algorithm have been discussed in the literature [905,906], and its application to state-space models, in [909–911] and others.

The advantage of the EM algorithm is that, although $p_\theta(Y)$ is hard to calculate and maximize, the joint density $p_\theta(X,Y)$ is straightforward to calculate before the E-step is carried out. Indeed, successively applying Bayes's rule we have:

$$p_\theta(X,Y) = p(\mathbf{x}_0)\prod_{n=0}^{N-1} p(\mathbf{x}_{n+1}|\mathbf{x}_n)\prod_{n=0}^{N} p(\mathbf{y}_n|\mathbf{x}_n)$$

Assuming Gaussian statistics and full-rank covariance matrices Q, R, Σ_0, we find for the negative-log-likelihood function up to a constant:

$$\begin{aligned}
-\ln p_\theta(X,Y) &= \frac{1}{2}\Big[\ln\det\Sigma_0 + (\mathbf{x}_0 - \bar{\mathbf{x}}_0)^T\Sigma_0^{-1}(\mathbf{x}_0 - \bar{\mathbf{x}}_0)\Big] \\
&\quad + \frac{1}{2}\sum_{n=0}^{N-1}\Big[\ln\det Q + (\mathbf{x}_{n+1} - A\mathbf{x}_n)^T Q^{-1}(\mathbf{x}_{n+1} - A\mathbf{x}_n)\Big] \\
&\quad + \frac{1}{2}\sum_{n=0}^{N}\Big[\ln\det R + (\mathbf{y}_n - C\mathbf{x}_n)^T R^{-1}(\mathbf{y}_n - C\mathbf{x}_n)\Big]
\end{aligned} \tag{13.13.20}$$

and using the identity, $\ln\det Q = \text{tr}\ln Q$, we obtain:

$$\begin{aligned}
-\ln p_\theta(X,Y) &= \frac{1}{2}\,\text{tr}\Big[\ln\Sigma_0 + \Sigma_0^{-1}(\mathbf{x}_0 - \bar{\mathbf{x}}_0)(\mathbf{x}_0 - \bar{\mathbf{x}}_0)^T\Big] \\
&\quad + \frac{1}{2}\,\text{tr}\Big[\ln Q + Q^{-1}\frac{1}{N}\sum_{n=0}^{N-1}(\mathbf{x}_{n+1} - A\mathbf{x}_n)(\mathbf{x}_{n+1} - A\mathbf{x}_n)^T\Big] \\
&\quad + \frac{1}{2}\,\text{tr}\Big[\ln R + R^{-1}\frac{1}{N+1}\sum_{n=0}^{N}(\mathbf{y}_n - C\mathbf{x}_n)(\mathbf{y}_n - C\mathbf{x}_n)^T\Big]
\end{aligned} \tag{13.13.21}$$

Taking conditional expectations with respect to $p(X|Y,\theta^{\text{old}})$, we have:

$$\begin{aligned}
-\mathcal{Q}(\theta, \theta^{\text{old}}) &= \frac{1}{2}\,\text{tr}\Big[\ln\Sigma_0 + \Sigma_0^{-1}E\big[(\mathbf{x}_0 - \bar{\mathbf{x}}_0)(\mathbf{x}_0 - \bar{\mathbf{x}}_0)^T\,|\,Y,\theta^{\text{old}}\big]\Big] \\
&\quad + \frac{1}{2}\,\text{tr}\Big[\ln Q + Q^{-1}\frac{1}{N}\sum_{n=0}^{N-1}E\big[(\mathbf{x}_{n+1} - A\mathbf{x}_n)(\mathbf{x}_{n+1} - A\mathbf{x}_n)^T\,|\,Y,\theta^{\text{old}}\big]\Big] \\
&\quad + \frac{1}{2}\,\text{tr}\Big[\ln R + R^{-1}\frac{1}{N+1}\sum_{n=0}^{N}E\big[(\mathbf{y}_n - C\mathbf{x}_n)(\mathbf{y}_n - C\mathbf{x}_n)^T\,|\,Y,\theta^{\text{old}}\big]\Big]
\end{aligned}$$

which can be written in terms of the definitions (13.13.8),

$$\begin{aligned}
-\mathcal{Q}(\theta, \theta^{\text{old}}) &= \frac{1}{2}\,\text{tr}\Big[\ln\Sigma_0 + \Sigma_0^{-1}\big[(\bar{\mathbf{x}}_0 - \hat{\mathbf{x}}_{0/N})(\bar{\mathbf{x}}_0 - \hat{\mathbf{x}}_{0/N})^T + P_{0/N}\big]\Big] \\
&\quad + \frac{1}{2}\,\text{tr}\Big[\ln Q + Q^{-1}\big[\hat{V}_{x_1x_1} - A\hat{V}_{xx_1} - \hat{V}_{x_1x}A^T + A\hat{V}_{xx}A^T\big]\Big] \\
&\quad + \frac{1}{2}\,\text{tr}\Big[\ln R + R^{-1}\big[\hat{U}_{yy} - C\hat{U}_{xy} - \hat{U}_{yx}C^T + C\hat{U}_{xx}C^T\big]\Big]
\end{aligned} \tag{13.13.22}$$

This represents the E-step of the algorithm. The M-step leads to the same equations as (13.13.9)–(13.13.11). Indeed, the minimization of $-\mathcal{Q}(\theta, \theta^{\text{old}})$ with respect to $\bar{\mathbf{x}}_0$ gives $\bar{\mathbf{x}}_0 = \hat{\mathbf{x}}_{0/N}$. Similarly, the first-order differentials with respect to A and C are:

$$\begin{aligned}
-d\mathcal{Q} &= \frac{1}{2}\,\text{tr}\Big[Q^{-1}\big[(A\hat{V}_{xx} - \hat{V}_{x_1x})dA^T + dA(\hat{V}_{xx}A^T - \hat{V}_{xx_1})\big]\Big] \\
-d\mathcal{Q} &= \frac{1}{2}\,\text{tr}\Big[R^{-1}\big[(C\hat{U}_{xx} - \hat{U}_{yx})dC^T + dC(\hat{U}_{xx} - \hat{U}_{xy})\big]\Big]
\end{aligned}$$

Since the variations dA, dC are arbitrary, the vanishing of their coefficients leads to the solutions given in Eq. (13.13.9). The minimizations of $-\mathcal{Q}(\theta, \theta^{\text{old}})$ with respect to Q, R, Σ_0 are similar and have the generic form of finding R that minimizes:

$$F = \text{tr}\big[\ln R + R^{-1}U\big]$$

If all entries of R are treated as independent variables, then the vanishing of the first-order differential of F gives:

$$dF = \text{tr}\big[R^{-1}dR - R^{-1}dR\,R^{-1}U\big] = 0 \quad\Rightarrow\quad R = U$$

where we used $d(R^{-1}) = -R^{-1}dR\,R^{-1}$. In our case, Q, R, Σ_0 are symmetric matrices and only their lower (or upper) triangular parts may be regarded as independent variables. However, the answer turns out to be the same as if all matrix elements were treated as independent. Thus, the minimization of $-\mathcal{Q}(\theta, \theta^{\text{old}})$ gives the same answers as those of Eqs. (13.13.9)–(13.13.11).

14

Spectrum Estimation and Array Processing

14.1 Spectrum Estimation by Autoregressive Modeling

When a block of signal samples is available, it may be too short to provide enough frequency resolution in the periodogram spectrum. Often, it may not even be correct to extend the length by collecting more samples, since this might come into conflict with the stationarity of the segment. In cases such as these, parametric representation of the spectra by means of autoregressive models can provide much better frequency resolution than the classical periodogram method [926]. This approach was discussed briefly in Sec. 1.13.

The spectrum estimation procedure is as follows: First, the given data segment $\{y_0, y_1, \dots, y_{N-1}\}$ is subjected to one of the analysis methods discussed in Sec. 12.12 to extract estimates of the LPC model parameters $\{a_1, a_2, \dots, a_M; E_M\}$. The choice of the order M is an important consideration. There are a number of criteria for model order selection [926], but there is no single one that works well under all circumstances. In fact, selecting the right order M is more often an art than science. As an example, we mention Akaike's final prediction error (FPE) criterion which selects the M that minimizes the quantity

$$E_M \cdot \frac{N+M+1}{N-M-1} = \min$$

where E_M is the estimate of the mean-square prediction error for the Mth order predictor, and N is the length of the sequence y_n. As M increases, the factor E_M decreases and the second factor increases, thus, there is a minimum value. Then, the spectrum estimate is given by

$$S_{AR}(\omega) = \frac{E_M}{\big|A_M(\omega)\big|^2} = \frac{E_M}{\big|1 + a_1 e^{-j\omega} + a_2 e^{-2j\omega} + \cdots + a_M e^{-Mj\omega}\big|^2} \tag{14.1.1}$$

Note that this would be the exact spectrum if y_n were autoregressive with the above set of model parameters. Generally, spectra that have a few dominant spectral peaks can be modeled quite successfully by such all-pole autoregressive models. One can also fit the given block of data to more general ARMA models. The decision to model a spectrum by ARMA, AR, or MA models should ultimately depend on some prior information regarding the physics of the process y_n. The reader is referred to the exhaustive review article of Kay and Marple [926], and to [1076–1080]

Next, we compare by means of a simulation example the classical periodogram method, the Yule-Walker method, and Burg's method of computing spectrum estimates. Generally, the rule of thumb to follow is that Burg's method should work better than the other methods on short records of data, and that all three methods tend to improve as the data record becomes longer. For our simulation example, we chose a fourth order autoregressive model characterized by two very sharp peaks in its spectrum. The signal generator for the sequence y_n was

$$y_n + a_1 y_{n-1} + a_2 y_{n-2} + a_3 y_{n-3} + a_4 y_{n-4} = \epsilon_n$$

where ϵ_n was zero-mean, unit-variance, white noise. The prediction-error filter $A(z)$ was defined in terms of its four zeros:

$$\begin{aligned} A(z) &= 1 + a_1 z^{-1} + a_2 z^{-2} + a_3 z^{-3} + a_4 z^{-4} \\ &= (1 - z_1 z^{-1})(1 - z_1^* z^{-1})(1 - z_2 z^{-1})(1 - z_2^* z^{-1}) \end{aligned}$$

where the zeros were chosen as

$$z_1 = 0.99\exp(0.2\pi j)\,, \quad z_2 = 0.99\exp(0.4\pi j)$$

This gives for the filter coefficients

$$a_1 = -2.2137\,, \quad a_2 = 2.9403\,, \quad a_3 = -2.1697\,, \quad a_4 = 0.9606$$

The exact spectrum is given by Eq. (14.1.1) with $E_4 = \sigma_\epsilon^2 = 1$. Since the two zeros z_1 and z_2, are near the unit circle, the spectrum will have two very sharp peaks at the normalized frequencies

$$\omega_1 = 0.2\pi\,, \quad \omega_2 = 0.4\pi \quad \text{[radians/sample]}$$

Using the above difference equation and a realization of ϵ_n, a sequence of length 20 of y_n samples was generated (the filter was run for a while until its transients died out and stationarity of y_n was reached). The same set of 20 samples was used to compute the ordinary periodogram spectrum and the autoregressive spectra using the Yule-Walker and Burg methods of extracting the model parameters. Then, the length of the data sequence y_n was increased to 100 and the periodogram, Yule-Walker, and Burg spectra were computed again. Fig. 14.1.1 shows the periodogram spectra for the two signal lengths of 20 and 100 samples. Fig. 14.1.2 show the Yule-Walker spectra, and Fig. 14.1.3, the Burg spectra.

The lack of sufficient resolution of both the periodogram and the Yule-Walker spectrum estimates for the shorter data record can be attributed to the windowing of the signal y_n. But as the length increases the effects of windowing become less pronounced and both methods improve. Burg's method is remarkable in that it works very well even on the basis of very short data records. The Burg spectral estimate is sometimes called the "maximum entropy" spectral estimate. The connection to entropy concepts is discussed in the above references.

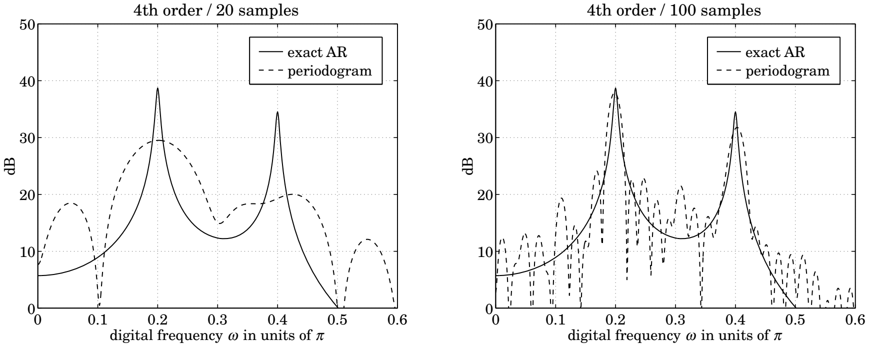


Fig. 14.1.1 Periodogram spectra based on 20 and 100 samples.

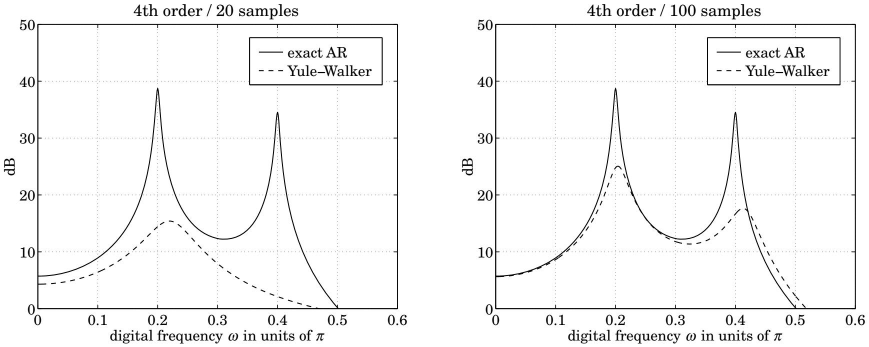


Fig. 14.1.2 Yule-Walker spectra based on 20 and 100 samples.

14.2 Spectral Analysis of Sinusoids in Noise

One of the most important signal processing problems is the estimation of the frequencies and amplitudes of sinusoidal signals buried in additive noise. In addition to its practical importance, this problem has served as the testing ground for all spectrum estimation techniques, new or old. In this section we discuss four approaches to this problem: (1) the classical method, based on the Fourier transform of the windowed autocorrelation; (2) the maximum entropy method, based on the autoregressive modeling of the spectrum; (3) the maximum likelihood, or minimum energy, method; and (4) Pisarenko's method of harmonic retrieval which offers the highest resolution.

Consider a signal consisting of L complex sinusoids with random phases in additive noise:

$$y_n = v_n + \sum_{i=1}^{L} A_i e^{j\omega_i n + j\phi_i} \tag{14.2.1}$$

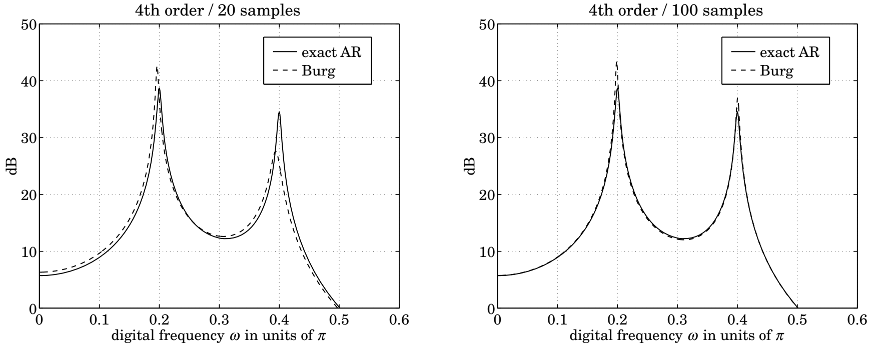


Fig. 14.1.3 Burg spectra based on 20 and 100 samples.

where the phases ϕ_i are uniformly distributed and independent of each other, and v_n is zero-mean white noise of variance σ_v^2, assumed to be independent of the phases ϕ_i:

$$E[v_n^* v_m] = \sigma_v^2 \delta_{nm}\,, \quad E[\phi_i v_n] = 0 \tag{14.2.2}$$

Under these assumptions, the autocorrelation of y_n is easily found to be

$$R(k) = E[y_{n+k} y_n^*] = \sigma_v^2 \delta(k) + \sum_{i=1}^{L} P_i e^{j\omega_i k} \tag{14.2.3}$$

where P_i denotes the power of the ith sinusoid; that is, $P_i = |A_i|^2$. The basic problem is to extract the set of frequencies $\{\omega_1, \omega_2, \dots, \omega_L\}$ and powers $\{P_1, P_2, \dots, P_L\}$ by appropriate processing a segment of signal samples y_n. The theoretical power spectrum is a line spectrum superimposed on a flat white-noise background:

$$S(\omega) = \sigma_v^2 + \sum_{i=1}^{L} P_i \, 2\pi\delta(\omega - \omega_i) \tag{14.2.4}$$

which is obtained by Fourier transforming Eq. (14.2.3):

$$S(\omega) = \sum_{k=-\infty}^{\infty} R(k) e^{-j\omega k} \tag{14.2.5}$$

Given a finite set of autocorrelation lags $\{R(0), R(1), \dots, R(M)\}$, the classical spectrum analysis method consists of windowing these lags by an appropriate window and then computing the sum (14.2.5), truncated to $-M \le k \le M$. We will use the triangular or Bartlett window which corresponds to the mean value of the ordinary periodogram spectrum [12]. This window is defined by

$$w_B(k) = \begin{cases} \dfrac{M+1-|k|}{M+1}\,, & \text{if } \; -M \le k \le M \\ 0\,, & \text{otherwise} \end{cases}$$

Replacing $R(k)$ by $w_B(k)R(k)$ in Eq. (14.2.5), we obtain the classical Bartlett spectrum estimate:

$$\hat{S}_B(\omega)=\sum_{k=-M}^{M} w_B(k)R(k)e^{-j\omega k} \tag{14.2.6}$$

We chose the Bartlett window because this expression can be written in a compact matrix form by introducing the $(M+1)$-dimensional phase vector

$$\mathbf{s}_\omega=\begin{bmatrix}1\\ e^{j\omega}\\ e^{2j\omega}\\ \vdots\\ e^{Mj\omega}\end{bmatrix}$$

and the $(M+1)\times(M+1)$ autocorrelation matrix R, defined as

$$R_{km}=R(k-m)=\sigma_v^2\delta(k-m)+\sum_{i=1}^{L}P_i e^{j\omega_i(k-m)}\,,\quad 0\le k,m\le M$$

Ignoring the $1/(M+1)$ scale factor arising from the definition of the Bartlett window, we may write Eq. (14.2.6) as

$$\hat{S}_B(\omega)=\mathbf{s}_\omega^\dagger R\mathbf{s}_\omega \qquad \text{(classical Bartlett spectrum)} \tag{14.2.7}$$

The autocorrelation matrix R of the sinusoids can also be written in terms of the phasing vectors as

$$R=\sigma_v^2 I+\sum_{i=1}^{L}P_i\,\mathbf{s}_{\omega_i}\mathbf{s}_{\omega_i}^\dagger \tag{14.2.8}$$

where I is the $(M+1)\times(M+1)$ identity matrix. It can be written even more compactly by introducing the $L\times L$ diagonal power matrix, and the $(M+1)\times L$ sinusoid matrix

$$P=\text{diag}\{P_1,P_2,\dots,P_L\}\,,\quad S=[\mathbf{s}_{\omega_1},\mathbf{s}_{\omega_2},\dots,\mathbf{s}_{\omega_L}]$$

Then, Eq. (14.2.8) becomes

$$R=\sigma_v^2 I+SPS^\dagger \tag{14.2.9}$$

Inserting Eq. (14.2.8) into (14.2.7) we find

$$\hat{S}_B(\omega)=\sigma_v^2\,\mathbf{s}_\omega^\dagger\mathbf{s}_\omega+\sum_{i=1}^{L}P_i\,\mathbf{s}_\omega^\dagger\mathbf{s}_{\omega_i}\mathbf{s}_{\omega_i}^\dagger\mathbf{s}_\omega$$

Defining the function

$$W(\omega)=\sum_{m=0}^{M}e^{-j\omega m}=\frac{1-e^{-j\omega(M+1)}}{1-e^{-j\omega}}=\frac{\sin\left(\dfrac{\omega(M+1)}{2}\right)}{\sin\left(\dfrac{\omega}{2}\right)}e^{-j\omega M/2} \tag{14.2.10}$$

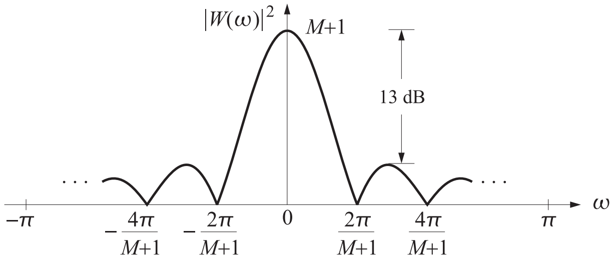
we note that

$$\mathbf{s}_\omega^\dagger\mathbf{s}_{\omega_i}=W(\omega-\omega_i)\quad\text{and}\quad \mathbf{s}_\omega^\dagger\mathbf{s}_\omega=W(0)=M+1$$

Then, in this notation, the Bartlett spectrum (14.2.7) becomes

$$\hat{S}_B(\omega)=\sigma_v^2(M+1)+\sum_{i=1}^{L}P_i\,\big|W(\omega-\omega_i)\big|^2 \tag{14.2.11}$$

The effect of $W(\omega-\omega_i)$ is to smear each spectral line $\delta(\omega-\omega_i)$ of the true spectrum. If the frequencies ω_i are too close to each other the smeared peaks will tend to overlap with a resulting loss of resolution. The function $W(\omega)$ is the Fourier transform of the rectangular window and is depicted below:



It has an effective resolution width of $\Delta\omega=2\pi/(M+1)$. For fairly large Ms, the first side lobe is about 13 dB down from the main lobe. As M increases, the main lobe becomes higher and thinner, resembling more and more a delta function, which improves the frequency resolution capability of this estimate.

Next, we derive a closed form expression [1085] for the AR, or maximum entropy, spectral estimate. It is given by Eq. (14.1.1) and is obtained by fitting an order-M autoregressive model to the autocorrelation lags $\{R(0),R(1),\dots,R(M)\}$. This can be done for any desired value of M. Autoregressive spectrum estimates generally work well in modeling “peaked” or resonant spectra; therefore, it is expected that they will work in this case, too. However, it should be kept in mind that AR models are not really appropriate for such sinusoidal signals. Indeed, AR models are characterized by all-pole stable filters that always result in autocorrelation functions $R(k)$ which decay exponentially with the lag k; whereas Eq. (14.2.3) is persistent in k and never decays.

As a rule, AR modeling of sinusoidal spectra works very well as long as the signal to noise ratios (SNRs) are fairly high. Pisarenko’s method, to be discussed later, provides unbiased frequency estimates *regardless* of the SNRs. The LPC model parameters for the AR spectrum estimate (14.1.1) are obtained by minimizing the mean-square prediction error:

$$\mathcal{E}=E[e_n^*e_n]=\mathbf{a}^\dagger R\mathbf{a}=\min\,,\quad e_n=\sum_{m=0}^{M}a_m y_{n-m} \tag{14.2.12}$$

where $\mathbf{a}=[1,a_1,a_2,\dots,a_M]^T$ is the prediction-error filter and R, the autocorrelation matrix (14.2.9). The minimization of $\mathcal{E}$ must be subject to the linear constraint that the first entry of $\mathbf{a}$ be unity. This constraint can be expressed in vector form

$$a_0=\mathbf{u}_0^\dagger\mathbf{a}=1 \tag{14.2.13}$$

where $\mathbf{u}_0 = [1, 0, 0, \dots, 0]^T$ is the unit vector consisting of 1 followed by M zeros. Incorporating this constraint with a Lagrange multiplier, we solve the minimization problem:

$$\mathcal{E} = \mathbf{a}^\dagger R\mathbf{a} + \mu(1 - \mathbf{u}_0^\dagger \mathbf{a}) = \min$$

Differentiating with respect to $\mathbf{a}$ we obtain the normal equations:

$$R\mathbf{a} = \mu\mathbf{u}_0$$

To fix the Lagrange multiplier, multiply from the left by $\mathbf{a}^\dagger$ and use Eq. (14.2.13) to get $\mathbf{a}^\dagger R\mathbf{a} = \mu\mathbf{a}^\dagger\mathbf{u}_0$, or, $\mathcal{E} = \mu$. Thus, μ is the minimized value of $\mathcal{E}$, which we denote by E. In summary, we have

$$R\mathbf{a} = E\mathbf{u}_0 \quad \Rightarrow \quad \mathbf{a} = ER^{-1}\mathbf{u}_0 \tag{14.2.14}$$

Multiplying from the left by $\mathbf{u}_0^\dagger$, we also find $1 = E(\mathbf{u}_0^\dagger R^{-1}\mathbf{u}_0)$, or

$$E^{-1} = \mathbf{u}_0^\dagger R^{-1}\mathbf{u}_0 = (R^{-1})_{00} \tag{14.2.15}$$

which is, of course, the same as Eq. (12.9.18). The special structure of R allows the computation of $\mathbf{a}$ and the AR spectrum in closed form. Applying the matrix inversion lemma to Eq. (14.2.9), we find the inverse of R:

$$R^{-1} = \frac{1}{\sigma_v^2}(I + SDS^\dagger) \tag{14.2.16}$$

where D is an $L{\times}L$ matrix given by

$$D = -\big[\sigma_v^2 P^{-1} + S^\dagger S\big]^{-1} \tag{14.2.17}$$

Equation (14.2.16) can also be derived directly by assuming such an expression for R^{-1} and then fixing D. The quantity $\sigma_v^2 P^{-1}$ in D is a matrix of noise to signal ratios. Inserting Eq. (14.2.16) into (14.2.14), we find for $\mathbf{a}$:

$$\mathbf{a} = ER^{-1}\mathbf{u}_0 = \frac{E}{\sigma_v^2}[\mathbf{u}_0 + SDS^\dagger\mathbf{u}_0] = \frac{E}{\sigma_v^2}[\mathbf{u}_0 + S\mathbf{d}]$$

where we used the fact that $\mathbf{s}_{\omega_i}^\dagger \mathbf{u}_0 = 1$, which implies that

$$S^\dagger\mathbf{u}_0 = \begin{bmatrix} \mathbf{s}_{\omega_1}^\dagger \\ \mathbf{s}_{\omega_2}^\dagger \\ \vdots \\ \mathbf{s}_{\omega_L}^\dagger \end{bmatrix} \mathbf{u}_0 = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \equiv \mathbf{v} \qquad \text{(i.e., a column of } L \text{ ones)}$$

and defined

$$\mathbf{d} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_L \end{bmatrix} = D\mathbf{v}, \quad \text{or,} \quad d_i = \sum_{j=1}^{L} D_{ij}$$

Using Eq. (14.2.15), we have also

$$E^{-1} = \mathbf{u}_0^\dagger R^{-1}\mathbf{u}_0 = \frac{1}{\sigma_v^2}\mathbf{u}_0^\dagger[I + SDS^\dagger] = \frac{1}{\sigma_v^2}[1 + \mathbf{v}^T D\mathbf{v}]$$

$$= \frac{1}{\sigma_v^2}[1 + \mathbf{v}^T\mathbf{d}] = \frac{1}{\sigma_v^2}\Big[1 + \sum_{i=1}^{L} d_i\Big]$$

and, therefore,

$$E = \sigma_v^2\Big[1 + \sum_{i=1}^{L} d_i\Big]^{-1} \tag{14.2.18}$$

We finally find for the prediction-error filter

$$\mathbf{a} = \frac{\mathbf{u}_0 + S\mathbf{d}}{1 + \mathbf{v}^T\mathbf{d}} = \frac{u_0 + \sum_{i=1}^{L} d_i\mathbf{s}_{\omega_i}}{1 + \sum_{i=1}^{L} d_i} \tag{14.2.19}$$

The frequency response $A(\omega)$ of the prediction-error filter is obtained by dotting the phasing vector $\mathbf{s}_\omega$ into $\mathbf{a}$:

$$A(\omega) = \sum_{m=0}^{M} a_m e^{-j\omega m} = \mathbf{s}_\omega^\dagger\mathbf{a} = \frac{1 + \sum_{i=1}^{L} d_i\mathbf{s}_\omega^\dagger\mathbf{s}_{\omega_i}}{1 + \sum_{i=1}^{L} d_i}$$

using the result that $\mathbf{s}_\omega^\dagger\mathbf{s}_{\omega_i} = W(\omega - \omega_i)$, we finally find:

$$A(\omega) = \frac{1 + \sum_{i=1}^{L} d_i W(\omega - \omega_i)}{1 + \sum_{i=1}^{L} d_i} \tag{14.2.20}$$

and for the AR, or maximum entropy, spectrum estimate:

$$\hat{S}_{AR}(\omega) = \frac{E}{|A(\omega)|^2} = \sigma_v^2 \frac{\Big|1 + \sum_{i=1}^{L} d_i\Big|}{\Big|1 + \sum_{i=1}^{L} d_i W(\omega - \omega_i)\Big|^2} \tag{14.2.21}$$

The frequency dependence is shown explicitly. Note, that the matrix $S^\dagger S$ appearing in the definition of D, can also be expressed in terms of $W(\omega)$. Indeed, the ijth element of $S^\dagger S$ is, for $0 \le i, j \le L$:

$$(S^\dagger S)_{ij} = \mathbf{s}_{\omega_i}^\dagger\mathbf{s}_{\omega_j} = W(\omega_i - \omega_j)$$

One interesting consequence of Eq. (14.2.21) is that in the limit of very weak noise $\sigma_v^2 \to 0$, it vanishes. In this limit the mean-square prediction error (14.2.18) vanishes. This is to be expected, since in this case the noise term v_n is absent from the sum (14.2.1), rendering y_n a deterministic signal; that is, one that can be predicted from a few past values with zero prediction error. To avoid such behavior when σ_v^2 is small, the factor E is sometimes dropped altogether from the spectral estimate resulting in the "pseudo-spectrum"

$$\hat{S}_{AR}(\omega)=\frac{1}{|A(\omega)|^2} \tag{14.2.22}$$

This expression will exhibit fairly sharp peaks at the sinusoid frequencies, but the magnitude of these peaks will no longer be representative of the power levels P_i. This expression can only be used to extract the frequencies ω_i. Up to a scale factor, Eq. (14.2.22) can also be written in the form

$$\hat{S}_{AR}(\omega)=\frac{1}{\big|\mathbf{s}_\omega^\dagger R^{-1}\mathbf{u}_0\big|^2}$$

Example 14.2.1: To see the effect of the SNR on the sharpness of the peaks in the AR spectrum, consider the case $M = L = 1$. Then,

$$S^\dagger S=\mathbf{s}_{\omega_1}^\dagger \mathbf{s}_{\omega_1}=\big[1,\; e^{-j\omega_1}\big]\begin{bmatrix}1\\ e^{j\omega_1}\end{bmatrix}=M+1=2$$

$$D=-[\sigma_v^2 P_1^{-1}+2]^{-1}$$

$$\mathbf{a}=\frac{\mathbf{u}_0+d_1\mathbf{s}_{\omega_1}}{1+d_1}=\begin{bmatrix}1\\ \dfrac{d_1}{1+d_1}\,e^{j\omega_1}\end{bmatrix}$$

Using $d_1 = D$, we find

$$\mathbf{a}=\begin{bmatrix}1\\ -\dfrac{P_1}{P_1+\sigma_v^2}\,e^{j\omega_1}\end{bmatrix},\qquad A(z)=1+a_1z^{-1}$$

The prediction-error filter has a zero at

$$z_1=-a_1=\frac{P_1}{P_1+\sigma_v^2}\,e^{j\omega_1}$$

The zero z_1 is inside the unit circle, as it should. The lower the $SNR = P_1/\sigma_v^2$, the more inside it lies, resulting in a more smeared peak about ω_1. As the SNR increases, the zero moves closer to the unit circle at the right frequency ω_1, resulting in a very sharp peak in the spectrum (14.2.22). □

Example 14.2.2: For the case of a single sinusoid and arbitrary order M, compute the 3-dB width of the spectral peak of AR spectrum and compare it with the width of the Bartlett spectrum. Using Eq. (14.2.20), we have

$$A(\omega)=\frac{1+d_1W(\omega-\omega_1)}{1+d_1},\qquad d_1=-[SNR^{-1}+M+1]^{-1}$$

where we set $SNR = P_1/\sigma_v^2$. The value of $A(\omega)$ at the sinusoid frequency is

$$A(\omega_1)=\frac{1+d_1W(0)}{1+d_1}=\frac{1}{1+SNR\cdot M}$$

It is small in the limit of high SNR resulting in a high peak in the spectrum. The half-width at half-maximum of the AR spectrum is defined by the condition

$$\frac{S(\omega_1+\Delta\omega)}{S(\omega_1)}=\frac{1}{2},\quad\text{or, equivalently,}\quad \frac{|A(\omega_1+\Delta\omega)|^2}{|A(\omega_1)|^2}=2$$

To first order in $\Delta\omega$, we have

$$W(\Delta\omega)=\sum_{m=0}^{M}e^{-jm\Delta\omega}=\sum_{m=0}^{M}(1-jm\Delta\omega)=(M+1)-\frac{1}{2}jM(M+1)\Delta\omega$$

where we used $\sum_{m=0}^{M} m = M(M+1)/2$. Then, we find

$$\frac{A(\omega_1+\Delta\omega)}{A(\omega_1)}=\frac{1+d_1W(\Delta\omega)}{1+d_1W(0)}=1-\frac{1}{2}\,SNR\cdot jM(M+1)\Delta\omega$$

The condition for half-maximum requires that the above imaginary part be unity, which gives for the 3-dB width [1083]

$$(\Delta\omega)_{3\text{dB}}=2\Delta\omega=\frac{4}{SNR\cdot M(M+1)}$$

Thus, the peak becomes narrower both with increasing SNR and with order M. Note that it depends on M like $O(1/M^2)$, which is a factor of M smaller than the Bartlett width that behaves like $O(1/M)$. □

More generally, in the case of multiple sinusoids, if the SNRs are high the spectrum (14.2.22) will exhibit sharp peaks at the desired sinusoid frequencies. The mechanism by which this happens can be seen qualitatively from Eq. (14.2.20) as follows: The matrix $S^\dagger S$ in D introduces cross-coupling among the various frequencies ω_i. However, if these frequencies are well separated from each other (by more than $2\pi/(M+1)$,) then the off-diagonal elements of $S^\dagger S$, namely $W(\omega_i-\omega_j)$ will be small, and for the purpose of this argument may be taken to be zero. This makes the matrix $S^\dagger S$ approximately diagonal. Since $W(0)=M+1$ it follows that $S^\dagger S=(M+1)I$, and D will become diagonal with diagonal elements

$$d_i=D_{ii}=-[\sigma_v^2P_i^{-1}+M+1]^{-1}=-\frac{P_i}{\sigma_v^2+(M+1)P_i}$$

Evaluating $A(\omega)$ at ω_i and keeping only the ith contribution in the sum we find, approximately,

$$A(\omega_i)\simeq\frac{1+d_iW(0)}{1+\sum_{j=0}^{L}d_j}=\frac{1}{1+\sum_{j=0}^{L}d_j}\;\frac{1}{1+(M+1)\left(\dfrac{P_i}{\sigma_v^2}\right)}$$

which shows that if the SNRs P_i/σ_v^2 are high, $A(\omega_i)$ will be very small, resulting in large spectral peaks in Eq. (14.2.22). The resolvability properties of the AR estimate improve both when the SNRs increase and when the order M increases. The mutual interaction of the various sinusoid components cannot be ignored altogether. One effect of this interaction is *biasing* in the estimates of the frequencies; that is, even if two nearby peaks are clearly separated, the peaks may not occur exactly at the desired sinusoid frequencies, but may be slightly shifted. The degree of bias depends on the relative separation of the peaks and on the SNRs. With the above qualifications in mind, we can state that the LPC approach to this problem is one of the most successful ones.

Capon's maximum likelihood (ML), or minimum energy, spectral estimator is given by the expression [1081]

$$\hat{S}_{ML}(\omega)=\frac{1}{\mathbf{s}_\omega^\dagger R^{-1}\mathbf{s}_\omega} \tag{14.2.23}$$

It can be justified by envisioning a bank of narrowband filters, each designed to allow a sinewave through at the filter's center frequency and to attenuate all other frequency components. Thus, the narrowband filter with center frequency ω is required to let this frequency go through unchanged, that is,

$$A(\omega)=\mathbf{s}_\omega^\dagger\mathbf{a}=1$$

while at the same time it is required to minimize the output power

$$\mathbf{a}^\dagger R\mathbf{a}=\min$$

The solution of this minimization problem subject to the above constraint is readily found to be

$$\mathbf{a}=\frac{R^{-1}\mathbf{s}_\omega}{\mathbf{s}_\omega^\dagger R^{-1}\mathbf{s}_\omega}$$

which gives for the minimized output power at this frequency

$$\mathbf{a}^\dagger R\mathbf{a}=\frac{1}{\mathbf{s}_\omega^\dagger R^{-1}\mathbf{s}_\omega}$$

Using Eq. (14.2.16), we find

$$\begin{aligned}\mathbf{s}_\omega^\dagger R^{-1}\mathbf{s}_\omega &= \frac{1}{\sigma_v^2}\Big[\mathbf{s}_\omega^\dagger\mathbf{s}_\omega+\sum_{i,j=1}^{L}D_{ij}\mathbf{s}_\omega^\dagger\mathbf{s}_{\omega_i}\mathbf{s}_{\omega_j}^\dagger\mathbf{s}_\omega\Big]\\ &=\frac{1}{\sigma_v^2}\Big[(M+1)+\sum_{i,j=1}^{L}D_{ij}W(\omega-\omega_i)W^*(\omega-\omega_j)\Big]\end{aligned}$$

and the theoretical ML spectrum becomes in this case:

$$\hat{S}_{ML}(\omega)=\frac{\sigma_v^2}{\Big[(M+1)+\sum_{i,j=1}^{L}D_{ij}W(\omega-\omega_i)W^*(\omega-\omega_j)\Big]} \tag{14.2.24}$$

Example 14.2.3: Determine the matrix D and vector $\mathbf{d}$ for the case of $L=2$ and arbitrary M. The matrix $S^\dagger S$ is in this case

$$S^\dagger S=\begin{bmatrix} W(0) & W(\omega_1-\omega_2)\\ W(\omega_2-\omega_1) & W(0)\end{bmatrix}=\begin{bmatrix} M+1 & W_{12}\\ W_{12}^* & M+1\end{bmatrix}$$

where $W_{12}=W(\omega_1-\omega_2)$. Then, D becomes

$$D=-\begin{bmatrix}\sigma_v^2P_1^{-1}+M+1 & W_{12}\\ W_{12}^* & \sigma_v^2P_2^{-1}+M+1\end{bmatrix}^{-1},\quad\text{or,}$$

$$D=\frac{1}{|W_{12}|^2-(\sigma_v^2P_1^{-1}+M+1)(\sigma_v^2P_2^{-1}+M+1)}\begin{bmatrix}\sigma_v^2P_2^{-1}+M+1 & -W_{12}\\ -W_{12}^* & \sigma_v^2P_1^{-1}+M+1\end{bmatrix}$$

and, hence

$$\mathbf{d}=D\begin{bmatrix}1\\1\end{bmatrix}=\frac{1}{|W_{12}|^2-(\sigma_v^2P_1^{-1}+M+1)(\sigma_v^2P_2^{-1}+M+1)}\begin{bmatrix}\sigma_v^2P_2^{-1}+M+1-W_{12}\\ \sigma_v^2P_1^{-1}+M+1-W_{12}^*\end{bmatrix}$$

Using the results of Example 14.2.3, we have carried out a computation illustrating the three spectral estimates. Fig. 14.2.1 shows the theoretical autoregressive, Bartlett, and maximum likelihood spectral estimates given by Eqs. (14.2.11), (14.2.22), and (14.2.24), respectively, for two sinusoids of frequencies

$$\omega_1=0.4\pi\,,\quad \omega_2=0.6\pi$$

and equal powers $SNR=10\log_{10}(P_1/\sigma_v^2)=6$ dB, and $M=6$. To facilitate the comparison, all three spectra have been normalized to 0 dB at the frequency ω_1 of the first sinusoid. It is seen that the length $M=6$ is too short for the Bartlett spectrum to resolve the two peaks. The AR spectrum is the best (however, close inspection of the graph will reveal a small bias in the frequency of the peaks, arising from the mutual interaction of the two sinewaves). The effect of increasing the SNR is shown on the right in Fig. 14.2.1, where the SNR has been changed to 12 dB. It is seen that the AR spectral peaks become narrower, thus increasing their resolvability.

To show the effect of increasing M, we kept $SNR=6$ dB, and increased the order to $M=12$ and $M=18$. The resulting spectra are shown in Fig. 14.2.2. It is seen that all spectra tend to become better. The interplay between resolution, order, SNR, and bias has been studied in [1083,1085,1088].

The main motivation behind the definition (14.2.22) for the pseudospectrum was to obtain an expression that exhibits very sharp spectral peaks at the sinusoid frequencies ω_i. Infinite resolution can, in principle, be achieved if we can find a polynomial $A(z)$ that has zeros *on the unit circle* at the desired frequency angles; namely, at

$$z_i=e^{j\omega_i}\,,\quad i=1,2,\dots,L \tag{14.2.25}$$

Pisarenko's method determines such a polynomial on the basis of the autocorrelation matrix R. The desired conditions on the polynomial are

$$A(z_i)=A(\omega_i)=0\,,\quad i=1,2,\dots,L \tag{14.2.26}$$

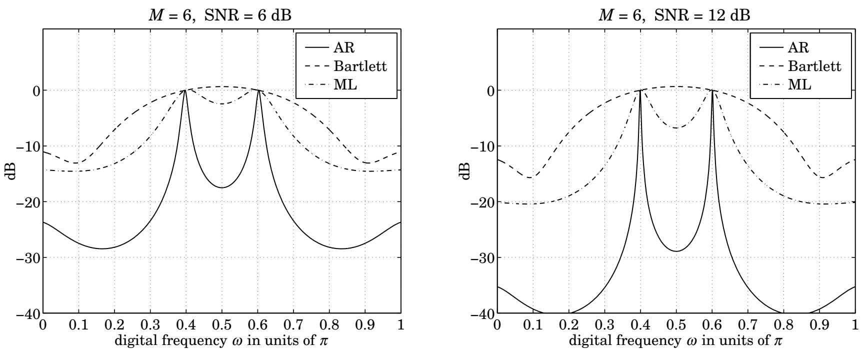


Fig. 14.2.1 AR, Bartlett, and ML spectrum estimates.

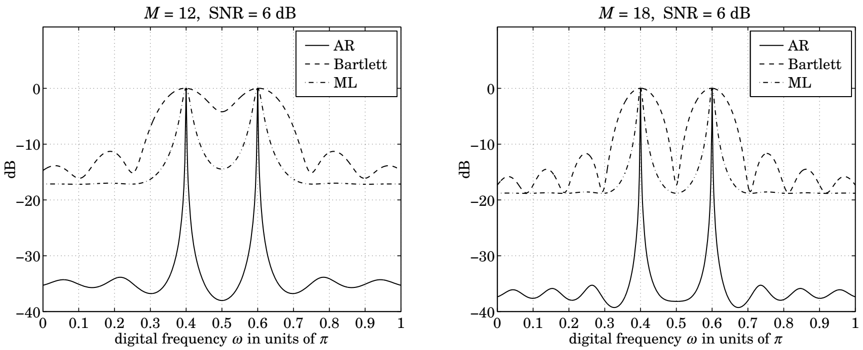


Fig. 14.2.2 AR, Bartlett, and ML spectrum estimates.

where we slightly abuse the notation and write $A(e^{j\omega})= A(\omega)$. To satisfy these conditions, the degree M of the polynomial $A(z)$ must necessarily be $M \geq L$; then, the remaining $M-L$ zeros of $A(z)$ could be arbitrary. Let $\mathbf{a}$ be the vector of coefficients of $A(z)$, so that

$$\mathbf{a} = \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_M \end{bmatrix}, \quad A(z)= a_0 + a_1 z^{-1} + \cdots + a_M z^{-M}$$

Noting that $A(\omega)= \mathbf{s}_\omega^\dagger \mathbf{a}$, Eqs. (14.2.26) may be combined into one vectorial equation

$$S^\dagger \mathbf{a} = \begin{bmatrix} \mathbf{s}_{\omega_1}^\dagger \\ \mathbf{s}_{\omega_2}^\dagger \\ \vdots \\ \mathbf{s}_{\omega_L}^\dagger \end{bmatrix} \mathbf{a} = \begin{bmatrix} A(\omega_1) \\ A(\omega_2) \\ \vdots \\ A(\omega_L) \end{bmatrix} = 0 \tag{14.2.27}$$

But then, Eq. (14.2.9) implies that

$$R\mathbf{a} = \sigma_v^2 \mathbf{a} + SPS^\dagger \mathbf{a} = \sigma_v^2 \mathbf{a}$$

or, that σ_v^2 must be an *eigenvalue* of R with $\mathbf{a}$ the corresponding eigenvector:

$$R\mathbf{a} = \sigma_v^2 \mathbf{a} \tag{14.2.28}$$

The quantity σ_v^2 is actually the *smallest* eigenvalue of R. To see this, consider any other eigenvector $\mathbf{a}$ of R, and normalize it to unit norm

$$R\mathbf{a} = \lambda \mathbf{a}, \quad \text{with} \quad \mathbf{a}^\dagger \mathbf{a} = 1 \tag{14.2.29}$$

Then, (14.2.9) implies that

$$\begin{aligned} \lambda &= \lambda \mathbf{a}^\dagger \mathbf{a} = \mathbf{a}^\dagger R \mathbf{a} = \sigma_v^2 \mathbf{a}^\dagger \mathbf{a} + \mathbf{a} SPS^\dagger \mathbf{a} \\ &= \sigma_v^2 + \big[A(\omega_1)^*,\, A(\omega_2)^*,\, \dots,\, A(\omega_L)^*\big] \begin{bmatrix} P_1 & & & \\ & P_2 & & \\ & & \ddots & \\ & & & P_L \end{bmatrix} \begin{bmatrix} A(\omega_1) \\ A(\omega_2) \\ \vdots \\ A(\omega_L) \end{bmatrix} \\ &= \sigma_v^2 + \sum_{i=1}^{L} P_i |A(\omega_i)|^2 \end{aligned}$$

which shows that λ is equal to σ_v^2 shifted by a non-negative amount. If the eigenvector satisfies the conditions (14.2.26), then the shift in λ vanishes. Thus, the desired polynomial $A(z)$ can be found by solving the eigenvalue problem (14.2.29) and selecting the eigenvector belonging to the minimum eigenvalue. This is Pisarenko's method [1084]. As a byproduct of the procedure, the noise power level σ_v^2 is also determined, which in turn allows the determination of the power matrix P, as follows. Writing Eq. (14.2.9) as

$$R - \sigma_v^2 I = SPS^\dagger$$

and acting by $S^\dagger$ and S from the left and right, we obtain

$$P = U^\dagger (R - \sigma_v^2 I) U, \quad \text{where} \quad U = S(S^\dagger S)^{-1} \tag{14.2.30}$$

Since there is freedom in selecting the *remaining* $M-L$ zeros of the polynomial $A(z)$, it follows that there are $(M-L)+1$ eigenvectors all belonging to the minimum eigenvalue σ_v^2. Thus, the $(M+1)$-dimensional eigenvalue problem (14.2.29) has two sets of eigenvalues: (a) $M+1-L$ *degenerate* eigenvalues equal to σ_v^2, and (b) L additional eigenvalues which are strictly greater than σ_v^2.

The $(M+1-L)$-dimensional subspace spanned by the degenerate eigenvectors belonging to σ_v^2 is called the *noise subspace*. The L-dimensional subspace spanned by the eigenvectors belonging to the remaining L eigenvalues is called the *signal subspace*. Since the signal subspace is orthogonal to the noise subspace. and the L linearly independent signal vectors $\mathbf{s}_{\omega_i}$, $i = 1, 2, \dots, L$ are also orthogonal to the noise subspace, it follows that the signal subspace is *spanned* by the $\mathbf{s}_{\omega_i}$s.

In the special case when $L = M$ (corresponding to the Pisarenko's method), there is no degeneracy in the minimum eigenvalue, and there is a *unique* minimum eigenvector. In this case, all $M = L$ zeros of $A(z)$ lie on the unit circle at the desired angles ω_i.

Example 14.2.4: Consider the case $L = M = 2$. The matrix R is written explicitly as:

$$R = \sigma_v^2 I + P_1 \mathbf{s}_{\omega_1}\mathbf{s}_{\omega_1}^\dagger + P_2 \mathbf{s}_{\omega_2}\mathbf{s}_{\omega_2}^\dagger\,, \quad \text{or,}$$

$$R = \begin{bmatrix} \sigma_v^2 + P_1 + P_2 & P_1 e^{-j\omega_1} + P_2 e^{-j\omega_2} & P_1 e^{-2j\omega_1} + P_2 e^{-2j\omega_2} \\ P_1 e^{j\omega_1} + P_2 e^{j\omega_2} & \sigma_v^2 + P_1 + P_2 & P_1 e^{-j\omega_1} + P_2 e^{-j\omega_2} \\ P_1 e^{2j\omega_1} + P_2 e^{2j\omega_2} & P_1 e^{j\omega_1} + P_2 e^{j\omega_2} & \sigma_v^2 + P_1 + P_2 \end{bmatrix}$$

It is easily verified that the (unnormalized) vector

$$\mathbf{a} = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 1 \\ -(e^{j\omega_1} + e^{j\omega_2}) \\ e^{j\omega_1} e^{j\omega_2} \end{bmatrix}$$

is an eigenvector of R belonging to $\lambda = \sigma_v^2$. In this case, the polynomial $A(z)$ is

$$\begin{aligned} A(z) &= a_0 + a_1 z^{-1} + a_2 z^{-2} = 1 - (e^{j\omega_1} + e^{j\omega_2})z^{-1} + e^{j\omega_1} e^{j\omega_2} z^{-2} \\ &= (1 - e^{j\omega_1} z^{-1})(1 - e^{j\omega_2} z^{-1}) \end{aligned}$$

exhibiting the two desired zeros at the sinusoid frequencies. □

Example 14.2.5: Consider the case $M = 2$, $L = 1$. The matrix R is

$$R = \sigma_v^2 I + P_1 \mathbf{s}_{\omega_1}\mathbf{s}_{\omega_1}^\dagger = \begin{bmatrix} \sigma_v^2 + P_1 & P_1 e^{-j\omega_1} & P_1 e^{-2j\omega_1} \\ P_1 e^{j\omega_1} & \sigma_v^2 + P_1 & P_1 e^{-j\omega_1} \\ P_1 e^{2j\omega_1} & P_1 e^{j\omega_1} & \sigma_v^2 + P_1 \end{bmatrix}$$

It is easily verified that the three eigenvectors of R are

$$\mathbf{e}_0 = \begin{bmatrix} 1 \\ -e^{j\omega_1} \\ 0 \end{bmatrix}, \quad \mathbf{e}_1 = \begin{bmatrix} 0 \\ 1 \\ -e^{j\omega_1} \end{bmatrix}, \quad \mathbf{e}_2 = \begin{bmatrix} 1 \\ e^{j\omega_1} \\ e^{2j\omega_1} \end{bmatrix}$$

belonging to the eigenvalues

$$\lambda = \sigma_v^2\,, \quad \lambda = \sigma_v^2\,, \quad \lambda = \sigma_v^2 + 3P_1$$

The first two eigenvectors span the noise subspace and the third, the signal subspace. Any linear combination of the noise eigenvectors also belongs to $\lambda = \sigma_v^2$. For example, if we take

$$\mathbf{a} = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 1 \\ -e^{j\omega_1} \\ 0 \end{bmatrix} - \rho \begin{bmatrix} 0 \\ 1 \\ -e^{j\omega_1} \end{bmatrix} = \begin{bmatrix} 1 \\ -(\rho + e^{j\omega_1}) \\ \rho e^{j\omega_1} \end{bmatrix}$$

the corresponding polynomial is

$$A(z) = 1 - (\rho + e^{j\omega_1})z^{-1} + \rho e^{j\omega_1} z^{-2} = (1 - e^{j\omega_1} z^{-1})(1 - \rho z^{-1})$$

showing one desired zero at $z_1 = e^{j\omega_1}$ and a spurious zero. □

The Pisarenko method can also be understood in terms of a *minimization* criterion of the type (14.2.12), as follows. For any set of coefficients $\mathbf{a}$, define the output signal

$$e_n = \sum_{m=0}^{M} a_m y_{n-m} = a_0 y_n + a_1 y_{n-1} + \cdots + a_M y_{n-M}$$

Then, the mean output power is expressed as

$$\mathcal{E} = E[e_n^* e_n] = \mathbf{a}^\dagger R \mathbf{a} = \sigma_v^2 \mathbf{a}^\dagger \mathbf{a} + \sum_{i=1}^{L} P_i |A(\omega_i)|^2$$

Imposing the quadratic constraint

$$\mathbf{a}^\dagger \mathbf{a} = 1 \tag{14.2.31}$$

we obtain

$$\mathcal{E} = E[e_n^* e_n] = \mathbf{a}^\dagger R \mathbf{a} = \sigma_v^2 + \sum_{i=1}^{L} P_i |A(\omega_i)|^2 \tag{14.2.32}$$

It is evident that the minimum of this expression is obtained when conditions (14.2.26) are satisfied. Thus, an equivalent formulation of the Pisarenko method is to minimize the performance index (14.2.32) subject to the quadratic constraint (14.2.31). The AR and the Pisarenko spectrum estimation techniques differ only in the type of constraint imposed on the filter weights $\mathbf{a}$.

We observed earlier that the AR spectral peaks become sharper as the SNR increases. One way to explain this is to note that in the high-SNR limit or, equivalently, in the noiseless limit $\sigma_v^2 \to 0$, the linear prediction filter tends to the Pisarenko filter, which has infinite resolution. This can be seen as follows. In the limit $\sigma_v^2 \to 0$, the matrix D defined in Eq. (14.2.17) tends to

$$D \to -(S^\dagger S)^{-1}$$

and therefore, R^{-1} given by Eq. (14.2.16) becomes singular, converging to

$$R^{-1} \to \frac{1}{\sigma_v^2}\big[I - S(S^\dagger S)^{-1} S^\dagger\big]$$

Thus, up to a scale factor the linear prediction solution, $R^{-1}\mathbf{u}_0$ will converge to

$$\mathbf{a} = \big[I - S(S^\dagger S)^{-1} S^\dagger\big]\mathbf{u}_0 \tag{14.2.33}$$

The matrix $\big[I - S(S^\dagger S)^{-1} S^\dagger\big]$ is the *projection* matrix onto the noise subspace, and therefore, $\mathbf{a}$ will lie in that subspace, that is, $S^\dagger \mathbf{a} = 0$. In the limit $\sigma_v^2 \to 0$, the noise subspace of R consists of all the eigenvectors with zero eigenvalue, $R\mathbf{a} = 0$. We note that the particular noise subspace eigenvector given in Eq. (14.2.33) corresponds to the so-called *minimum-norm* eigenvector, discussed in Sec. 14.6.

In his original method, Pisarenko considered the special case when the number of sinusoids was equal to the filter order, $L = M$. This implies that the noise subspace is one-dimensional, $M + 1 - L = 1$, consisting of a single eigenvector with zero eigenvalue, such that $R\mathbf{a} = 0$. In this case, the $(M + 1)\times(M + 1)$ singular matrix R has rank M and all its principal submatrices are nonsingular. As we mentioned in Sec. 12.5, such singular Toeplitz matrices admit a general sinusoidal representation. It is obtained by setting $\sigma_v^2 = 0$ and $L = M$ in Eq. (14.2.8):

$$R = \sum_{i=1}^{L} P_i \mathbf{s}_{\omega_i} \mathbf{s}_{\omega_i}^\dagger\,, \quad \text{or,} \quad R(k) = \sum_{i=1}^{L} P_i e^{j\omega_i k}$$

In summary, we have discussed the theoretical aspects of four methods of estimating the frequencies of sinusoids in noise. In practice, an estimate of the correlation matrix R can be obtained in terms of the sample autocorrelations from a block of data values:

$$\hat{R}(k) = \frac{1}{N} \sum_{n=0}^{N-1-k} y_{n+k} y_n^*\,, \quad k = 0, 1, \dots, M$$

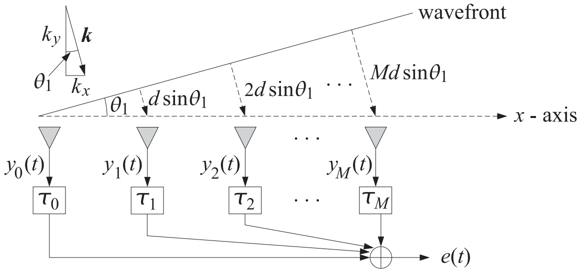
The quality of the resulting estimates of the eigenvectors will be discussed in Section 14.11. The AR and Pisarenko methods can also be implemented *adaptively*. The adaptive approach is based on the minimization criteria (14.2.12) and (14.2.32) and will be discussed in Chap. 16, where also some simulations will be presented.

14.3 Superresolution Array Processing

One of the main signal processing functions of sonar, radar, or seismic arrays of sensors is to detect the presence of one or more radiating point-sources. This is a problem of spectral analysis, and it is the *spatial frequency* analog of the problem of extracting sinusoids in noise discussed in the previous section. The same spectral analysis techniques can be applied to this problem. All methods aim at producing a high-resolution estimate of the spatial frequency power spectrum of the signal field incident on the array of sensors. The *directions* of point-source emitters can be extracted by identifying the sharpest peaks in this spectrum.

In this section, we discuss conventional (Bartlett) *beamforming*, as well as the *maximum likelihood*, *linear prediction*, and *eigenvector* based methods. We also discuss some aspects of *optimum beamforming* for interference nulling [1093–1095,1352,1167–1170].

Consider a linear array of $M + 1$ sensors equally spaced at distances d, and a plane wave incident on the array at an angle θ_1 with respect to the array normal, as shown below.



The conventional beamformer introduces appropriate delays at the outputs of each sensor to compensate for the propagation delays of the wavefront reaching the array. The output of the beamformer (the "beam") is the sum

$$e(t) = \sum_{m=0}^{M} y_m(t - \tau_m) \tag{14.3.1}$$

where $y_m(t)$, $m = 0, 1, \dots, M$ is the signal at the mth sensor. To reach sensor 1, the wavefront must travel an extra distance $d\sin\theta_1$, to reach sensor 2 it must travel distance $2d\sin\theta_1$, and so on. Thus, it reaches these sensors with a propagation delay of $d\sin\theta_1/c$, $2d\sin\theta_1/c$, and so on. The last sensor is reached with a delay of $Md\sin\theta_1/c$ seconds. Thus, to time-align the first and the last sensor, the output of the first sensor must be delayed by $\tau_0 = Md\sin\theta_1/c$, and similarly, the mth sensor is time-aligned with the last one, with a delay of

$$\tau_m = \frac{1}{c}(M - m)d\sin\theta_1 \tag{14.3.2}$$

In this case, all terms in the sum (14.3.1) are equal to the value measured at the last sensor, that is, $y_m(t - \tau_m) = y_M(t)$, and the output of the beamformer is $e(t) = (M + 1)y_M(t)$, thus enhancing the received signal by a factor of $M + 1$ and hence its power by a factor $(M + 1)^2$. The concept of beamforming is the same as that of signal averaging. If there is additive noise present, it will contribute incoherently to the output power, that is, by a factor of $(M+1)$, whereas the signal power is enhanced by $(M+1)^2$. Thus, the gain in the signal to noise ratio at the output of the array (the array gain) is a factor of $M + 1$.

In the frequency domain, the above delay-and-sum operation becomes equivalent to linear weighting. Fourier transforming Eq. (14.3.1) we have

$$e(\omega) = \sum_{m=0}^{M} y_m(\omega) e^{-j\omega\tau_m}$$

which can be written compactly as:

$$e = \mathbf{a}^T \mathbf{y} \tag{14.3.3}$$

where $\mathbf{a}$ and $\mathbf{y}$ are the $(M + 1)$-vectors of weights and sensor outputs:

$$\mathbf{a} = \begin{bmatrix} e^{-j\omega\tau_0} \\ e^{-j\omega\tau_1} \\ \vdots \\ e^{-j\omega\tau_M} \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_0(\omega) \\ y_1(\omega) \\ \vdots \\ y_M(\omega) \end{bmatrix}$$

From now on, we will concentrate on narrow-band arrays operating at a given frequency ω and the dependence on ω will not be shown explicitly. This assumes that the signals from all the sensors have been subjected to narrow-band prefiltering that leaves only the narrow operating frequency band. The beamformer now acts as a linear combiner, as shown in Fig. 14.3.1. A plane wave at the operating frequency ω, of amplitude

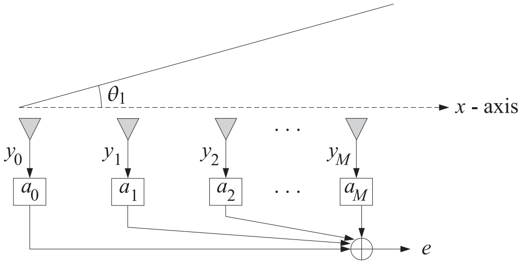


Fig. 14.3.1 Beamforming

A_1, and incident at the above angle θ_1, will have a value at the space-time point $(t, \mathbf{r})$ given by

$$A_1 e^{j\omega t - j\mathbf{k}\cdot\mathbf{r}}$$

Dropping the sinusoidal t-dependence and evaluating this expression on the x-axis, we have

$$A_1 e^{-jk_x x}$$

where k_x is the x-components of the wave vector $\mathbf{k}$

$$k_x = \frac{\omega}{c}\sin\theta_1$$

The value of this field at the mth sensor, $x_m = md$, is then

$$A_1 e^{-jmk_1}$$

where k_1 denotes the *normalized* wavenumber

$$k_1 = k_x d = \frac{\omega d}{c}\sin\theta_1 = \frac{2\pi d}{\lambda}\sin\theta_1\,, \quad \lambda = \text{wavelength} \tag{14.3.4}$$

This is the spatial analog of the digital frequency. To avoid aliasing effects arising from the spatial sampling process, the spatial sampling frequency $1/d$ must be greater than or equal to twice the spatial frequency $1/\lambda$ of the wave. Thus, we must have $d^{-1} \geq 2\lambda^{-1}$, or $d \leq \lambda/2$. Since $\sin\theta_1$ has magnitude less than one, the sampling condition forces k_1 to lie within the Nyquist interval

$$-\pi \leq k_1 \leq \pi$$

In this case the correspondence between k_1 and θ_1, is unique. For any angle θ and corresponding normalized wavenumber k, we introduce the *phasing*, or *steering vector*

$$\mathbf{s}_k = \begin{bmatrix} 1 \\ e^{jk} \\ e^{2jk} \\ \vdots \\ e^{Mjk} \end{bmatrix}, \quad k = \frac{2\pi d}{\lambda}\sin\theta \tag{14.3.5}$$

In this notation, the plane wave measured at the sensors is represented by the vector

$$\mathbf{y} = A_1 \mathbf{s}_{k_1}^* = A_1 \begin{bmatrix} 1 \\ e^{-jk_1} \\ e^{-2jk_1} \\ \vdots \\ e^{-Mjk_1} \end{bmatrix}$$

The steering vector of array weights $\mathbf{a}$, steered towards an arbitrary direction θ, is also expressed in terms of the phasing vector $\mathbf{s}_k$; we have

$$a_m = e^{-j\omega\tau_m} = e^{-j\omega(M-m)(d\sin\theta/c)} = e^{-jMk}e^{jmk}$$

or, ignoring the overall common phase e^{-jMk}, we have

$$\mathbf{a} = \mathbf{s}_k \qquad (\text{steering vector towards } k = \frac{2\pi d}{\lambda}\sin\theta) \tag{14.3.6}$$

The output of the beamformer, steered towards θ, is

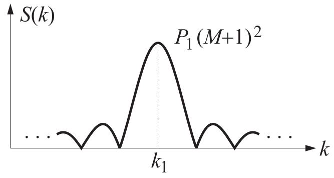
$$e = \mathbf{a}^T\mathbf{y} = \mathbf{s}_k^T\mathbf{y} = A_1\,\mathbf{s}_k^T\mathbf{s}_{k_1}^* = A_1\,\mathbf{s}_{k_1}^\dagger\mathbf{s}_k = A_1 W(k-k_1)^*$$

where $W(\cdot)$ was defined in Sec. 14.2. The mean output power of the beamformer steered towards k is

$$S(k) = E[e^*e] = \mathbf{a}^\dagger E[\mathbf{y}^*\mathbf{y}^T]\mathbf{a} = \mathbf{a}^\dagger R\mathbf{a} = \mathbf{s}_k^\dagger R\mathbf{s}_k$$

Using $\mathbf{y} = A_1\mathbf{s}_{k_1}^*$, we find $R = E[\mathbf{y}^*\mathbf{y}^T] = P_1\mathbf{s}_{k_1}\mathbf{s}_{k_1}^\dagger$, where $P_1 = E[|A_1|^2]$, and

$$\begin{aligned} S(k) &= \mathbf{s}_k^\dagger R\mathbf{s}_k = P_1\mathbf{s}_k^\dagger\mathbf{s}_{k_1}\mathbf{s}_{k_1}^\dagger\mathbf{s}_k \\ &= P_1|W(k-k_1)|^2 \end{aligned}$$



If the beam is steered on target, that is, if $\theta = \theta_1$, or, $k = k_1$, then $S(k_1) = P_1(M+1)^2$ and the output power is enhanced. The response pattern of the array has the same shape as the function $W(k)$, and therefore its resolution capability is limited to the width $\Delta k = 2\pi/(M+1)$ of the main lobe of $W(k)$. Setting $\Delta k = (2\pi d/\lambda)\Delta\theta$, we find the basic angular resolution to be $\Delta\theta = \lambda/\big((M+1)d\big)$, or, $\Delta\theta = \lambda/D$, where $D = (M+1)d$ is the effective aperture of the array. This is the classical *Rayleigh limit* on the resolving power of an optical system with aperture D [1092].

Next, we consider the problem of resolving the directions of arrival of multiple plane waves incident on an array in the presence of background noise. We assume L planes waves incident on an array of $M+1$ sensors from angles θ_i, $i = 1, 2, \dots, L$. The incident field is sampled at the sensors giving rise to a series of "snapshots." At the nth snapshot time instant, the field received at the mth sensor has the form [1394]

$$y_m(n) = v_m(n) + \sum_{i=1}^{L} A_i(n)e^{-jmk_i}\,, \quad m = 0, 1, \dots, M \tag{14.3.7}$$

where $A_i(n)$ is the amplitude of the ith wave (it would be a constant independent of time if we had exact sinusoidal dependence at the operating frequency), and k_i are the normalized wavenumbers related to the angles of arrival by

$$k_i = \frac{2\pi d}{\lambda}\sin\theta_i\,,\quad i = 1,2,\dots,L \tag{14.3.8}$$

and $v_m(n)$ is the background noise, which is assumed to be *spatially incoherent*, and also uncorrelated with the signal amplitudes $A_i(n)$; that is,

$$E[v_m(n)^* v_k(n)] = \sigma_v^2\delta_{mk}\,,\quad E[v_m(n)^* A_i(n)] = 0 \tag{14.3.9}$$

Eq. (14.3.7) can be written in vector form as follows

$$\mathbf{y}(n) = \mathbf{v}(n) + \sum_{i=1}^{L} A_i(n)\mathbf{s}_{k_i}^* \tag{14.3.10}$$

The autocorrelation matrix of the signal field sensed by the array is

$$R = E[\mathbf{y}(n)^*\mathbf{y}(n)^T] = \sigma_v^2 I + \sum_{i,j=1}^{L}\mathbf{s}_{k_i}P_{ij}\mathbf{s}_{k_j}^\dagger \tag{14.3.11}$$

where I is the $(M+1)\times(M+1)$ unit matrix, and P_{ij} is the amplitude correlation matrix

$$P_{ij} = E[A_i(n)^* A_j(n)]\,,\quad 1\le i,j\le L \tag{14.3.12}$$

If the sources are uncorrelated with respect to each other, the power matrix P_{ij} is diagonal. Introducing the $(M+1)\times L$ signal matrix

$$S = [\mathbf{s}_{k_1},\ \mathbf{s}_{k_2},\ \dots,\ \mathbf{s}_{k_L}]$$

we may write Eq. (14.3.11) as

$$R = \sigma_v^2 I + SPS^\dagger \tag{14.3.13}$$

which is the same as Eq. (14.2.9) of the previous section. Therefore, the analytical expressions of the various spectral estimators can be transferred to this problem as well. We summarize the various spectrum estimators below:

$$\hat{S}_B(k) = \mathbf{s}_k^\dagger R\mathbf{s}_k \qquad \text{(conventional Bartlett beamformer)}$$

$$\hat{S}_{LP}(k) = \frac{1}{|\mathbf{s}_k^\dagger R^{-1}\mathbf{u}_0|^2} \qquad \text{(LP spectrum estimate)}$$

$$\hat{S}_{ML}(k) = \frac{1}{\mathbf{s}_k^\dagger R^{-1}\mathbf{s}_k} \qquad \text{(ML spectrum estimate)}$$

For example, for uncorrelated sources $P_{ij} = P_i\delta_{ij}$, the Bartlett spatial spectrum will be

$$\hat{S}_B(k) = \mathbf{s}_k^\dagger R\mathbf{s}_k = \sigma_v^2(M+1) + \sum_{i=1}^{L}P_i|W(k-k_i)|^2$$

which gives rise to peaks at the desired wavenumbers k_i from which the angles θ_i can be extracted. When the beam is steered towards the ith plane wave, the measured power at the output of the beamformer will be

$$\hat{S}_B(k_i) = \sigma_v^2(M+1) + P_i(M+1)^2 + \sum_{j\neq i}P_j|W(k_i-k_j)|^2$$

Ignoring the third term for the moment, we observe the basic improvement in the SNR offered by beamforming:

$$\frac{P_i(M+1)^2}{\sigma_v^2(M+1)} = \frac{P_i}{\sigma_v^2}(M+1)$$

If the sources are too close to each other [closer than the beamwidth of $W(k)$], the resolution ability of the beamformer worsens. In such cases, the alternative spectral estimates offer better resolution, with the LP estimate typically having a better performance. The resolution capability of both the ML and the LP estimates improves with higher SNR, whereas that of the conventional beamformer does not.

The Pisarenko method can also be applied here. As discussed in the previous section, the $(M+1)$-dimensional eigenvalue problem $R\mathbf{a} = \lambda\mathbf{a}$ has an L-dimensional *signal subspace* with eigenvalues greater than σ_v^2, and an $(M+1-L)$-dimensional *noise subspace* spanned by the degenerate eigenvectors belonging to the minimum eigenvalue of σ_v^2. Any vector $\mathbf{a}$ in the noise subspace will have at least L zeros at the desired wavenumber frequencies k_i, that is, the polynomial

$$A(z) = a_0 + a_1z^{-1} + a_2z^{-2} + \cdots + a_Mz^{-M}$$

will have L zeros at

$$z_i = e^{jk_i}\,,\quad i = 1,2,\dots,L$$

and $(M-L)$ other spurious zeros. This can be seen as follows: If $R\mathbf{a} = \sigma_v^2\mathbf{a}$, then Eq. (14.3.13) implies that

$$(\sigma_v^2 I + SPS^\dagger)\mathbf{a} = \sigma_v^2\mathbf{a} \quad\Rightarrow\quad SPS^\dagger\mathbf{a} = 0$$

Dotting with $\mathbf{a}^\dagger$, we find that $\mathbf{a}^\dagger SPS^\dagger\mathbf{a} = 0$, and since P is assumed to be strictly positive definite, it follows that $S^\dagger\mathbf{a} = 0$, or

$$S^\dagger\mathbf{a} = \begin{bmatrix} A(k_1)\\ A(k_2)\\ \vdots\\ A(k_L)\end{bmatrix} = 0$$

The L largest eigenvalues of R correspond to the signal subspace eigenvectors and can be determined by reducing the original $(M+1)\times(M+1)$ eigenvalue problem for R into a smaller $L\times L$ eigenvalue problem.

Let $\mathbf{e}$ be any eigenvector in the signal subspace, that is, $R\mathbf{e} = \lambda\mathbf{e}$, with $\lambda > \sigma_v^2$. It follows that $SPS^\dagger\mathbf{e} = (\lambda-\sigma_v^2)\mathbf{e}$. Multiplying both sides by $S^\dagger$ we obtain $(S^\dagger SP)(S^\dagger\mathbf{e}) = (\lambda-\sigma_v^2)(S^\dagger\mathbf{e})$, which states that the L-dimensional vector $S^\dagger\mathbf{e}$ is an eigenvector of the

$L\times L$ matrix $S^\dagger SP$. We can turn this into a hermitian eigenvalue problem by factoring the power matrix P into its square root factors, $P = GG^\dagger$, and multiplying both sides of the reduced eigenvalue problem by $G^\dagger$. This gives $(G^\dagger S^\dagger SG)(G^\dagger S^\dagger \mathbf{e}) = (\lambda - \sigma_v^2)(G^\dagger S^\dagger \mathbf{e})$. Thus, we obtain the $L\times L$ hermitian eigenvalue problem

$$F\mathbf{f} = (\lambda - \sigma_v^2)\mathbf{f}\,, \quad \text{where} \quad F = G^\dagger S^\dagger SG\,, \quad \mathbf{f} = G^\dagger S^\dagger \mathbf{e} \tag{14.3.14}$$

The L signal subspace eigenvalues are obtained from the solution of this reduced eigenproblem. From each L-dimensional eigenvector $\mathbf{f}$, one can also construct the corresponding $(M+1)$-dimensional eigenvector $\mathbf{e}$. Because $\mathbf{e}$ lies in the signal subspace, it can be expressed as a linear combination of the plane waves

$$\mathbf{e} = \sum_{i=1}^{L} c_i \mathbf{s}_{k_i} = [\mathbf{s}_{k_1},\, \mathbf{s}_{k_2},\, \dots,\, \mathbf{s}_{k_L}] \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_L \end{bmatrix} = S\mathbf{c}$$

It, then, follows from Eq. (14.3.14) that

$$\mathbf{f} = G^\dagger S^\dagger \mathbf{e} = G^\dagger S^\dagger S\mathbf{c} \quad \Rightarrow \quad \mathbf{c} = (S^\dagger S)^{-1} G^{-\dagger}\mathbf{f}$$

and therefore,

$$\mathbf{e} = S\mathbf{c} = S(S^\dagger S)^{-1} G^{-\dagger}\mathbf{f} \tag{14.3.15}$$

Example 14.3.1: Using the above reduction method, determine the signal subspace eigenvectors and eigenvalues for the case of two equal-power uncorrelated plane waves and arbitrary M. The 2×2 matrix P becomes proportional to the identity matrix $P = P_1 I$. The reduced matrix F is then

$$F = P_1 S^\dagger S = P_1 \begin{bmatrix} \mathbf{s}_1^\dagger \mathbf{s}_1 & \mathbf{s}_1^\dagger \mathbf{s}_2 \\ \mathbf{s}_2^\dagger \mathbf{s}_1 & \mathbf{s}_2^\dagger \mathbf{s}_2 \end{bmatrix} = P_1 \begin{bmatrix} M+1 & W_{12} \\ W_{12}^* & M+1 \end{bmatrix}$$

where $\mathbf{s}_1 = \mathbf{s}_{k_1}$, $\mathbf{s}_2 = \mathbf{s}_{k_2}$, and $W_{12} = W(k_1 - k_2)$. In the equal-power case, F is always proportional to $S^\dagger S$, and therefore, $\mathbf{f}$ is an eigenvector of that. It follows that $(S^\dagger S)^{-1}\mathbf{f}$ will be a scalar multiple of $\mathbf{f}$ and that Eq. (14.3.15) can be simplified (up to a scalar factor) to $\mathbf{e} = S\mathbf{f}$. The two eigenvalues and eigenvectors of F are easily found to be

$$\lambda - \sigma_v^2 = P_1(M + 1 \pm |W_{12}|)\,, \quad \mathbf{f} = \begin{bmatrix} 1 \\ \pm e^{-j\theta_{12}} \end{bmatrix}$$

where θ_{12} is the phase of W_{12}. Using $\mathbf{e} = S\mathbf{f}$, it follows that the two signal subspace eigenvectors will be

$$\mathbf{e} = \mathbf{s}_1 \pm e^{-j\theta_{12}}\mathbf{s}_2$$

The eigenvalue spread of R is in this case

$$\frac{\lambda_{\max}}{\lambda_{\min}} = \frac{\sigma_v^2 + (M + 1 + |W_{12}|)P_1}{\sigma_v^2} = 1 + SNR\cdot(M + 1 + |W_{12}|)$$

where $SNR = P_1/\sigma_v^2$. It can be written in the form

$$\frac{\lambda_{\max}}{\lambda_{\min}} = 1 + SNR_{\text{eff}}\cdot(1 + |\cos\phi_{12}|)$$

where $SNR_{\text{eff}} = SNR\cdot(M+1)$ is the effective SNR of the array, or the array gain, and ϕ_{12} is the angle between the two signal vectors, that is, $\cos\phi_{12} = \mathbf{s}_1^\dagger\mathbf{s}_2/(\|\mathbf{s}_1\|\cdot\|\mathbf{s}_2\|)$. □

In practice, estimates of the covariance matrix R are used. For example, if the sensor outputs are recorded over N snapshots, that is, $\mathbf{y}(n)$, $n = 0, 1, \dots, N-1$, then, the covariance matrix R may be estimated by replacing the ensemble average of Eq. (14.3.11) with the time-average:

$$\hat{R} = \frac{1}{N}\sum_{n=0}^{N-1}\mathbf{y}(n)^*\mathbf{y}(n)^T \qquad \text{(empirical } R\text{)}$$

Since the empirical R will not be of the exact theoretical form of Eq. (14.3.11) the degeneracy of the noise subspace will be lifted somewhat. The degree to which this happens depends on how much the empirical R differs from the theoretical R. One can still use the minimum eigenvector $\mathbf{a}$ to define the polynomial $A(z)$ and from it an approximate Pisarenko spectral estimator

$$\hat{S}_P(k) = \frac{1}{|A(z)|^2}\,, \quad \text{where } z = e^{jk}$$

which will have sharp and possibly biased peaks at the desired wavenumber frequencies.

Example 14.3.2: Consider the case $L = M = 1$, defined by the theoretical autocorrelation matrix

$$R = \sigma_v^2 I + P_1\mathbf{s}_{k_1}\mathbf{s}_{k_1}^\dagger = \begin{bmatrix} \sigma_v^2 + P_1 & P_1 e^{-jk_1} \\ P_1 e^{jk_1} & \sigma_v^2 + P_1 \end{bmatrix}$$

Its eigenvectors are:

$$\mathbf{e}_0 = \begin{bmatrix} 1 \\ -e^{jk_1} \end{bmatrix}, \quad \mathbf{e}_1 = \mathbf{s}_{k_1} = \begin{bmatrix} 1 \\ e^{jk_1} \end{bmatrix}$$

belonging to the eigenvalues $\lambda_0 = \sigma_v^2$ and $\lambda_1 = \sigma_v^2 + 2P_1$, respectively. Selecting as the array vector

$$\mathbf{a} = \mathbf{e}_0 = \begin{bmatrix} 1 \\ -e^{jk_1} \end{bmatrix}$$

we obtain a polynomial with a zero at the desired location:

$$A(z) = a_0 + a_1 z^{-1} = 1 - e^{jk_1}z^{-1}$$

Now, suppose that the analysis is based on an empirical autocorrelation matrix R which differs from the theoretical one by a small amount:

$$\hat{R} = R + \Delta R$$

Using standard first-order perturbation theory, we find the correction to the minimum eigenvalue λ_0 and eigenvector $\mathbf{e}_0$

$$\hat{\lambda}_0 = \lambda_0 + \Delta\lambda_0\,, \quad \hat{\mathbf{e}}_0 = \mathbf{e}_0 + \Delta c\,\mathbf{e}_1$$

where the first-order correction terms are

$$\Delta\lambda_0 = \frac{\mathbf{e}_0^\dagger(\Delta R)\mathbf{e}_0}{\mathbf{e}_0^\dagger\mathbf{e}_0}\,, \quad \Delta c = \frac{\mathbf{e}_1^\dagger(\Delta R)\mathbf{e}_0}{(\lambda_0 - \lambda_1)\mathbf{e}_1^\dagger\mathbf{e}_1}$$

The change induced in the zero of the eigenpolynomial is found as follows

$$\hat{\mathbf{a}} = \hat{\mathbf{e}}_0 = \begin{bmatrix} 1 \\ -e^{jk_1} \end{bmatrix} + \Delta c \begin{bmatrix} 1 \\ e^{jk_1} \end{bmatrix} = \begin{bmatrix} 1 + \Delta c \\ -(1 - \Delta c)\, e^{jk_1} \end{bmatrix}$$

so that

$$\hat{A}(z) = (1 + \Delta c) - (1 - \Delta c)\, e^{jk_1} z^{-1}$$

and the zero is now at

$$z_1 = \frac{1 - \Delta c}{1 + \Delta c}\, e^{jk_1} \simeq (1 - 2\Delta c)\, e^{jk_1}$$

to first-order in Δc. Since Δc is generally complex, the factor $(1 - 2\Delta c)$ will cause both a change (bias) in the phase of the zero e^{jk_1}, and will move it off the unit circle reducing the resolution. Another way to see this is to compute the value of the polynomial steered on target; that is,

$$\hat{A}(k_1) = \mathbf{s}_{k_1}^{\dagger} \mathbf{a} = \mathbf{s}_{k_1}^{\dagger} (\mathbf{e}_0 + \Delta c\, \mathbf{e}_1) = \Delta c\, \mathbf{s}_{k_1}^{\dagger} \mathbf{e}_1 = 2\Delta c$$

which is small but not zero. □

The high resolution properties of the Pisarenko and other eigenvector methods depend directly on the assumption that the background noise field is spatially incoherent, resulting in the special structure of the autocorrelation matrix R. When the noise is spatially coherent, a different eigenanalysis must be carried out. Suppose that the covariance matrix of the noise field $\mathbf{v}$ is

$$E[\mathbf{v}^* \mathbf{v}^T] = \sigma_v^2 Q$$

where Q reflects the spatial coherence of $\mathbf{v}$. Then the covariance matrix of Eq. (14.3.13) is replaced by

$$R = \sigma_v^2 Q + SPS^{\dagger} \tag{14.3.16}$$

The relevant eigenvalue problem is now the *generalized* eigenvalue problem

$$R\mathbf{a} = \lambda Q \mathbf{a} \tag{14.3.17}$$

Consider any such generalized eigenvector $\mathbf{a}$, and assume it is normalized such that

$$\mathbf{a}^{\dagger} Q \mathbf{a} = 1 \tag{14.3.18}$$

Then, the corresponding eigenvalue is expressed as

$$\lambda = \lambda \mathbf{a}^{\dagger} Q \mathbf{a} = \mathbf{a}^{\dagger} R \mathbf{a} = \sigma_v^2 \mathbf{a}^{\dagger} Q \mathbf{a} + \mathbf{a}^{\dagger} SPS^{\dagger} \mathbf{a} = \sigma_v^2 + \mathbf{a}^{\dagger} SPS^{\dagger} \mathbf{a}$$

which shows that the minimum eigenvalue is σ_v^2 and is attained whenever $\mathbf{a}^{\dagger} SPS^{\dagger} \mathbf{a} = 0$, or equivalently (assuming that P has full rank), $S^{\dagger}\mathbf{a} = 0$, or, $A(k_i) = 0$, $i = 1, 2, \dots, L$. Therefore, the eigenpolynomial $A(z)$ can be used to determine the wavenumbers k_i.

Thus, the procedure is to solve the generalized eigenvalue problem and select the minimum eigenvector. This eigenvalue problem is also equivalent to the minimization problem

$$\mathcal{E} = \mathbf{a}^{\dagger} R \mathbf{a} = \min\,, \quad \text{subject to } \ \mathbf{a}^{\dagger} Q \mathbf{a} = 1 \tag{14.3.19}$$

This criterion, and its solution as the minimum eigenvector, is equivalent to the unconstrained minimization of the *Rayleigh quotient*, that is,

$$\frac{\mathbf{a}^{\dagger} R \mathbf{a}}{\mathbf{a}^{\dagger} Q \mathbf{a}} = \min \quad \Leftrightarrow \quad R\mathbf{a} = \lambda_{\min} Q \mathbf{a} \tag{14.3.20}$$

The practical implementation of the method requires knowledge of the noise covariance matrix Q, which is not always possible to obtain. Covariance difference methods [1135–1138] can be used in the case of unknown Q. Such methods work with measurements from two different arrays, translated or rotated with respect to each other. Assuming that the background noise is invariant under translation or rotation, the covariance matrices of the two arrays will be $R_1 = S_1 P_1 S_1^{\dagger} + \sigma_v^2 Q$ and $R_2 = S_2 P_2 S_2^{\dagger} + \sigma_v^2 Q$. The eigenstructure of the covariance difference $R_1 - R_2 = S_1 P_1 S_1^{\dagger} - S_2 P_2 S_2^{\dagger}$ can be used to extract the signal information.

The two spectral analysis problems discussed in this and the previous section—direction finding and harmonic retrieval—are *dual* to each other; one dealing with spatial frequencies and the other with time frequencies. The optimum processing part is the same in both cases. The optimum processor does not care how its inputs are supplied, it only “sees” the *correlations* among the inputs and its function is to “break down” these correlations thereby extracting the sinusoidal components. The two cases differ only in the way the inputs to the optimum processor are supplied. This conceptual separation between the input part and the optimum processing part is shown in Fig. 14.3.2. In the time series case, the correlations among the inputs are *sequential correlations* in time, whereas in the array case they are *spatial correlations*, such as those that exist along a coherent wavefront.

A problem related, but not identical, to direction finding is that of *optimum beamforming* for interference nulling [1093–1095,1352,1167–1170]. In this case, one of the plane waves, say, $\mathbf{s}_{k_1}$, is assumed to be a desired plane wave with *known* direction of arrival θ_1, or wavenumber k_1. The other plane waves are considered as interferers or jammers to be nulled. Assuming for simplicity uncorrelated sources, the covariance matrix (14.3.11) may be decomposed into a part due to the desired signal and a part due to the noise plus interference:

$$R = \sigma_v^2 I + \sum_{i=1}^{L} P_i\, \mathbf{s}_i \mathbf{s}_i^{\dagger} = P_1 \mathbf{s}_1 \mathbf{s}_1^{\dagger} + \Big[\sigma_v^2 I + \sum_{i=2}^{L} P_i\, \mathbf{s}_i \mathbf{s}_i^{\dagger}\Big] = P_1 \mathbf{s}_1 \mathbf{s}_1^{\dagger} + R_n$$

where we denoted $\mathbf{s}_i = \mathbf{s}_{k_i}$. The output power of the array with weights $\mathbf{a}$ will be

$$\mathcal{E} = \mathbf{a}^{\dagger} R \mathbf{a} = P_1 |\mathbf{s}_1^{\dagger} \mathbf{a}|^2 + \mathbf{a}^{\dagger} R_n \mathbf{a} \tag{14.3.21}$$

The first term is the output power due to the desired signal; the second term is due to the presence of noise plus interference. This expression suggests two possible optimization criteria for $\mathbf{a}$. First, choose $\mathbf{a}$ to maximize the relative *signal to noise plus interference ratio* (SNIR):

$$\mathit{SNIR} = \frac{P_1 |\mathbf{s}_1^{\dagger} \mathbf{a}|^2}{\mathbf{a}^{\dagger} R_n \mathbf{a}} = \max \tag{14.3.22}$$

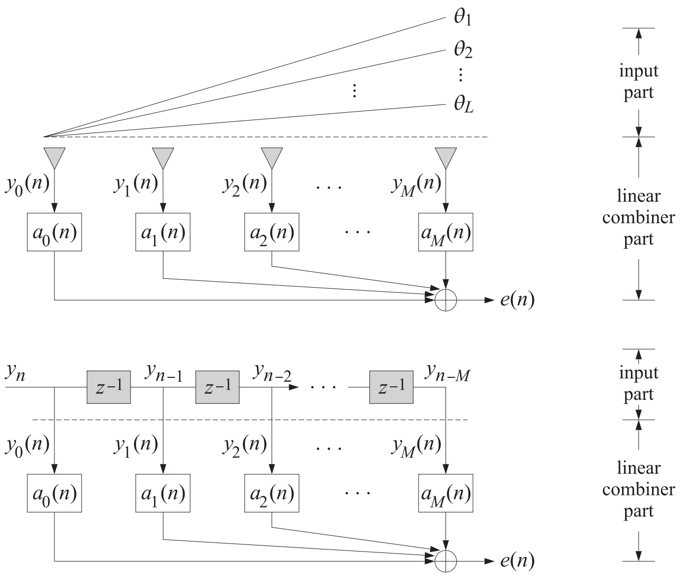


Fig. 14.3.2 Duality between time series and array problems.

The second criterion is to keep the output of the array toward the look direction $\mathbf{s}_1$ fixed, while minimizing the output power:

$$\mathbf{s}_1^\dagger\mathbf{a} = 1 \quad\text{and}\quad \mathcal{E} = \mathbf{a}^\dagger R\mathbf{a} = P_1 + \mathbf{a}^\dagger R_n\mathbf{a} = \min \tag{14.3.23}$$

This is equivalent to minimizing the noise plus interference term $\mathbf{a}^\dagger R_n\mathbf{a}$. These two criteria are essentially equivalent. This is seen as follows. Equation (14.3.22) is equivalent to *minimizing* the inverse function $SNIR^{-1}$. Adding one to it, we obtain the equivalent criterion

$$1 + SNIR^{-1} = 1 + \frac{\mathbf{a}^\dagger R_n\mathbf{a}}{P_1|\mathbf{s}_1^\dagger\mathbf{a}|^2} = \frac{\mathbf{a}^\dagger R\mathbf{a}}{P_1|\mathbf{s}_1^\dagger\mathbf{a}|^2} = \min$$

This is identical to the Rayleigh quotient (14.3.20) with the choice $Q = P_1\mathbf{s}_1\mathbf{s}_1^\dagger$. It is equivalent to the minimum eigenvector solution of

$$R\mathbf{a} = \lambda Q\mathbf{a} = \lambda P_1\mathbf{s}_1\mathbf{s}_1^\dagger\mathbf{a} = \mu\mathbf{s}_1 \quad\Rightarrow\quad \mathbf{a} = \mu R^{-1}\mathbf{s}_1$$

where we put all the scalar factors into μ. Similarly, the constraint $\mathbf{s}_1^\dagger\mathbf{a} = 1$ implies that $\mathbf{a}^\dagger Q_1\mathbf{a} = 1$ with $Q_1 = \mathbf{s}_1\mathbf{s}_1^\dagger$. It follows from Eq. (14.3.19), applied with Q_1, that the solution of Eq. (14.3.23) is again the generalized eigenvector

$$R\mathbf{a} = \lambda_1 Q_1\mathbf{a} = \lambda_1\mathbf{s}_1\mathbf{s}_1^\dagger\mathbf{a} = \mu_1\mathbf{s}_1 \quad\Rightarrow\quad \mathbf{a} = \mu_1 R^{-1}\mathbf{s}_1$$

Thus, up to a scale factor, the optimum solution for both criteria is

$$\mathbf{a} = R^{-1}\mathbf{s}_1 \tag{14.3.24}$$

This solution admits, yet, a third interpretation as the *Wiener solution* of an ordinary mean-square estimation problem. The term $\mathbf{y}_1(n) = A_1(n)\mathbf{s}_1^*$ of Eq. (14.3.10) is the desired signal. A reference signal $x(n)$ could be chosen to correlate highly with this term and not at all with the other terms in Eq. (14.3.10). For example, $x(n) = f(n)A_1(n)$. The array weights can be designed by demanding that the scalar output of the array, $\mathbf{a}^T\mathbf{y}(n)$, be the best mean-square estimate of $x(n)$. This gives the criterion

$$E\big[|x(n) - \mathbf{a}^T\mathbf{y}(n)|^2\big] = E\big[|x(n)|^2\big] - \mathbf{a}^\dagger\mathbf{r} - \mathbf{r}^\dagger\mathbf{a} + \mathbf{a}^\dagger R\mathbf{a}$$

where we set $\mathbf{r} = E[x(n)\mathbf{y}(n)^*]$. Minimizing with respect to $\mathbf{a}$ (and $\mathbf{a}^*$) gives the Wiener solution $\mathbf{a} = R^{-1}\mathbf{r}$. Now, because $x(n)$ is correlated only with $\mathbf{y}_1(n)$, it follows that $\mathbf{r}$ will be proportional to $\mathbf{s}_1$:

$$\mathbf{r} = E[x(n)\mathbf{y}(n)^*] = E[x(n)\mathbf{y}_1(n)^*] = E[x(n)A_1(n)^*]\,\mathbf{s}_1$$

Thus, again up to a scale, we obtain the solution (14.3.24). Using the matrix inversion lemma (see Problem 14.6), we can write the inverse of $R = P_1\mathbf{s}_1\mathbf{s}_1^\dagger + R_n$, in the form

$$R^{-1} = R_n^{-1} - cR_n^{-1}\mathbf{s}_1\mathbf{s}_1^\dagger R_n^{-1}\,, \quad c = (P_1^{-1} + \mathbf{s}_1^\dagger R_n^{-1}\mathbf{s}_1)^{-1}$$

Acting by both sides on $\mathbf{s}_1$, we find

$$R^{-1}\mathbf{s}_1 = c_1 R_n^{-1}\mathbf{s}_1\,, \quad c_1 = cP_1^{-1}$$

Therefore, the optimal solution can also be written (up to another scale factor) in terms of the noise plus interference covariance matrix R_n:

$$\mathbf{a} = R_n^{-1}\mathbf{s}_1 \tag{14.3.25}$$

These solutions, known as *steered solutions*, are sometimes modified to include arbitrary tapering weights for the array—replacing the steering vector $\mathbf{s}_1$ with a generalized steering vector

$$\mathbf{s} = \begin{bmatrix} b_0 \\ b_1 e^{jk_1} \\ \vdots \\ b_M e^{jk_1 M} \end{bmatrix} = B\,\mathbf{s}_1\,, \quad B = \text{diag}\{b_0, b_1, \dots, b_M\} \tag{14.3.26}$$

The weights b_m can be chosen to attain a prescribed shape for the quiescent response of the array in absence of interference. Typical choices are (with $k_1 = 0$)

$$\mathbf{s} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad \mathbf{s} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$$

To appreciate the properties of the optimum solution, we consider the case of one jammer, so that

$$R = P_1\mathbf{s}_1\mathbf{s}_1^\dagger + R_n\,, \quad R_n = \sigma_v^2 I + P_2\mathbf{s}_2\mathbf{s}_2^\dagger$$

Using the matrix inversion lemma on R_n^{-1}, we obtain

$$R_n^{-1} = \frac{1}{\sigma_v^2}\left[I - \frac{1}{\sigma_v^2 P_2^{-1} + \mathbf{s}_2^\dagger \mathbf{s}_2}\mathbf{s}_2\mathbf{s}_2^\dagger\right]$$

Therefore, the optimum solution given by Eq. (14.3.25) becomes

$$\mathbf{a} = R_n^{-1}\mathbf{s}_1 = \frac{1}{\sigma_v^2}\left[\mathbf{s}_1 - \frac{P_2 W(k_2-k_1)}{\sigma_v^2 + P_2(M+1)}\mathbf{s}_2\right]$$

where we used $\mathbf{s}_2^\dagger\mathbf{s}_2 = M+1$ and $\mathbf{s}_2^\dagger\mathbf{s}_1 = W(k_2-k_1)$. Dropping the overall factor of $1/\sigma_v^2$, we find for the array pattern as a function of wavenumber k or angle θ

$$A(k) = \mathbf{s}_k^\dagger\mathbf{a} = W(k-k_1) - \frac{P_2 W(k_2-k_1)}{\sigma_v^2 + P_2(M+1)}W(k-k_2) \tag{14.3.27}$$

In the absence of the jammer, $P_2 = 0$, we obtain the usual quiescent Bartlett response, $W(k-k_1)$. The presence of the second term, called a *retrodirective beam*, will partially distort the quiescent pattern but it will suppress the jammer. Indeed, the array response steered toward the jammer at $k = k_2$, becomes

$$A(k_2) = W(k_2-k_1) - \frac{P_2 W(k_2-k_1)}{\sigma_v^2 + P_2(M+1)}W(0) = \frac{W(k_2-k_1)}{\sigma_v^2 + P_2(M+1)}$$

The ratio $A(k_2)/W(k_2-k_1)$ is the array response, in the direction of the jammer, relative to the quiescent response. Thus, if the signal to noise ratio $SNR_2 = P_2/\sigma_v^2$ is large, the jammer will be suppressed. Only in the limit of infinite SNR is the jammer completely nulled.

The reason for the incomplete nulling can be traced, as in the case of linear prediction, to the linear constraint on the weights (14.3.23). To get exact nulling of the jammers, we must force the zeros of the polynomial $\mathbf{a}$ to lie on the unit circle at the jammer positions. As suggested in Problem 14.13, this can be accomplished by imposing a quadratic constraint $\mathbf{a}^\dagger Q\mathbf{a} = \text{const.}$, where Q must be chosen as $Q = \sigma_v^2 I + P_1\mathbf{s}_1\mathbf{s}_1^\dagger$ instead of $Q = P_1\mathbf{s}_1\mathbf{s}_1^\dagger$. The optimum weight is the minimum eigenvector solution of the generalized eigenproblem $R\mathbf{a} = \lambda Q\mathbf{a}$ and will have exact zeros at the jammer positions. As in the linear prediction case, the linearly constrained optimum beamformer solution tends to this eigenvector solution in the limit $\sigma_v^2 \to 0$.

14.4 Eigenvector Methods

The single most important property of eigenvector methods is that, at least in principle, they produce unbiased frequency estimates with infinite resolution, regardless of the signal to noise ratios. This property is not shared by the older methods. For example, the resolution of the Bartlett method is limited by the array aperture, and the resolution of the linear prediction and maximum likelihood methods degenerates with decreasing SNRs. Because of this property, eigenvector methods have received considerable attention in signal processing and have been applied to several problems, such as harmonic retrieval, direction finding, echo resolution, and pole identification [1084,1109–1163].

In the remainder of this chapter, we discuss the theoretical aspects of eigenvector methods in further detail, and present several versions of such methods, such as MUSIC, Minimum-Norm, and ESPRIT.

We have seen that the eigenspace of the covariance matrix R consists of two mutually orthogonal parts: the $(M+1-L)$-dimensional noise subspace spanned by the eigenvectors belonging to the minimum eigenvalue σ_v^2, and the L-dimensional signal subspace spanned by the remaining L eigenvectors having eigenvalues strictly greater than σ_v^2. Let $\mathbf{e}_i$, $i = 0,1,\dots,M$, denote the orthonormal eigenvectors of R in order of increasing eigenvalue, and let $K = M+1-L$ denote the dimension of the noise subspace. Then, the first K eigenvectors, $\mathbf{e}_i$, $i = 0,1,\dots,K-1$, form an orthonormal basis for the noise subspace, and the last L eigenvectors, $\mathbf{e}_i$, $i = K, K+1,\dots,M$, form a basis for the signal subspace. We arrange these basis vectors into the eigenvector matrices:

$$E_N = [\mathbf{e}_0,\,\mathbf{e}_1,\,\dots,\,\mathbf{e}_{K-1}],\quad E_S = [\mathbf{e}_K,\,\mathbf{e}_{K+1},\,\dots,\,\mathbf{e}_M] \tag{14.4.1}$$

Their dimensions are $(M+1)\times K$ and $(M+1)\times L$. The full eigenvector matrix of R is:

$$E = [E_N, E_S] = [\mathbf{e}_0,\,\mathbf{e}_1,\,\dots,\,\mathbf{e}_{K-1},\,\mathbf{e}_K,\,\mathbf{e}_{K+1},\,\dots,\,\mathbf{e}_M] \tag{14.4.2}$$

The *orthonormality* of the eigenvectors is expressed by the unitarity property $E^\dagger E = I$, where I is the $(M+1)$-dimensional unit matrix. The unitarity can be written in terms of the submatrices (14.4.1):

$$E_N^\dagger E_N = I_K\,,\quad E_N^\dagger E_S = 0\,,\quad E_S^\dagger E_S = I_L \tag{14.4.3}$$

where I_K and I_L are the $K\times K$ and $L\times L$ unit matrices. The *completeness* of the eigenvectors is expressed also by the unitarity of E, i.e., $EE^\dagger = I$. In terms of the submatrices, it reads:

$$E_N E_N^\dagger + E_S E_S^\dagger = I \tag{14.4.4}$$

These two terms are the *projection* matrices onto the noise and signal subspaces. We have seen that the L signal direction vectors $\mathbf{s}_{k_i}$ belong to the signal subspace, and therefore, are expressible as linear combinations of E_S. It follows that the signal matrix $S = [s_{k_1},\,\dots,\,s_{k_L}]$ is a non-orthogonal basis of the signal subspace and must be related to E_S by $S = E_S C$, where C is an $L\times L$ *invertible* matrix. Using the orthonormality of E_S, it follows that $S^\dagger S = C^\dagger E_S^\dagger E_S C = C^\dagger C$. Thus, the projector onto the signal subspace may be written as

$$P_S = E_S E_S^\dagger = (SC^{-1})(C^{-\dagger}S^\dagger) = S(C^\dagger C)^{-1}S^\dagger = S(S^\dagger S)^{-1}S^\dagger \tag{14.4.5}$$

We may also obtain a non-orthogonal, but useful, basis for the noise subspace. We have seen that an $(M+1)$-dimensional vector $\mathbf{e}$ lies in the noise subspace—equivalently, it is an eigenvector belonging to the minimum eigenvalue σ_v^2—if and only if the corresponding order-M eigenfilter $E(z)$ has L zeros on the unit circle at the desired signal zeros, $z_i = e^{jk_i}$, $i = 1,2,\dots,L$, and has $M-L = K-1$ other spurious zeros. Such a polynomial will factor into the product:

$$E(z) = A(z)F(z) = A(z)\left[f_0 + f_1 z^{-1} + \cdots + f_{K-1}z^{-(K-1)}\right] \tag{14.4.6}$$

where the zeros of $F(z)$ are the spurious zeros, and $A(z)$ is the *reduced-order* polynomial of order L whose zeros are the desired zeros; that is,

$$A(z)=\prod_{i=1}^{L}(1-e^{jk_i}z^{-1})=1+a_1z^{-1}+\cdots+a_Lz^{-L} \tag{14.4.7}$$

Introducing the K delayed polynomials:

$$B_i(z)=z^{-i}A(z)\,,\quad i=0,1,\dots,K-1 \tag{14.4.8}$$

we may write Eq. (14.4.6) in the form

$$E(z)=f_0B_0(z)+f_1B_1(z)+\cdots+f_{K-1}B_{K-1}(z)=\sum_{i=0}^{K-1}f_iB_i(z) \tag{14.4.9}$$

and in coefficient form

$$\mathbf{e}=\sum_{i=0}^{K-1}f_i\,\mathbf{b}_i=[\mathbf{b}_0,\,\mathbf{b}_1,\,\dots,\,\mathbf{b}_{K-1}]\begin{bmatrix}f_0\\ f_1\\ \vdots\\ f_{K-1}\end{bmatrix}\equiv B\,\mathbf{f} \tag{14.4.10}$$

Because each of the polynomials $B_i(z)$ has L desired zeros, it follows that the corresponding vectors $\mathbf{b}_i$ will lie in the noise subspace. Thus, the matrix B defined in Eq. (14.4.10) will be a non-orthogonal basis of the noise subspace. It is a useful basis because the expansion coefficients $\mathbf{f}$ of any noise subspace vector $\mathbf{e}$ are the coefficients of the spurious polynomial $F(z)$ in the factorization (14.4.6). Put differently, Eq. (14.4.10) parametrizes explicitly the spurious degrees of freedom arising from the K-fold degeneracy of the minimum eigenvalue. The basis vectors $\mathbf{b}_i$, considered as $(M+1)$-dimensional vectors, are simply the delayed versions of the vector of coefficients, $\mathbf{a}=[1,a_1,\dots,a_L]^T$, of the polynomial $A(z)$, that is,

$$\mathbf{b}_i=[\underbrace{0,\,\dots,\,0}_{i\text{ zeros}},\,1,\,a_1,\,\dots,\,a_L,\,\underbrace{0,\,\dots,\,0}_{K-1-i\text{ zeros}}]^T \tag{14.4.11}$$

For example, in the case $L=2$ and $M=5$, we have $K=M+1-L=4$ and B is:

$$B=[\mathbf{b}_0,\,\mathbf{b}_1,\,\mathbf{b}_2,\,\mathbf{b}_3]=\begin{bmatrix}1&0&0&0\\ a_1&1&0&0\\ a_2&a_1&1&0\\ 0&a_2&a_1&1\\ 0&0&a_2&a_1\\ 0&0&0&a_2\end{bmatrix}$$

It follows that the basis B must be linearly related to the orthonormal basis E_N by $B=E_NC$, where C is a $K\times K$ invertible matrix. Then, $B^\dagger B=C^\dagger C$ and the projector onto the noise subspace becomes:

$$P_N=E_NE_N^\dagger=(BC^{-1})(C^{-\dagger}B^\dagger)=B(C^\dagger C)^{-1}B^\dagger=B(B^\dagger B)^{-1}B^\dagger \tag{14.4.12}$$

Combining Eqs. (14.4.12) and (14.4.5), we may write the completeness relation (14.4.4) in terms of the non-orthogonal bases B and S:

$$B(B^\dagger B)^{-1}B^\dagger+S(S^\dagger S)^{-1}S^\dagger=I \tag{14.4.13}$$

The objective of all eigenvector methods is to estimate the signal zeros $z_i=e^{jk_i}$, $i=1,2,\dots,L$. All methods begin with an eigenanalysis of R, such that E_N and E_S are available. In practice, the eigenanalysis is based on the sample covariance matrix $\hat{R}$ defined on the basis of a finite number of snapshots, say N:

$$\hat{R}=\frac{1}{N}\sum_{n=0}^{N-1}\mathbf{y}(n)^*\mathbf{y}(n)^T \tag{14.4.14}$$

Sometimes, a symmetrized version is preferred, obtained from $\hat{R}$ by

$$\hat{R}_s=\frac{1}{2}(\hat{R}+J\hat{R}^*J) \tag{14.4.15}$$

where J the $(M+1)$-dimensional reversing matrix. The matrix $\hat{R}_s$ is invariant under reversal, that is, $J\hat{R}_sJ=\hat{R}_s^*$. This version is appropriate when the theoretical R is Toeplitz. This case arises if and only if the $L\times L$ power matrix P is diagonal; that is, when the L sources are mutually uncorrelated. As the number of snapshots increases, the eigenstructure of $\hat{R}$ or $\hat{R}_s$ becomes a better and better approximation of the eigenstructure of R. Such asymptotic statistical properties will be discussed in Sec. 14.11. Next, we discuss several practical approaches.

14.5 MUSIC method

Let $E_i(z)$ denote the eigenfilters of the noise subspace eigenvectors $\mathbf{e}_i$, $i=0,1,\dots,K-1$. According to Eq. (14.4.5), we can write $E_i(z)=A(z)F_i(z)$, which shows that $E_i(z)$ have a common set of L zeros at the desired signal locations, but each may have a different set of $K-1$ spurious zeros. It is possible for these spurious zeros to lie very close to or on the unit circle. Therefore, if only one eigenfilter is used, there may be an ambiguity in distinguishing the desired zeros from the spurious ones. The *multiple signal classification* (MUSIC) method [1110] attempts to average out the effect of the spurious zeros by forming the sum of the magnitude responses of the K noise subspace eigenfilters, that is, setting $z=e^{jk}$,

$$\frac{1}{K}\sum_{i=0}^{K-1}|E_i(k)|^2=|A(k)|^2\frac{1}{K}\sum_{i=0}^{K-1}|F_i(k)|^2$$

Because the polynomials $F_i(z)$ are all different, the averaging operation will tend to smear out any spurious zero of any individual term in the sum. Thus, the above expression will effectively vanish only at the L desired zeros of the common factor$|A(k)|^2$. The MUSIC pseudospectrum is defined as the inverse

$$S_{MUS}(k)=\frac{1}{\dfrac{1}{K}\displaystyle\sum_{i=0}^{K-1}|E_i(k)|^2} \tag{14.5.1}$$

It will exhibit peaks at the L desired wavenumbers k_i, $i = 0,1,\dots,L$. The sum may also be replaced by a weighted sum [1118]. The sum may be written compactly in terms of the projection matrices onto the noise or signal subspaces. Noting that $|E_i(k)|^2 = \mathbf{s}_k^\dagger(\mathbf{e}_i\mathbf{e}_i^\dagger)\mathbf{s}_k$, we find

$$\sum_{i=0}^{K-1}|E_i(k)|^2 = \mathbf{s}_k^\dagger\left[\sum_{i=0}^{K-1}\mathbf{e}_i\mathbf{e}_i^\dagger\right]\mathbf{s}_k = \mathbf{s}_k^\dagger E_N E_N^\dagger\mathbf{s}_k = \mathbf{s}_k^\dagger(I - E_S E_S^\dagger)\mathbf{s}_k$$

where we used Eq. (14.4.4). The practical version of the MUSIC method is summarized below:

1. Based on a finite number of snapshots, compute the sample covariance matrix $\hat{R}$, solve its eigenproblem, and obtain the estimated eigenvector matrix E with eigenvalues arranged in increasing order.
2. Estimate the dimension K of the noise subspace as the number of the smallest, approximately equal, eigenvalues. This can be done systematically using the AIC or MDL criteria discussed later. The estimated number of plane waves will be $L = M + 1 - K$. Divide E into its noise and signal subspace parts, E_N and E_S.
3. Compute the spectrum (14.5.1) and extract the desired wavenumbers k_i from the L peaks in this spectrum.

The Akaike (AIC) and minimum description length (MDL) information-theoretic criteria have been suggested to determine the number of plane waves that are present, or equivalently, the dimension of the noise subspace [1125]. They are defined by

$$\begin{aligned} AIC(k) &= -2Nk\mathcal{L}(k) + 2(M+1-k)(M+1+k) \\ MDL(k) &= -Nk\mathcal{L}(k) + \frac{1}{2}(M+1-k)(M+1+k)\log(N) \end{aligned} \tag{14.5.2}$$

for $k = 1,2,\dots,M+1$, where N is the number of snapshots and $\mathcal{L}(k)$ is a likelihood function defined as the log of the ratio of the harmonic and arithmetic means of the first k estimated eigenvalues $\{\hat{\lambda}_0,\ \hat{\lambda}_1,\ \dots,\hat{\lambda}_{k-1}\}$ of $\hat{R}$; namely,

$$\mathcal{L}(k) = \ln\left[\frac{(\hat{\lambda}_0\hat{\lambda}_1\cdots\hat{\lambda}_{k-1})^{1/k}}{\dfrac{1}{k}(\hat{\lambda}_0 + \hat{\lambda}_1 + \cdots + \hat{\lambda}_{k-1})}\right]$$

The dimension K of the noise subspace is chosen to be that k that *minimizes* the functions $AIC(k)$ or $MDL(k)$. The above definition is equivalent to that of [1125], but produces the value of K instead of L. The function **aicmdl** takes as inputs the $M+1$ estimated eigenvalues in increasing order and the number N, and computes the values of the AIC and MDL functions. Once K is known, an estimate of the minimum eigenvalue can be obtained by

$$\hat{\sigma}_v^2 = \hat{\lambda}_{\min} = \frac{1}{K}(\hat{\lambda}_0 + \hat{\lambda}_1 + \cdots + \hat{\lambda}_{K-1}) \tag{14.5.3}$$

Next, we present some simulation examples. First, we compare the MUSIC method against the linear prediction method. We considered two uncorrelated equal-power plane waves incident on an array of 8 sensors ($M = 7$). The SNR of the waves, defined by $SNR_i = 10\log_{10}(P_i/\sigma_v^2)$, was -5 dB and their wavenumbers $k_1 = 0.2\pi$ and $k_2 = 0.4\pi$. For half-wavelength array spacing ($d = \lambda/2$), these correspond, through (14.3.8), to the angles of arrival $\theta_1 = 11.54^{\text{o}}$ and $\theta_2 = 23.58^{\text{o}}$.

The number of snapshots was $N = 500$. The snapshots were simulated using Eq. (14.3.10). Each $\mathbf{v}(n)$ was generated as a complex vector of $M+1$ zero-mean independent gaussian components of variance $\sigma_v^2 = 1$.

Note that to generate a zero-mean complex random variable v of variance σ_v^2, one must generate two zero-mean independent real random variables v_1 and v_2, each with variance $\sigma_v^2/2$ and set $v = v_1 + jv_2$; then, $E[v^*v] = E[v_1^2] + E[v_2^2] = 2(\sigma_v^2/2) = \sigma_v^2$. The amplitudes $A_i(n)$ were assumed to have only random phases; that is, $A_i(n) = (P_i)^{1/2}e^{j\phi_{in}}$, where ϕ_{in}, were independent angles uniformly distributed in $[0, 2\pi]$. The function **snap** takes as input an integer seed, generates a snapshot vector $\mathbf{y}$, and updates the seed. Successive calls to **snap**, in conjunction with the (complex version) of the function **sampcov**, can be used to generate the sample covariance matrix $\hat{R}$. In this particular example, we used the symmetrized version $\hat{R}_s$, because the two sources were uncorrelated.

Fig. 14.5.1 shows the MUSIC spectrum computed using Eq. (14.5.1) together with the LP spectrum $S_{LP}(k) = 1/|\mathbf{s}_k^\dagger\mathbf{a}|^2$, where $\mathbf{a} = \hat{R}_s^{-1}\mathbf{u}_0$. Because each term in the sum (14.5.1) arises from a unit-norm eigenvector, we have normalized the LP vector $\mathbf{a}$ also to unit norm for the purpose of plotting the two spectra on the same graph. Increasing the number of snapshots will improve the MUSIC spectrum because the covariance matrix $\hat{R}_s$ will become a better estimate of R, but it will not improve the LP spectrum because the theoretical LP spectrum does not perform well at low SNRs.

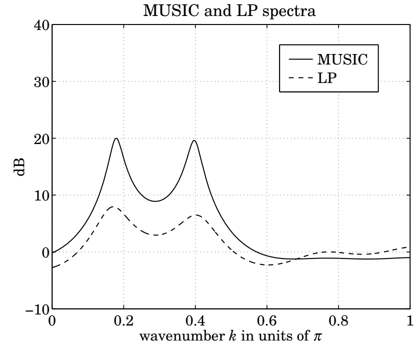


Fig. 14.5.1 MUSIC and LP spectra.

To facilitate the computation and manipulation of spectra, we have included the following small functions. The built-in function **norm** converts a vector a to a unit-norm vector and the function **freqz** computes the magnitude response squared, $|A(k)|^2 = |\mathbf{s}_k^\dagger\mathbf{a}|^2$, of an Mth order filter $\mathbf{a}$ at a specified number of equally-spaced frequency points within the right-half of the Nyquist interval, $0 \le k \le \pi$. It can be modified easily to include the entire Nyquist interval or any subinterval. The function **invresp** inverts a given spectrum, $S(k)\to 1/S(k)$. The functions **abs2db** and **db2abs** convert a spectrum from absolute units to decibels and back, $S(k)=10\log_{10}S(k)$. The function **select** picks out any eigenvector from the $M+1$ ones of the eigenvector matrix E. The function **music** computes Eq. (14.5.1) over a specified number of frequency points. It is built out of the functions **select**, **freqz**, and **invresp**.

In the second simulation example, we increased the SNR of the two plane waves to 10 dB and reduced the number of snapshots to $N=100$. The theoretical and empirical eigenvalues of R and $\hat{R}_s$, were found to be

i	0	1	2	3	4	5	6	7
λ_i	1	1	1	1	1	1	61.98	100.02
$\hat{\lambda}_i$	0.70	0.76	0.83	0.87	1.05	1.28	64.08	101.89

The values of the AIC and MDL functions were

k	1	2	3	4	5	6	7	8
$AIC(k)$	126.0	120.3	111.4	98.7	87.2	81.1	2544.2	3278.2
$MDL(k)$	145.1	138.3	127.4	111.9	94.4	77.0	1291.6	1639.1

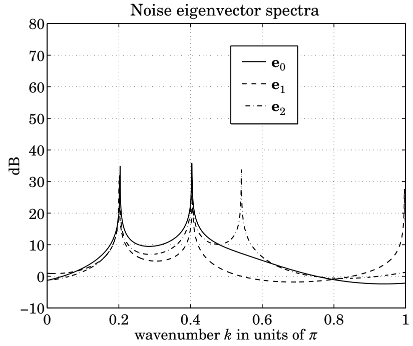


Fig. 14.5.2 Spectra of the first three noise subspace eigenvectors.

Both functions achieve their minimum value at $K=6$ and therefore, $L=M+1-K=2$. The estimated value of σ_v^2, computed by Eq. (14.5.3), was $\hat{\sigma}_v^2=0.915$. Fig. 14.5.2 shows the spectra of the first three noise subspace eigenvectors; namely, $S_i(k)=1/|E_i(k)|^2=1/|\mathbf{s}_k^\dagger\mathbf{e}_i|^2$, for $i=0,1,2$. We note the presence of a common set of peaks at the two desired wavenumbers and several spurious peaks. The spurious peaks are different, however, in each spectrum and therefore, the averaging operation will tend to eliminate them. The averaged MUSIC spectrum, based on all $K=6$ noise subspace eigenvectors, is plotted in Fig. 14.6.1 using the same scale as in Fig. 14.5.2.

The averaging operation has had two effects. First, the removal of all spurious peaks and second, the broadening and reduction in sharpness of the two desired peaks. This broadening is the result of statistical sampling; that is, using $\hat{R}$ instead of R, causes small biases in the peaks of individual eigenvectors about their true locations. These biases are not inherent in the theoretical method, as they are in the linear prediction case; they are statistical in nature and disappear in the limit of large number of snapshots. Fig. 14.6.1 also shows the performance of the minimum-norm method, which we discuss next. It appears to produce somewhat sharper peaks than MUSIC, but it can sometimes exhibit higher levels of spurious peaks.

14.6 Minimum-Norm Method

The minimum-norm method [1117] attempts to eliminate the effect of spurious zeros by pushing them inside the unit circle, leaving the L desired zeros on the circle. This is accomplished by finding a noise subspace vector $\mathbf{d}=[d_0,d_1,\dots,d_M]^T$ such that the corresponding eigenfilter $D(z)$ will have all its spurious zeros within the unit circle. This means that in the factorization (14.4.6), $D(z)=A(z)F(z)$, the spurious polynomial $F(z)$ must be chosen to have all its zeros strictly inside the unit circle, equivalently, $F(z)$ must be a minimum-phase polynomial. If $F(z)$ were the prediction-error filter of a linear prediction problem, then it would necessarily be a minimum-phase filter. Thus, the design strategy for $\mathbf{d}$ is to make $F(z)$ a linear prediction filter. This can be done by requiring that $\mathbf{d}$ have minimum norm subject to the constraint that its first coefficient be unity; that is,

$$\mathbf{d}^\dagger\mathbf{d}=\min\,,\quad \text{subject to}\quad \mathbf{u}_0^\dagger\mathbf{d}=d_0=1 \tag{14.6.1}$$

The minimization is carried over the noise subspace vectors. In the B basis (14.4.10), the vector $\mathbf{d}$ is expressed by $\mathbf{d}=B\mathbf{f}$, where $\mathbf{f}$ are the coefficients of $F(z)$, and the constraint equation becomes $\mathbf{u}_0^\dagger B\mathbf{f}=1$. With the exception of $\mathbf{b}_0$, all basis vectors $\mathbf{b}_i$ start with zero; therefore, $\mathbf{u}_0^\dagger B=[\mathbf{u}_0^\dagger\mathbf{b}_0,\mathbf{u}_0^\dagger\mathbf{b}_1,\dots,\mathbf{u}_0^\dagger\mathbf{b}_{K-1}]=[1,0,\dots,0]\equiv\mathbf{u}^\dagger$, that is, a K-dimensional unit vector. Therefore, in the B basis Eq. (14.6.1) becomes

$$\mathbf{d}^\dagger\mathbf{d}=\mathbf{f}^\dagger R_{aa}\,\mathbf{f}=\min\,,\quad \text{subject to}\quad \mathbf{u}^\dagger\mathbf{f}=1 \tag{14.6.2}$$

where we set $R_{aa}=B^\dagger B$. This is recognized as the Toeplitz matrix of autocorrelations of the filter $\mathbf{a}$, as defined in Eq. (1.19.5) of Sec. 1.19. For the 6×4 example above, we verify,

$$R_{aa}=B^\dagger B=\begin{bmatrix} R_{aa}(0) & R_{aa}(1)^* & R_{aa}(2)^* & 0\\ R_{aa}(1) & R_{aa}(0) & R_{aa}(1)^* & R_{aa}(2)^*\\ R_{aa}(2) & R_{aa}(1) & R_{aa}(0) & R_{aa}(1)^*\\ 0 & R_{aa}(2) & R_{aa}(1) & R_{aa}(0)\end{bmatrix}$$

where $R_{aa}(0)=|a_0|^2+|a_1|^2+|a_2|^2$, $R_{aa}(1)=a_1a_0^*+a_2a_1^*$, $R_{aa}(2)=a_2a_0^*$, and $R_{aa}(3)=0$. Note that the autocorrelation function of an order-M filter $\mathbf{a}$ vanishes for lags greater than $M+1$. It follows that Eq. (14.6.2) represents an ordinary linear prediction problem and its solution $\mathbf{f}$ will be a minimum-phase filter with all its zeros inside the unit circle. Up to a scale factor, we may write this solution as $\mathbf{f}=R_{aa}^{-1}\mathbf{u}=(B^\dagger B)^{-1}\mathbf{u}$. Writing $\mathbf{u}=B^\dagger\mathbf{u}_0$, we have $\mathbf{f}=(B^\dagger B)^{-1}B^\dagger\mathbf{u}_0$, and the solution for $\mathbf{d}$ becomes

$$\mathbf{d}=B\mathbf{f}=B(B^\dagger B)^{-1}B^\dagger\mathbf{u}_0=E_NE_N^\dagger\mathbf{u}_0 \tag{14.6.3}$$

This is the solution of criterion (14.6.1) up to a scale. Interestingly, the locations of the spurious zeros do not depend on the signal to noise ratios, but depend only on the desired zeros on the unit circle. This follows from the fact that the solution for $\mathbf{f}$ depends only on B. Using Eq. (14.4.13), we may also write $\mathbf{d}$ in terms of the signal subspace basis

$$\mathbf{d} = [I - E_S E_S^\dagger]\mathbf{u}_0 = [I - S(S^\dagger S)^{-1}S^\dagger]\mathbf{u}_0$$

Recall from Sec. 14.2 that this is the large-SNR limit of the LP solution. Noting that $E_N^\dagger \mathbf{u}_0$, is the complex conjugate of the top row of the eigenvector matrix E_N, we write Eq. (14.6.3) explicitly as a linear combination of noise subspace eigenvectors

$$\mathbf{d} = \sum_{i=0}^{K-1} E_{0i}^* \mathbf{e}_i \tag{14.6.4}$$

where E_{0i}^* the conjugate of the $0i$-th matrix element of E. The function **minorm** computes $\mathbf{d}$ using Eq. (14.6.4). The corresponding pseudospectrum estimate is defined as the inverse magnitude response of the filter $\mathbf{d}$

$$S_{\text{MIN}}(k) = \frac{1}{|D(k)|^2} = \frac{1}{|\mathbf{s}_k^\dagger \mathbf{d}|^2} \tag{14.6.5}$$

The practical implementation of this method requires the same two initial steps as MUSIC; namely, eigenanalysis of $\hat{R}$ and estimation of K. In Fig. 14.6.1, the minimum-norm spectrum was computed by calling the functions **minorm**. The vector $\mathbf{d}$ was normalized to unit norm to make a fair comparison with the MUSIC spectrum. Looking at the spectra is not the best way to evaluate this method because the spurious zeros—even though inside the unit circle—interact with the desired zeros to modify the shape of the spectrum. The minimum-norm method is better judged by comparing the theoretical and empirical zeros of the polynomial $D(z)$, computed from R and $\hat{R}$. They are shown in the following table. The first two zeros are the desired ones.

zeros of $D(z)$			
theoretical		empirical	
$\lvert z_i \rvert$	$\arg(z_i)/\pi$	$\lvert z_i \rvert$	$\arg(z_i)/\pi$
1.0000	0.2000	0.9989	0.2020
1.0000	0.4000	1.0059	0.4026
0.8162	−0.1465	0.8193	−0.1441
0.7810	−0.4251	0.7820	−0.4227
0.7713	−0.7000	0.7759	−0.6984
0.8162	0.7465	0.8188	0.7481
0.7810	−0.9749	0.7832	−0.9729

The main idea of the minimum-norm method was to separate the desired zeros from the spurious ones by pushing the latter inside the unit circle. In some applications of eigenvector methods, such as pole identification, the desired zeros lie themselves inside the unit circle (being the poles of a stable and causal system) and therefore, cannot be separated from the spurious ones. To separate them, we need a modification of the method that places all the spurious zeros to the outside of the unit circle. This can be done by replacing the vector $\mathbf{f}$ by its reverse $\mathbf{f}^R = J\mathbf{f}^*$, where J is the $K\times K$ reversing matrix. The resulting polynomial will be the reverse of $F(z)$, with all its zeros reflected to the outside of the unit circle. The reverse vector $\mathbf{f}^R$ is the backward prediction filter obtained by minimizing (14.6.2) subject to the constraint that its last element be unity. Using the reversal invariance of R_{aa}, namely, $JR_{aa}J = R_{aa}^*$, we find

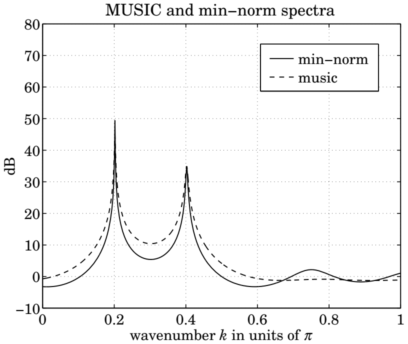


Fig. 14.6.1 MUSIC and min-norm spectra.

$$\mathbf{f}^R = J\mathbf{f}^* = J(R_{aa}^{-1})^*\mathbf{u} = R_{aa}^{-1}J\mathbf{u} = R_{aa}^{-1}\mathbf{v}$$

where $\mathbf{v} = J\mathbf{u} = [0,\dots,0,1]^T$ is the reverse of $\mathbf{u}$. With the exception of $\mathbf{b}_{K-1}$, the last element of all basis vectors $\mathbf{b}_i$ is zero. Denoting by $\mathbf{v}_0$, the reverse of $\mathbf{u}_0$, it follows that $\mathbf{v}_0^\dagger B = [0,0,\dots,0,a_L] = a_L\mathbf{v}^\dagger$. Thus, up to a scale factor, $\mathbf{v}$ can be replaced by $B^\dagger\mathbf{v}_0$, and hence, The vector $\mathbf{d}$ becomes

$$\mathbf{d} = B\mathbf{f}^R = B(B^\dagger B)^{-1}B^\dagger\mathbf{v}_0 = E_N E_N^\dagger \mathbf{v}_0 \tag{14.6.6}$$

Up to a scale, this is the minimum-norm vector subject to the constraint that its last element be unity; that is, $\mathbf{v}_0^\dagger\mathbf{d} = d_M = 1$. In terms of the matrix elements of the eigenvector matrix E it reads

$$\mathbf{d} = \sum_{i=0}^{K-1} E_{Mi}^* \mathbf{e}_i \tag{14.6.7}$$

where E_{Mi}^* is the conjugate of the last row of E. The spurious zeros of this vector will lie outside the unit circle. We may refer to this method as the modified minimum-norm method.

14.7 Reduced-Order Method

The basis B of the noise subspace has very special structure, being constructed in terms of the delayed replicas of the same reduced-order vector $\mathbf{a}$. It is evident from Eq. (14.4.11) that $\mathbf{a}$ can be extracted from any column $\mathbf{b}_i$ or B by advancing it by i units. The B basis is linearly related to the orthonormal eigenvector basis by $B = E_N C$ with some $K\times K$

invertible matrix C. Thus, the vector $\mathbf{b}_i$ is expressible as a linear combination of the noise subspace eigenvectors

$$\mathbf{b}_i = \sum_{j=0}^{K-1} \mathbf{e}_j C_{ji}\,, \quad i = 0, 1, \dots, K-1$$

This vector has a total of $K-1$ vanishing coefficients, namely, the first i and the last $K-1-i$ coefficients. Component-wise, we may write $b_{im} = 0$, for $0 \le m \le i-1$ and for $i+L+1 \le m \le M$. This vector may be specified up to an overall scale factor because we are interested only in the zeros of the reduced-order vector $\mathbf{a}$. Therefore, we may arbitrarily fix one of the coefficients C_{ji} to unity. For example, we may single out the 0th eigenvector:

$$\mathbf{b}_i = \mathbf{e}_0 + \sum_{j=1}^{K-1} \mathbf{e}_j C_{ji} \tag{14.7.1}$$

If $\mathbf{e}_0$ happens to be absent from the sum, we may single out $\mathbf{e}_1$ and so on. The coefficient b_{ii} will no longer be unity, but may be normalized so later. The $K-1$ unknown coefficients C_{ji}, $j = 1, 2, \dots, K-1$ can be determined by the $K-1$ conditions that the first i and last $K-1-i$ coefficients of $\mathbf{b}_i$ be zero. Written in terms of the matrix elements of the eigenvector matrix E, these conditions read for each $i = 0, 1, \dots, K-1$:

$$E_{m0} + \sum_{j=1}^{K-1} E_{mj} C_{ji} = 0\,, \quad \text{for} \quad 0 \le m \le i-1 \quad \text{and} \quad i+L+1 \le m \le M \tag{14.7.2}$$

Thus, solving the linear Eqs. (14.7.2) for the coefficients C_{ji} and substituting in Eq. (14.7.1), we obtain $\mathbf{b}_i$ and, advancing it by i units, the reduced-order vector $\mathbf{a}$. Because $B_i(z) = z^{-i}A(z)$, the polynomial $B_i(z)$ has no spurious zeros. In effect, forming the linear combination Eq. (14.7.1) of noise subspace eigenvectors removes the spurious zeros completely by placing them at the *origin* of the z-plane. In a sense, this procedure carries the philosophy of the minimum-norm method further.

When the theoretical R is replaced by the empirical $\hat{R}$ and the corresponding E_N is replaced by the estimated $\hat{E}_N$, it is no longer possible to linearly transform the basis $\hat{E}_N$ to a B basis constructed from a single reduced-order vector $\mathbf{a}$. It is still possible, however, to form linear combinations of the estimated eigenvectors.

$$\hat{\mathbf{b}}_i = \sum_{j=0}^{K-1} \hat{\mathbf{e}}_j C_{ji}\,, \quad i = 0, 1, \dots, K-1 \tag{14.7.3}$$

such that the resulting vectors $\hat{\mathbf{b}}_i$ will have vanishing first i and last $K-1-i$ coefficients; that is, of the form

$$\hat{\mathbf{b}}_i = [\underbrace{0, \dots, 0}_{i \text{ zeros}}, 1, a_{i1}, \dots, a_{iL}, \underbrace{0, \dots, 0}_{K-1-i \text{ zeros}}]^T \tag{14.7.4}$$

This can be done by solving Eq. (14.7.2) with E replaced by its estimate, $\hat{E}$, obtained from $\hat{R}$. The resulting K reduced-order vectors $\mathbf{a}_i = [1, a_{i1}, \dots, a_{iL}]^T$, $i = 0, 1, \dots, K-1$, will not be the same necessarily. But, each can be considered to be an approximate estimate of the true reduced-order vector $\mathbf{a}$, and its L zeros will be estimates of the true desired zeros.

It turns out that individually none of the $\mathbf{a}_i$ is a particularly good estimate of $\mathbf{a}$. They may be combined, however, to produce a better estimate. This is analogous to MUSIC, where individual spectra of noise eigenvectors are not good, but combining them by averaging produces a better spectrum. To see how we may best combine the $\mathbf{a}_i$, we form a new basis of the estimated noise subspace in terms of the vectors $\hat{\mathbf{b}}_i$, namely, $\hat{B} = [\hat{\mathbf{b}}_0, \hat{\mathbf{b}}_1, \dots, \hat{\mathbf{b}}_{K-1}]$. For our 6×4 example, we have

$$\hat{B} = [\hat{\mathbf{b}}_0, \hat{\mathbf{b}}_1, \hat{\mathbf{b}}_2, \hat{\mathbf{b}}_3] = \begin{bmatrix} 1 & 0 & 0 & \\ a_{01} & 1 & 0 & 0 \\ a_{02} & a_{11} & 1 & 0 \\ 0 & a_{12} & a_{21} & 1 \\ 0 & 0 & a_{22} & a_{31} \\ 0 & 0 & 0 & a_{32} \end{bmatrix}$$

The linear transformations (14.7.3) may be written compactly as $\hat{B} = \hat{E}_N C$. Note that $\hat{B}^\dagger \hat{B}$ is no longer Toeplitz and therefore, the LP solution $\mathbf{f}$ of (14.6.2) will not necessarily have minimum phase. Thus, the empirical minimum-norm solution can have spurious zeros outside or near the unit circle. Because the basis $\hat{B}$ is an estimate of the true B, we may try to fit $\hat{B}$ to a matrix of the type B having the special structure (14.4.11) by minimizing the distance between the two matrices according to some matrix norm. For example, we may minimize the Frobenius matrix distance [1166]:

$$\|\hat{B} - B\|^2 = \operatorname{tr}\big[(\hat{B} - B)^\dagger (\hat{B} - B)\big] = \sum_{i=0}^{K-1} \|\hat{\mathbf{b}}_i - \mathbf{b}_i\|^2 = \min$$

Because $\hat{\mathbf{b}}_i$ and $\mathbf{b}_i$ are the delayed versions of the reduced-order vectors $\mathbf{a}_i$ and $\mathbf{a}$, it follows that $\|\hat{\mathbf{b}}_i - \hat{\mathbf{b}}_i\|^2 = \|\mathbf{a}_i - \mathbf{a}\|^2$. Therefore,

$$\|\hat{B} - B\|^2 = \operatorname{tr}\big[(\hat{B} - B)^\dagger (\hat{B} - B)\big] = \sum_{i=0}^{K-1} \|\mathbf{a}_i - \mathbf{a}\|^2 = \min \tag{14.7.5}$$

Minimizing with respect to $\mathbf{a}$ gives the result:

$$\hat{\mathbf{a}} = \frac{1}{K}\sum_{i=0}^{K-1} \mathbf{a}_i\,, \qquad \hat{A}(z) = \frac{1}{K}\sum_{i=0}^{K-1} A_i(z) \tag{14.7.6}$$

that is, the average of the K filters. Thus, we obtain the following *reduced-order* or, *reduced-MUSIC* algorithm [1139]:

1. Solve the eigenproblem for the estimated covariance matrix $\hat{R}$.
2. Using the estimated noise subspace eigenvectors, solve (14.7.2) for the coefficients C_{ji} and using Eq. (14.7.3) obtain the basis vectors $\hat{\mathbf{b}}_i$ and hence the reduced-order vectors $\mathbf{a}_i$, $i = 0, 1, \dots, K-1$.

3. Use the average (14.7.6) to get an estimate $\hat{A}(z)$ of the reduced-order polynomial $A(z)$. Obtain estimates of the desired zeros by a root-finding procedure on $\hat{A}(z)$, or, by finding the peaks in the pseudospectrum

$$\hat{S}(k)=\frac{1}{|\hat{A}(k)|^2}=\frac{1}{|\mathbf{s}_k^\dagger\hat{\mathbf{a}}|^2} \qquad (14.7.7)$$

The MATLAB function **rmusic** implements this algorithm. Fig. 14.7.1 shows a comparison between the reduced-order algorithm and MUSIC for the same example considered in Fig. 14.6.1, where, again, for the purposes of comparison the vector $\hat{\mathbf{a}}$ was normalized to unit norm. As in the case of MUSIC, the spectrum of any individual reduced-order vector $\mathbf{a}_i$ is not good, but the spectrum based on the average $\hat{\mathbf{a}}$ is better. This can be appreciated by comparing the two zeros $(L=2)$ of the six $(K=6)$ individual filters $\hat{A}_i(z)$, $i=0,1,\dots,5$ with the two zeros of the averaged polynomial $\hat{A}(z)$ and with the theoretical zeros. They are shown in the table below.

zeros	$\hat{A}_0$	$\hat{A}_1$	$\hat{A}_2$	$\hat{A}_3$	$\hat{A}_4$	$\hat{A}_5$	$\hat{A}$	A
$\lvert z_1\rvert$	0.976	1.032	0.964	1.038	0.969	1.025	0.999	1.000
$\arg(z_1)/\pi$	0.197	0.203	0.199	0.199	0.203	0.197	0.201	0.200
$\lvert z_2\rvert$	1.056	0.944	1.115	0.896	1.059	0.947	1.002	1.000
$\arg(z_2)/\pi$	0.393	0.407	0.402	0.402	0.407	0.393	0.399	0.400

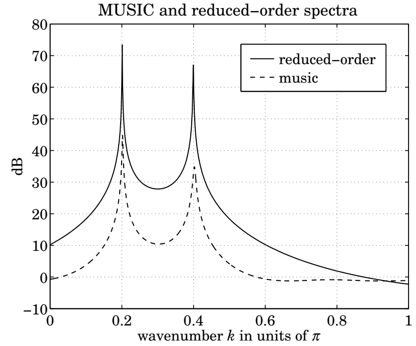


Fig. 14.7.1 MUSIC and reduced-order method.

An alternative method of combining the K estimates is as follows [1163]. Form the $(L+1)\times K$ matrix $A=[\mathbf{a}_0,\mathbf{a}_1,\dots,\mathbf{a}_{K-1}]$ and note that if the $\mathbf{a}_i$ were computed on the basis of the theoretical covariance matrix R, then A would have rank one because each $\mathbf{a}_i$ would be exactly equal to $\mathbf{a}$. But if the empirical matrix $\hat{R}$ is used, then the matrix A will only approximately have rank one, in the sense of its singular value decomposition (SVD). Thus, we may replace A by its rank-one SVD approximant, namely, the rank-one matrix closest to A with respect to the Frobenius or Euclidean matrix norms. This amounts to finding the largest eigenvalue of the $(L+1)\times(L+1)$ matrix

$$AA^\dagger=\sum_{i=0}^{K-1}\mathbf{a}_i\mathbf{a}_i^\dagger \qquad (14.7.8)$$

and choosing the corresponding eigenvector to be the estimate of $\mathbf{a}$. This eigenvector is expressible as a weighted sum of the $\mathbf{a}_i$ but with different weights than Eq. (14.7.6). To see this, let σ and $\hat{\mathbf{a}}$ be the largest eigenvalue and eigenvector of $AA^\dagger$. Using $AA^\dagger\hat{\mathbf{a}}=\sigma\hat{\mathbf{a}}$, and defining $\mathbf{w}=\sigma^{-1}A^\dagger\hat{\mathbf{a}}$, we find

$$\hat{\mathbf{a}}=A\mathbf{w}=\sum_{i=0}^{K-1}w_i\mathbf{a}_i \qquad (14.7.9)$$

where w_i are the components of $\mathbf{w}=[w_0,w_1,\dots,w_{K-1}]^T$. The constraint that $\hat{\mathbf{a}}$ and $\mathbf{a}_i$, have first coefficients of unity implies the normalization condition $\sum_{i=0}^{K-1}w_i=1$.

Even though this method is computationally more complex than Eq. (14.7.6), it allows one to judge the quality of the resulting estimate. This may be done by inspecting the relative magnitudes of the singular values of A, equivalently, the $L+1$ eigenvalues of $AA^\dagger$. Theoretically, all but the maximum eigenvalue must be zero. Applying this method to the above simulation example, we find the estimated zeros:

$$z_1=0.9985\,e^{j0.2011\pi}\,,\quad z_2=1.0037\,e^{j0.3990\pi}$$

and the theoretical and empirical SVD values of the matrix A:

theoretical	5.8059	0	0
empirical	5.8139	0.1045	0.0187

14.8 Maximum Likelihood Method

The maximum likelihood method is not, strictly speaking, an eigenvector method; however, some of the ideas we have been discussing apply to it. The method determines the plane wave frequencies and amplitudes by fitting them *directly* to the measured snapshot data using a criterion, such as maximum likelihood or least-squares. Each snapshot is modeled according to Eq. (14.3.10), which can be written compactly as

$$\mathbf{y}(n)=[\mathbf{s}_{k_1}^*,\dots,\mathbf{s}_{k_L}^*]\begin{bmatrix}A_1(n)\\ \vdots\\ A_L(n)\end{bmatrix}+\mathbf{v}(n)=S^*\mathbf{A}(n)+\mathbf{v}(n) \qquad (14.8.1)$$

The unknown amplitudes $\mathbf{A}(n)$ and wavenumbers k_i, $i=1,2,\dots,L$ are treated as deterministic parameters to be fitted to the snapshot data $Y=\{\mathbf{y}(n),\,0\le n\le N-1\}$. The maximum likelihood estimates of these parameters are obtained by maximizing the joint density of the snapshots, $p(Y)=\max$. If the wave parameters are deterministic, then the randomness in $\mathbf{y}(n)$ arises only from $\mathbf{v}(n)$. Assuming that $\mathbf{v}(n)$ are complex gaussian (see Problem 14.16) and independent, the joint density of Y is the product of marginal densities:

$$\begin{aligned}p(Y)&=\prod_{n=0}^{N-1}p\big(\mathbf{v}(n)\big)=\frac{1}{(\pi\sigma_v^2)^{N(M+1)}}\exp\left[-\frac{1}{\sigma_v^2}\sum_{n=0}^{N-1}\|\mathbf{v}(n)\|^2\right]\\ &=\frac{1}{(\pi\sigma_v^2)^{N(M+1)}}\exp\left[-\frac{1}{\sigma_v^2}\sum_{n=0}^{N-1}\|\mathbf{y}(n)-S^*\mathbf{A}(n)\|^2\right]\end{aligned}$$

Thus, under gaussian statistics, the maximum likelihood criterion is equivalent to the least-squares minimization criterion:

$$J=\sum_{n=0}^{N-1}\|\mathbf{y}(n)-S^*\mathbf{A}(n)\|^2=\min \tag{14.8.2}$$

According to the general discussion of [1165], the simultaneous minimization of J with respect to k_i and $\mathbf{A}(n)$ can be done in two steps. First, minimize with respect to the amplitudes $\mathbf{A}(n)$ and then, minimize with respect to the wavenumbers k_i. Setting the gradients with respect to $\mathbf{A}(n)$ to zero, we obtain

$$\frac{\partial J}{\partial \mathbf{A}(n)}=-S^\dagger\big[\mathbf{y}(n)^*-S\mathbf{A}^*(n)\big]=0 \quad\Rightarrow\quad \mathbf{A}(n)^*=(S^\dagger S)^{-1}S^\dagger\mathbf{y}(n)^*$$

Inserting this solution into Eq. (14.8.2), we obtain

$$J=\sum_{n=0}^{N-1}\|\mathbf{y}(n^*)-S\mathbf{A}(n)^*\|^2=\sum_{n=0}^{N-1}\big|\big|[I-S(S^\dagger S)^{-1}S^\dagger]\mathbf{y}(n)^*\big|\big|^2$$

Using Eq. (14.4.13), we may rewrite it in terms of the projector onto the noise subspace, namely, $P_N=B(B^\dagger B)^{-1}B^\dagger=I-S(S^\dagger S)^{-1}S^\dagger$

$$J=\sum_{n=0}^{N-1}\big|\big|B(B^\dagger B)^{-1}B^\dagger\mathbf{y}(n)^*\big|\big|^2=\sum_{n=0}^{N-1}\big|\big|P_N\mathbf{y}(n)^*\big|\big|^2$$

Using the projection property $P_N^\dagger P_N=P_N$, and the definition (14.4.14) of the sample covariance matrix, we find

$$J=\sum_{n=0}^{N-1}\mathbf{y}(n)^TP_N\mathbf{y}(n)^*=\mathrm{tr}\left[\sum_{n=0}^{N-1}P_N\mathbf{y}(n)^T\mathbf{y}(n)^*\right]=N\,\mathrm{tr}[P_N\hat{R}]$$

The minimization of J with respect to the coefficients of the reduced-order vector $\mathbf{a}$ is a highly nonlinear problem. It may be solved, however, iteratively by the solution of a succession of simpler problems, by the following procedure [1141–1143,1159,1161]. Write $\mathbf{y}(n)^TB=[\mathbf{y}(n)^T\mathbf{b}_0,\mathbf{y}(n)^T\mathbf{b}_1,\dots,\mathbf{y}(n)^T\mathbf{b}_{K-1}]$ and note that $\mathbf{y}(n)^T\mathbf{b}_i=\mathbf{a}^T\mathbf{y}_i(n)$, where $\mathbf{y}_i(n)$ is the $(L+1)$-dimensional portion of $\mathbf{y}(n)$ starting at the ith position, namely,

$$\mathbf{y}_i(n)=\big[y_i(n),y_{i+1}(n),\dots,y_{i+L}(n)\big]^T,\quad i=0,1,\dots,K-1$$

Then, $\mathbf{y}(n)^TB=\mathbf{a}^T[\mathbf{y}_0(n),\mathbf{y}_1(n),\dots,\mathbf{y}_{K-1}(n)]\equiv\mathbf{a}^TY(n)$. And, J can be written as

$$J=\sum_{n=0}^{N-1}\mathbf{y}(n)^TB(B^\dagger B)^{-1}B^\dagger\mathbf{y}(n)^*=\mathbf{a}^T\left[\sum_{n=0}^{N-1}Y(n)\,(B^\dagger B)^{-1}Y(n)^\dagger\right]\mathbf{a}^*$$

The minimization of J is obtained by solving the succession of problems, for $i=1,2,\dots,$

$$J_i=\mathbf{a}_i^T\left[\sum_{n=0}^{N-1}Y(n)\,(B_{i-1}^\dagger B_{i-1})^{-1}Y(n)^\dagger\right]\mathbf{a}_i^*=\min \tag{14.8.3}$$

where $B_{i-1}^\dagger B_{i-1}$ is constructed from the solution $\mathbf{a}_{i-1}$ of the previous iteration. The iteration is initialized by $\mathbf{a}_0=[1,0,\dots,0]^T$, which gives $B_0^\dagger B_0=I_K$. At each iteration, Eq. (14.8.3) is subject to an appropriate constraint on $\mathbf{a}_i$ such as that its first coefficient be unity, or, that its zeros lie on the unit circle. Note that $B^\dagger B$ is Toeplitz and therefore, its inverse can be computed efficiently by the Levinson recursion.

14.9 ESPRIT Method

There exist a number of eigenvector methods that employ two or more sets of snapshot measurements obtained from two or more arrays related to each other either by translation or by rotation. Examples are the estimation of signal parameters via rotational invariance techniques (ESPRIT) method [1145], the covariance difference method [1135–1138], and the spatial smoothing method for dealing with coherent signals [1119,1126].

Consider two arrays related to each other by an overall translation by distance Δ along the x-axis. The effect of translation shows up as an overall phase change in each direction vector. For example, the value of a wave on the x-axis with respect to the original and the translated x-axes will be:

$$A_1e^{-jk_xx}\to A_1e^{-jk_x(x+\Delta)}=A_1e^{-jk_xx}e^{-jk_x\Delta}$$

Setting $x_m=md$ and letting $\delta=\Delta/d$ be the displacement in units of d, we obtain at the original and translated mth array elements

$$A_1e^{-jk_1m}\to A_1e^{-jk_1m}e^{-jk_1\delta}$$

or, in terms of the direction vectors

$$A_1\mathbf{s}_1^*\to A_1\mathbf{s}_1^*e^{-jk_1\delta}$$

It follows that the matrix $S=[\mathbf{s}_{k_1},\dots,\mathbf{s}_{k_L}]$ transforms under translation as

$$S\to SD_\delta\,,\quad D_d=\mathrm{diag}\big\{e^{jk_1\delta},\,e^{jk_2\delta},\,\dots,\,e^{jk_L\delta}\big\} \tag{14.9.1}$$

Therefore, the snapshot measurements at the original and translated arrays are

$$\begin{aligned}\mathbf{y}(n)&=S^*\mathbf{A}(n)+\mathbf{v}(n)\\ \mathbf{y}_\delta(n)&=S^*D_\delta^*\mathbf{A}(n)+\mathbf{v}_\delta(n)\end{aligned} \tag{14.9.2}$$

The covariance and cross-covariance matrices are

$$\begin{aligned}R_{yy}&=E[\mathbf{y}(n)^*\mathbf{y}(n)^T]=SPS^\dagger+\sigma_v^2I\\ R_{y_\delta y_\delta}&=E[\mathbf{y}_\delta(n)^*\mathbf{y}_\delta(n)^T]=SD_\delta PD_\delta^\dagger S^\dagger+\sigma_v^2I\end{aligned} \tag{14.9.3}$$

$$R_{yy_\delta}=E[\mathbf{y}(n)^*\mathbf{y}_\delta(n)^T]=SPD_\delta^\dagger S^\dagger \tag{14.9.4}$$

where we used $E[\mathbf{v}_\delta(n)^*\mathbf{v}_\delta(n)^T]=E[\mathbf{v}(n)^*\mathbf{v}(n)^T]=\sigma_v^2I$ and $E[\mathbf{v}(n)^*\mathbf{v}_\delta(n)^T]=0$.

The ESPRIT method works with the *matrix pencil*, $C(\lambda)= C - \lambda C_\delta$, defined by the pair of matrices

$$C = R_{yy} - \sigma_v^2 I = SPS^\dagger\,, \qquad C_\delta = R_{yy_\delta} = SPD_\delta^\dagger S^\dagger \tag{14.9.5}$$

The generalized eigenvalues of this matrix pencil are, by definition [102], the solutions of $\det(C - \lambda C_\delta)= 0$, and the corresponding generalized eigenvectors satisfy $C\mathbf{e} = \lambda C_\delta \mathbf{e}$. The ESPRIT method is based on the observation that the *nonzero* generalized eigenvalues of $C(\lambda)$ are simply

$$\lambda_i = e^{jk_i\delta}\,, \quad i = 1, 2, \dots, L \tag{14.9.6}$$

and therefore, the desired wavenumbers k_i can be extracted from the knowledge of the λ_i. Note that $\lambda = 0$ is a generalized eigenvalue because $\det(C)= \det(SPS^\dagger)= 0$. This follows from the fact that $SPS^\dagger$ is an $(M + 1)\times(M + 1)$ matrix of rank $L < M + 1$. The generalized eigenvectors corresponding to $\lambda = 0$ are the vectors in the null space of $SPS^\dagger$; namely, they satisfy $SPS^\dagger\mathbf{e} = 0$, or, equivalently, $S^\dagger\mathbf{e} = 0$. These are the noise subspace eigenvectors of R_{yy}. Next, we show that the only nonzero generalized eigenvalues are those in Eq. (14.9.6). The corresponding generalized eigenvector $\mathbf{e}$ must satisfy

$$SPS^\dagger\mathbf{e} = \lambda SPD_\delta^\dagger S^\dagger\mathbf{e}$$

Multiplying both sides by $S^\dagger$ and removing the common matrix factor $(S^\dagger S)P$, we obtain $S^\dagger\mathbf{e} = \lambda D_\delta^\dagger S^\dagger\mathbf{e}$. Using the fact that $D_\delta^\dagger = D_\delta^{-1}$, and defining the L-dimensional vector $\mathbf{f} = S^\dagger\mathbf{e}$, we obtain

$$D_\delta\mathbf{f} = \lambda\mathbf{f}$$

Clearly, if $\mathbf{e}$ is not in the noise subspace, then $\mathbf{f} = S^\dagger\mathbf{e} \neq 0$; therefore, λ must be an eigenvalue of D_δ, which is already diagonal. This proves Eq. (14.9.6). The eigenvectors of D_δ will be the L-dimensional unit vectors; that is, the columns of the $L\times L$ unit matrix, $\mathbf{f}_i = \mathbf{u}_i$, $i = 1, 2, \dots, L$. The generalized eigenvectors will be $\mathbf{e}_i = S(S^\dagger S)^{-1}\mathbf{u}_i$. These are obtained by an argument similar to Eq. (14.3.15). Thus, the L columns of the matrix $S(S^\dagger S)^{-1}$ are simply the generalized eigenvectors corresponding to the generalized eigenvalues (14.9.6).

In the practical implementation of the method, we assume we have two sets of snapshots, $\mathbf{y}(n)$ and $\mathbf{y}_\delta(n)$, for $n = 0, 1, \dots, N - 1$, measured at the original and translated arrays. The covariance matrix R_{yy} is estimated by Eq. (14.4.14) and the cross-covariance matrix by

$$\hat{C}_\delta = \hat{R}_{yy_\delta} = \frac{1}{N}\sum_{n=0}^{N-1}\mathbf{y}(n)^*\mathbf{y}_\delta(n)^T$$

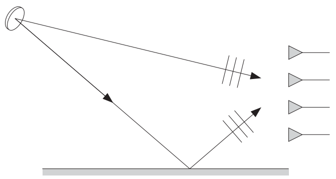
From the eigenproblem of $\hat{R}_{yy}$, we obtain an estimate of $\hat{\sigma}_v^2$, either as the minimum eigenvalue or, as the average of the eigenvalues of the noise subspace. Then, set $\hat{C} = \hat{R}_{yy} - \hat{\sigma}_v^2 I$ and solve the generalized eigenproblem for the pair $\{\hat{C}, \hat{C}_\delta\}$. The L generalized eigenvalues closest to the unit circle are used to extract estimates of the desired wavenumbers k_i by Eq. (14.9.6).

Unlike the minimum-norm and reduced-order methods that require equally spaced linear arrays, the MUSIC and ESPRIT methods can be applied to arrays of arbitrary geometry.

14.10 Spatial Smoothing

Eigenvector methods rely on the property that the noise subspace eigenvectors have at least L zeros on the unit circle at the desired frequency locations. As we saw in Sec. 14.3, this property requires that the $L\times L$ power matrix P have full rank equal to L. To repeat the argument, the condition $R\mathbf{a} = \sigma_v^2\mathbf{a}$ implies that $SPS^\dagger\mathbf{a} = 0$, but what we want is $S^\dagger\mathbf{a} = 0$. Multiplying by $\mathbf{a}^\dagger$, we obtain $(S^\dagger\mathbf{a})^\dagger P(S^\dagger\mathbf{a})= 0$, but this does not necessarily imply that $S^\dagger\mathbf{a} = 0$ unless P has full rank.

The case of diagonal P corresponds to mutually uncorrelated sources for the L plane waves. The case of a nondiagonal P of full rank implies that the sources are partially correlated. The case of a non-diagonal P with less than full rank implies that some or all of the sources are coherent with each other. This case commonly arises in multipath situations, as shown in the following diagram



To see how eigenvector methods fail if P does not have full rank, consider the worst case when all the sources are coherent, which means that the wave amplitudes $A_i(n)$ are all proportional to each other, say, $A_i(n)= c_i A_1(n)$, $i = 1, 2, \dots, L$, where the $c_i \neq 0$ (with $c_1 = 1$) are attenuation factors corresponding to the different paths. Compactly, we may write $\mathbf{A}(n)= A_1(n)\mathbf{c}$. Then, the power matrix becomes

$$P = E\big[\mathbf{A}(n)^*\mathbf{A}(n)^T\big] = E\big[|A_1(n)|^2\big]\mathbf{c}^*\mathbf{c}^T = P_1\mathbf{c}^*\mathbf{c}^T \tag{14.10.1}$$

It has rank one. The corresponding covariance matrix is

$$R = SPS^\dagger + \sigma_v^2 I = P_1 S\mathbf{c}^*\mathbf{c}^T S^\dagger + \sigma_v^2 I = P_1\mathbf{s}\mathbf{s}^\dagger + \sigma_v^2 I \tag{14.10.2}$$

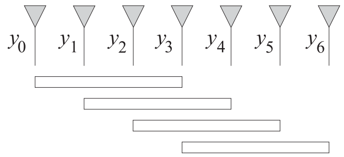
where $\mathbf{s} = S\mathbf{c}^*$. Similarly,

$$\mathbf{y}(n)= A_1(n)S^*\mathbf{c} + \mathbf{v}(n)= A_1(n)\mathbf{s}^* + \mathbf{v}(n)$$

Because R is a rank-one modification of the identity matrix, it will have a one-dimensional signal subspace spanned by $\mathbf{s}$ and a noise subspace of dimension $K = M + 1 - 1 = M$ spanned by the eigenvectors belonging to the minimum eigenvalue σ_v^2. Thus, although we have L different signals, the solution of the eigenproblem will result in a one-dimensional signal subspace. Moreover, the noise eigenvectors, will not necessarily have zeros at the L desired locations. This can be seen as follows. If $R\mathbf{a} = \sigma_v^2\mathbf{a}$, then $P_1\mathbf{s}\mathbf{s}^\dagger\mathbf{a} = 0$, or, $\mathbf{s}^\dagger\mathbf{a} = \mathbf{c}^T S^\dagger\mathbf{a} = 0$, which gives

$$\mathbf{c}^T S^\dagger\mathbf{a} = [c_1, \dots, c_L]\begin{bmatrix} A(k_1) \\ \vdots \\ A(k_L)\end{bmatrix} = \sum_{i=1}^{L} c_i A(k_i)= 0$$

This does not imply that the individual terms in the sum are zero. One solution to this problem is the method of spatial smoothing [1119,1126], which restores P to full rank, so that the eigenstructure methods can be applied as usual. The method is as follows. The given array of $M+1$ sensors is subdivided into J subarrays each having $\bar{M}+1$ sensors. The first subarray consists of the first $\bar{M}+1$ elements of the given array. Each subsequent subarray is obtained by shifting ahead one array element at a time, as shown in the following diagram



Formally, we define the J subarrays by

$$\bar{\mathbf{y}}_i(n)=\big[y_i(n),y_{i+1}(n),\dots,y_{i+\bar{M}}(n)\big]^T,\quad i=0,1,\dots,J-1 \tag{14.10.3}$$

where the bar indicates that the size of the subarray is $\bar{M}+1$. That is the $(\bar{M}+1)$-dimensional portion of $\mathbf{y}(n)$ starting at the ith array element. Using Eq. (14.9.2), we may write compactly

$$\bar{\mathbf{y}}_i(n)=\bar{S}^*D_i^*\mathbf{A}(n)+\bar{\mathbf{v}}_i(n)$$

where $\bar{S}$ is the same as S but of dimension $\bar{M}+1$. The matrix D_i is given by Eq. (14.9.1) with $\delta=i$, corresponding to translation by i units. The covariance matrix of the ith subarray will be

$$\bar{R}_i=E[\bar{\mathbf{y}}_i(n)^*\bar{\mathbf{y}}_i(n)^T]=\bar{S}D_iPD_i^\dagger\bar{S}^\dagger+\sigma_v^2\bar{I}$$

where $\bar{I}$ is the $(\bar{M}+1)$-dimensional identity matrix. The average of the subarray covariances is

$$\bar{R}=\frac{1}{J}\sum_{i=0}^{J-1}\bar{R}_i=\bar{S}\bar{P}\bar{S}^\dagger+\sigma_v^2\bar{I} \tag{14.10.4}$$

where

$$\bar{P}=\frac{1}{J}\sum_{i=0}^{J-1}D_iPD_i^\dagger \tag{14.10.5}$$

To be able to resolve L sources by the $(\bar{M}+1)$-dimensional eigenproblem (14.10.4), we must have $\bar{M}\geq L$, and the rank of $\bar{P}$ must be L. It has been shown [1126] that if the number of subarrays J is greater than the number of signals, $J\geq L$, then, $\bar{P}$ has full rank. If the J subarrays are to fit within the original array of length $M+1$, then we must have $M+1\geq(\bar{M}+1)+(J-1)$, that is, the length of the first subarray plus the $J-1$ subsequent shifts. Thus, $M+1\geq\bar{M}+J$. If both J and $\bar{M}$ are greater than L, then we must have $M+1\geq 2L$. Therefore, the price for restoring the rank of P is that we must use twice as long an array as in the ordinary full-rank case with L sources. A somewhat stronger result is that $J\geq L+1-\rho$, where ρ is the rank of P [1150]; equivalently, we have $J\geq\nu+1$, where $\nu=L-\rho$ is the nullity of P. This would give for the minimum number of array elements, $M+1\geq 2L+1-\rho$, [1127,1143,1150]. Following [1126], we derive the condition $J\geq L$ for the worst case, when all the signals are coherent. In that case, P has rank one $(\rho=1)$ and is given by Eq. (14.10.1); $\bar{P}$ becomes

$$\bar{P}=\frac{P_1}{J}\sum_{i=0}^{J-1}D_i\mathbf{c}^*\mathbf{c}^TD_i^\dagger=\frac{P_1}{J}\sum_{i=0}^{J-1}\mathbf{d}_i\mathbf{d}_i^\dagger\,,\quad \mathbf{d}_i=D_i\mathbf{c}^*$$

Writing $\sum_{i=0}^{J-1}\mathbf{d}_i\mathbf{d}_i^\dagger=DD^\dagger$, where $D=[\mathbf{d}_0,\mathbf{d}_1,\dots,\mathbf{d}_{J-1}]$, it follows that the rank of $\bar{P}$ is the same as the rank of D. The matrix element D_{li} is the lth component of the ith column; that is, $D_{li}=(\mathbf{d}_i)_l=c_l^*e^{jk_li}$. Thus, D can be written as the product, $D=C^*V$, of the diagonal matrix $C^*=\text{diag}\{c_1^*,\dots,c_L^*\}$ and the $L\times J$ Vandermonde matrix V with matrix elements $V_{li}=e^{jk_li}$; for example, if $L=3$ and $J=4$,

$$V=\begin{bmatrix}1 & e^{jk_1} & e^{2jk_1} & e^{3jk_1}\\ 1 & e^{jk_2} & e^{2jk_2} & e^{3jk_2}\\ 1 & e^{jk_3} & e^{2jk_3} & e^{3jk_3}\end{bmatrix}$$

The rank of Vandermonde matrices is always full; that is, it is the minimum of the column and row dimensions, $\min(L,J)$. It follows that the rank of $\bar{P}$ is equal to $\min(L,J)$, therefore, it is equal to L only if $J\geq L$.

To appreciate the mechanism by which the rank is restored, let us consider an example with two $(L=2)$ fully coherent sources. The minimum number of subarrays needed to *decohere* the sources is $J=L=2$. This implies $\bar{M}=M+1-J=M-1$. The covariance matrix of the full array is

$$R=P_1[\mathbf{s}_1,\mathbf{s}_2]\begin{bmatrix}c_1^*\\ c_2^*\end{bmatrix}[c_1,c_2]\begin{bmatrix}\mathbf{s}_1^\dagger\\ \mathbf{s}_2^\dagger\end{bmatrix}+\sigma_v^2I$$

The covariance matrices of the two subarrays are

$$\bar{R}_0=P_1[\bar{\mathbf{s}}_1,\bar{\mathbf{s}}_2]\begin{bmatrix}c_1^*\\ c_2^*\end{bmatrix}[c_1,c_2]\begin{bmatrix}\bar{\mathbf{s}}_1^\dagger\\ \bar{\mathbf{s}}_2^\dagger\end{bmatrix}+\sigma_v^2\bar{I}$$

$$\bar{R}_1=P_1[\bar{\mathbf{s}}_1,\bar{\mathbf{s}}_2]\begin{bmatrix}e^{jk_1}c_1^*\\ e^{jk_2}c_2^*\end{bmatrix}[e^{-jk_1}c_1,e^{-jk_2}c_2]\begin{bmatrix}\bar{\mathbf{s}}_1^\dagger\\ \bar{\mathbf{s}}_2^\dagger\end{bmatrix}+\sigma_v^2\bar{I}$$

Their average becomes

$$\bar{R}=\frac{1}{2}(\bar{R}_0+\bar{R}_1)=[\bar{\mathbf{s}}_1,\bar{\mathbf{s}}_2]\bar{P}\begin{bmatrix}\bar{\mathbf{s}}_1^\dagger\\ \bar{\mathbf{s}}_2^\dagger\end{bmatrix}+\sigma_v^2\bar{I}$$

where

$$\begin{aligned}\bar{P}&=\frac{P_1}{2}\begin{bmatrix}c_1^*\\ c_2^*\end{bmatrix}[c_1,c_2]+\frac{P_1}{2}\begin{bmatrix}e^{jk_1}c_1^*\\ e^{jk_2}c_2^*\end{bmatrix}[e^{-jk_1}c_1,e^{-jk_2}c_2]\\ &=P_1\begin{bmatrix}c_1^*c_1 & c_1^*c_2\big(1+e^{j(k_1-k_2)}\big)/2\\ c_1c_2^*\big(1+e^{j(k_2-k_1)}\big)/2 & c_2^*c_2\end{bmatrix}\end{aligned}$$

Clearly, $\bar{P}$ is non-singular. The presence of the translation phases makes the two column vectors $[c_1^*,c_2^*]^T$ and $[e^{jk_1}c_1^*,e^{jk_2}c_2^*]^T$ linearly independent. The determinant of $\bar{P}$ is easily found to be

$$\det\bar{P}=|c_1c_2|^2\sin^2\left(\frac{k_1-k_2}{2}\right)$$

Perhaps, an even simpler example is to consider the two quadratic forms

$$Q_0 = (f_1+f_2)^2 = \mathbf{f}^T\begin{bmatrix}1\\1\end{bmatrix}[1,1]\mathbf{f}\,,\quad \mathbf{f}=\begin{bmatrix}f_1\\f_2\end{bmatrix}$$

$$Q_1 = f_1^2 = \mathbf{f}^T\begin{bmatrix}1\\0\end{bmatrix}[1,0]\mathbf{f}$$

Separately, they have rank one, but their sum has full rank

$$Q = Q_0+Q_1 = (f_1+f_2)^2+f_1^2 = 2f_1^2+2f_1f_2+f_2^2 = \mathbf{f}^T\begin{bmatrix}2&1\\1&1\end{bmatrix}\mathbf{f}$$

where the 2×2 coefficient matrix has rank two, being the sum of two rank-one matrices defined by two linearly independent two-dimensional vectors

$$\begin{bmatrix}2&1\\1&1\end{bmatrix} = \begin{bmatrix}1\\1\end{bmatrix}[1,1]+\begin{bmatrix}1\\0\end{bmatrix}[1,0]$$

Such quadratic forms can be formed, for example, by $\mathbf{a}^\dagger SPS^\dagger\mathbf{a} = \mathbf{f}^\dagger P\mathbf{f}$, where $\mathbf{f} = S^\dagger\mathbf{a}$. In the practical implementation of the method, the subarray covariances are computed by sample averages over N snapshots; that is,

$$\bar{R}_i = \frac{1}{N}\sum_{n=0}^{N-1}\bar{\mathbf{y}}_i(n)^*\bar{\mathbf{y}}_i(n)^T$$

and then forming the average

$$\bar{R} = \frac{1}{J}\sum_{i=0}^{J-1}\bar{R}_i$$

In addition to spatial smoothing, there exist other methods for dealing with the problem of coherent signal sources [1147,1148,1151,1152].

14.11 Asymptotic Properties

Statistically, the sample covariance matrix $\hat{R}$ approximates the theoretical R, and therefore, the linear predictor based on $\hat{R}$ will approximate the one based on R. Similarly, the eigenstructure of $\hat{R}$ will approximate that of R. In this section, we derive the asymptotic statistical properties that justify such approximations [1179–1205].

The basic technique for deriving asymptotic results is to perform a linearization of the empirical solution about the theoretical one and then use the asymptotic statistical properties of $\hat{R}$. In Sec. 1.6, we obtained the asymptotic covariance of $\hat{R}$ for a large number of snapshots N:

$$E[\Delta R_{ij}\Delta R_{kl}] = \frac{1}{N}(R_{ik}R_{jl}+R_{il}R_{jk}) \tag{14.11.1}$$

where $\Delta R = \hat{R}-R$ is the deviation of $\hat{R}$ from its mean. This was valid in the real-valued case; the complex-valued version will be considered shortly. The normal equations of linear prediction based on $\hat{R}$ and R are

$$\hat{R}\hat{\mathbf{a}} = \hat{E}\mathbf{u}_0\,,\quad \hat{\mathbf{a}} = \begin{bmatrix}1\\\hat{\boldsymbol{\alpha}}\end{bmatrix}\quad\text{and}\quad R\mathbf{a} = E\mathbf{u}_0\,,\quad \mathbf{a} = \begin{bmatrix}1\\\boldsymbol{\alpha}\end{bmatrix}$$

where $\hat{E}$ and E are the minimized values of the mean-square prediction errors given by $\hat{E} = \hat{\mathbf{a}}^T\hat{R}\hat{\mathbf{a}}$ and $E = \mathbf{a}^TR\mathbf{a}$. Setting $\hat{\mathbf{a}} = \mathbf{a}+\Delta\mathbf{a}$ and $\hat{E} = E+\Delta E$, we obtain

$$(R+\Delta R)(\mathbf{a}+\Delta\mathbf{a}) = (E+\Delta E)\mathbf{u}_0 \quad\Rightarrow\quad R(\Delta\mathbf{a})+(\Delta R)\mathbf{a} = (\Delta E)\mathbf{u}_0 \tag{14.11.2}$$

where we kept only the first-order terms. Because $\hat{\mathbf{a}}$ and $\mathbf{a}$ have first coefficient of unity, $\Delta\mathbf{a} = \hat{\mathbf{a}}-\mathbf{a}$ will have zero first coefficient, that is, $\mathbf{u}_0^T(\Delta\mathbf{a}) = 0$. Multiplying both sides of Eq. (14.11.2) by $\mathbf{a}^T$, we obtain $\mathbf{a}^TR(\Delta\mathbf{a})+\mathbf{a}^T(\Delta R)\mathbf{a} = \Delta E$. Using the normal equations for $\mathbf{a}$, we have $\mathbf{a}^TR(\Delta\mathbf{a}) = E\mathbf{u}_0^T(\Delta\mathbf{a}) = 0$. Thus, $\Delta E = \mathbf{a}^T(\Delta R)\mathbf{a}$. Solving Eq. (14.11.2) for $\Delta\mathbf{a}$ and using $R^{-1}\mathbf{u}_0 = E^{-1}\mathbf{a}$, we find

$$\Delta\mathbf{a} = E^{-1}(\Delta E)\mathbf{a}-R^{-1}(\Delta R)\mathbf{a}\,,\quad \Delta E = \mathbf{a}^T(\Delta R)\mathbf{a} \tag{14.11.3}$$

For the purpose of computing the asymptotic covariances of $\Delta\mathbf{a}$ and ΔE, it proves convenient to express Eq. (14.11.3) in terms of the vector $\delta\mathbf{a}\equiv(\Delta R)\mathbf{a}$. Then,

$$\Delta\mathbf{a} = E^{-1}(\Delta E)\mathbf{a}-R^{-1}(\delta\mathbf{a})\,,\quad \Delta E = \mathbf{a}^T(\delta\mathbf{a}) \tag{14.11.4}$$

Using Eq. (14.11.1), we find for the covariance of $\delta\mathbf{a}$

$$\begin{aligned}E[\delta a_i\delta a_k] &= E[\sum_j\Delta R_{ij}a_j\sum_l\Delta R_{kl}a_l] = \sum_{jl}E[\Delta R_{ij}\Delta R_{kl}]a_ja_l\\ &= \frac{1}{N}\sum_{jl}(R_{ik}R_{jl}+R_{jk}R_{il})a_ja_l = \frac{1}{N}[R_{ik}(\mathbf{a}^TR\mathbf{a})+(R\mathbf{a})_i(\mathbf{a}^TR)_k]\end{aligned}$$

or,

$$E[\delta\mathbf{a}\,\delta\mathbf{a}^T] = \frac{1}{N}[ER+R\mathbf{a}\,\mathbf{a}^TR] \tag{14.11.5}$$

Writing $\Delta E = \delta\mathbf{a}^T\mathbf{a}$, we find

$$E[\delta\mathbf{a}\,\Delta E] = E[\delta\mathbf{a}\,\delta\mathbf{a}^T]\mathbf{a} = \frac{1}{N}[ER+R\mathbf{a}\,\mathbf{a}^TR]\mathbf{a} = \frac{1}{N}[ER\mathbf{a}+R\mathbf{a}(\mathbf{a}^TR\mathbf{a})] = \frac{2E}{N}R\mathbf{a}$$

Using this result, we find for the asymptotic variance of $\hat{E}$:

$$E[(\Delta E)^2] = \mathbf{a}^TE[\delta\mathbf{a}\,\Delta E] = \frac{2E}{N}\mathbf{a}^TR\mathbf{a} = \frac{2E^2}{N} \tag{14.11.6}$$

This generalizes Eq. (1.16.4). Similarly, we find for the cross-covariance between $\hat{E}$ and $\hat{\mathbf{a}}$:

$$E[\Delta\mathbf{a}\,\Delta E] = E[(E^{-1}\Delta E\mathbf{a}-R^{-1}\delta\mathbf{a})\Delta E] = E^{-1}E[(\Delta E)^2]\mathbf{a}-R^{-1}E[\delta\mathbf{a}\,\Delta E]\,,\quad\text{or,}$$

$$E[\Delta\mathbf{a}\,\Delta E]= E^{-1}\frac{2E^2}{N}\mathbf{a}-R^{-1}\big(\frac{2E}{N}R\mathbf{a}\big)=0 \tag{14.11.7}$$

Finally, we find for the covariance of the predictor $\hat{\mathbf{a}}$

$$\begin{aligned}E[\Delta\mathbf{a}\,\Delta\mathbf{a}^T]&= E\big[\Delta\mathbf{a}(E^{-1}\Delta E\mathbf{a}^T-\delta\mathbf{a}R^{-1})\big]=-E[\Delta\mathbf{a}\,\delta\mathbf{a}^T]R^{-1}\\ &=-E\big[(E^{-1}\mathbf{a}\,\Delta E-R^{-1}\delta\mathbf{a})\,\delta\mathbf{a}^T\big]R^{-1}=-\big[E^{-1}\mathbf{a}\frac{2E}{N}\mathbf{a}^TR-R^{-1}\frac{1}{N}(ER+R\mathbf{a}\,\mathbf{a}^TR)\big]R^{-1}\\ &=\frac{E}{N}(R^{-1}-E^{-1}\mathbf{a}\,\mathbf{a}^T)=\frac{E}{N}\begin{bmatrix}0 & \mathbf{0}^T\\ \mathbf{0} & \tilde{R}^{-1}\end{bmatrix}\end{aligned}$$

where we used Eq. (12.9.16) or (1.8.35), and $\tilde{R}$ is the lower-order portion of R. Such result was expected because $\Delta\mathbf{a}$ is of the form $\Delta\mathbf{a}=\begin{bmatrix}0\\ \Delta\boldsymbol{\alpha}\end{bmatrix}$. Thus,

$$E[\Delta\boldsymbol{\alpha}\,\Delta\boldsymbol{\alpha}^T]=\frac{E}{N}\tilde{R}^{-1} \tag{14.11.8}$$

This is a well-known result, and although we obtained it for sample covariance matrices of the type (1.6.21), where the snapshots $\mathbf{y}(n)$ were assumed to be independent, it can be proved in the case of autoregressive models where $\hat{R}$ is built out of the sample autocorrelation function [1171,1181–1191].

It can also be shown that asymptotically $\hat{E}$ and $\hat{\boldsymbol{\alpha}}$ are the *maximum likelihood* estimates of the LP parameters E and $\boldsymbol{\alpha}$, having all the good properties of such estimates, namely, asymptotic unbiasedness, consistency, efficiency, and gaussian distribution about the theoretical values with covariances given by Eqs. (14.11.6)–(14.11.8), which are none other than the Cramér-Rao bounds of these parameters. It is instructive to use the general formula (1.18.17) to derive these bounds, where the parameter vector is defined as $\boldsymbol{\lambda}=[E,\boldsymbol{\alpha}^T]^T$. We must determine the dependence of R on these parameters and then compute the derivatives $\partial R/\partial E$ and $\partial R/\partial\boldsymbol{\alpha}$. We write the UL factorization of R in the form of Eq. (1.8.33):

$$R=\begin{bmatrix}\rho_a & \mathbf{r}_a^T\\ \mathbf{r}_a & \tilde{R}\end{bmatrix}=U^{-1}D_aU^{-T}=\begin{bmatrix}1 & \boldsymbol{\alpha}^T\\ \mathbf{0} & \tilde{U}\end{bmatrix}^{-1}\begin{bmatrix}E & \mathbf{0}^T\\ \mathbf{0} & \tilde{D}\end{bmatrix}\begin{bmatrix}1 & \mathbf{0}^T\\ \boldsymbol{\alpha} & \tilde{U}^T\end{bmatrix}$$

The parametrization of R on the parameters E and $\boldsymbol{\alpha}$ is shown explicitly. It is evident that the entries ρ_a and $\mathbf{r}_a$ depend on E and $\boldsymbol{\alpha}$, whereas $\tilde{R}$ does not. We have

$$\mathbf{r}_a=-\tilde{R}\boldsymbol{\alpha}\,,\quad \rho_a=E-\boldsymbol{\alpha}^T\mathbf{r}_a=E+\boldsymbol{\alpha}^T\tilde{R}\boldsymbol{\alpha}$$

Working with differentials, we find $d\mathbf{r}_a=-\tilde{R}d\boldsymbol{\alpha}$ and $d\rho_a=dE+2\boldsymbol{\alpha}^T\tilde{R}d\boldsymbol{\alpha}$. Differentiating R entry-by-entry and using Eq. (1.8.35) for R^{-1}, we find

$$R^{-1}dR=E^{-1}\begin{bmatrix}dE+\boldsymbol{\alpha}^T\tilde{R}d\boldsymbol{\alpha} & -d\boldsymbol{\alpha}^T\tilde{R}\\ (dE+\boldsymbol{\alpha}^T\tilde{R}d\boldsymbol{\alpha})\boldsymbol{\alpha}-Ed\boldsymbol{\alpha} & -\boldsymbol{\alpha}d\boldsymbol{\alpha}^T\tilde{R}\end{bmatrix} \tag{14.11.9}$$

Writing a similar expression for a second differential $R^{-1}\delta R$, multiplying the two, and taking the trace, we find

$$\mathrm{tr}(R^{-1}dRR^{-1}\delta R)=E^{-2}dE\delta E+2E^{-1}d\boldsymbol{\alpha}^T\tilde{R}\delta\boldsymbol{\alpha} \tag{14.11.10}$$

This gives for the matrix elements of the Fisher information matrix

$$\begin{aligned}J_{EE}&=\frac{1}{2}N\,\mathrm{tr}\Big[R^{-1}\frac{\partial R}{\partial E}R^{-1}\frac{\partial R}{\partial E}\Big]=\frac{N}{2E^2}\\ J_{\alpha E}&=\frac{1}{2}N\,\mathrm{tr}\Big[R^{-1}\frac{\partial R}{\partial\boldsymbol{\alpha}}R^{-1}\frac{\partial R}{\partial E}\Big]=0\\ J_{\alpha\alpha}&=\frac{1}{2}N\,\mathrm{tr}\Big[R^{-1}\frac{\partial R}{\partial\boldsymbol{\alpha}}R^{-1}\frac{\partial R}{\partial\boldsymbol{\alpha}^T}\Big]=\frac{N}{E}\tilde{R}\end{aligned}$$

As we know, the inverse of the information matrix is the Cramér-Rao bound for unbiased estimates. This inverse agrees with Eqs. (14.11.6)–(14.11.8).

Following the discussion of [1186,1192], we may also derive the asymptotic covariances of the reflection coefficients. The forward and backward Levinson recursion establishes a one-to-one correspondence between the prediction coefficients $\boldsymbol{\alpha}$ and the vector of reflection coefficients $\boldsymbol{\gamma}$. Therefore, we have the differential correspondence $\Delta\boldsymbol{\gamma}=\Gamma\Delta\boldsymbol{\alpha}$, where Γ is the matrix of partial derivatives $\Gamma_{ij}=\partial\gamma_i/\partial\alpha_j$. It follows that the asymptotic covariance of $\boldsymbol{\gamma}$ will be

$$E[\Delta\boldsymbol{\gamma}\Delta\boldsymbol{\gamma}^T]=\Gamma E[\Delta\boldsymbol{\alpha}\Delta\boldsymbol{\alpha}^T]\Gamma^T=\frac{E}{N}\Gamma\tilde{R}^{-1}\Gamma^T \tag{14.11.11}$$

Example 14.11.1: For the first-order case, we have $\tilde{R}=[R(0)]$ and $E_1=(1-\gamma_1^2)R(0)$, where $\gamma_1=-a_{11}$. Thus, we obtain Eq. (1.16.4) as a special case

$$E\big[(\Delta a_{11})^2\big]=E\big[(\Delta\gamma_1)^2\big]=\frac{1-\gamma_1^2}{N}$$

For the second-order case, $\Delta\boldsymbol{\alpha}=[\Delta a_{12},\Delta a_{22}]^T$, and we have $E_2=R(0)\,(1-\gamma_1^2)\,(1-\gamma_2^2)$ and $\tilde{R}$ is the order-one autocorrelation matrix. Thus, we find

$$\begin{aligned}E[\Delta\boldsymbol{\alpha}\,\Delta\boldsymbol{\alpha}^T]&=\frac{E_2}{N}\tilde{R}^{-1}=\frac{E_2}{N}\begin{bmatrix}R(0) & R(1)\\ R(1) & R(0)\end{bmatrix}^{-1}\\ &=\frac{(1-\gamma_1^2)\,(1-\gamma_2^2)}{N(1-\gamma_1^2)}\begin{bmatrix}1 & -\gamma_1\\ -\gamma_1 & 1\end{bmatrix}=\frac{1-\gamma_2^2}{N}\begin{bmatrix}1 & -\gamma_1\\ -\gamma_1 & 1\end{bmatrix}\end{aligned}$$

From the Levinson recursion, we find for the second-order predictor $a_{22}=-\gamma_1(1-\gamma_2)$ and $a_{22}=-\gamma_2$. Differentiating, we have

$$d\boldsymbol{\alpha}=\begin{bmatrix}da_{12}\\ da_{22}\end{bmatrix}=\begin{bmatrix}-(1-\gamma_2) & \gamma_1\\ 0 & -1\end{bmatrix}\begin{bmatrix}d\gamma_1\\ d\gamma_2\end{bmatrix}$$

Inverting, we find

$$d\boldsymbol{\gamma}=\begin{bmatrix}d\gamma_1\\ d\gamma_2\end{bmatrix}=\frac{1}{1-\gamma_2}\begin{bmatrix}-1 & -\gamma_1\\ 0 & -(1-\gamma_2)\end{bmatrix}d\boldsymbol{\alpha}=\Gamma d\boldsymbol{\alpha}$$

Forming the product $\Gamma\tilde{R}^{-1}\Gamma^T$, we finally find

$$E[\Delta\boldsymbol{\gamma}\,\Delta\boldsymbol{\gamma}^T]=\frac{1}{N}\frac{1-\gamma_2^2}{(1-\gamma_2)^2}\begin{bmatrix}1-\gamma_1^2 & 0\\ 0 & (1-\gamma_2)^2\end{bmatrix}$$

which gives component-wise

$$E[(\Delta\gamma_1)^2] = \frac{1}{N}\frac{(1+\gamma_2)(1-\gamma_1^2)}{1-\gamma_2}\,,\quad E[\Delta\gamma_1\Delta\gamma_2]=0\,,\quad E[(\Delta\gamma_2)^2]=\frac{1-\gamma_2^2}{N}$$

Setting $\gamma_2 = 0$, the variance of γ_1 becomes equal to that of the first-order case and $E[(\Delta\gamma_2)^2] = 1/N$. More generally, for an autoregressive process of order M, all reflection coefficients of order greater than M vanish, but their asymptotic variances are equal to $1/N$, that is, $E[(\Delta\gamma_p)^2] = 1/N$, for $p > M$, [1186,1192]. □

Next, we consider the asymptotic properties of the eigenstructure of $\hat{R}$ [1197–1205]. In the complex-valued case $\hat{R}$ is given by Eq. (14.4.14), and Eq. (14.11.1) is replaced by

$$E[\Delta R_{ij}\,\Delta R_{kl}] = \frac{1}{N}R_{il}R_{kj} \tag{14.11.12}$$

where again $\Delta R = \hat{R} - R$. This can be shown in the same way as Eq. (1.6.23) using the following expression for the expectation value of the product of four complex gaussian random variables arising from the (independent) snapshots $\mathbf{y}(n)$ and $\mathbf{y}(m)$:

$$E[y_i(n)^* y_j(n) y_k(m)^* y_l(m)] = R_{ij}R_{kl} + \delta_{nm}R_{il}R_{kj}$$

Equation (14.11.12) may be written more conveniently in the form

$$E[(\mathbf{a}^\dagger\Delta R\mathbf{b})\,(\mathbf{c}^\dagger\Delta R\mathbf{d})] = \frac{1}{N}(\mathbf{a}^\dagger R\mathbf{d})\,(\mathbf{c}^\dagger R\mathbf{b}) \tag{14.11.13}$$

for any four $(M+1)$-dimensional vectors $\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}$. In particular, we may apply it to four eigenvectors of R. Let $\mathbf{e}_i$ denote the orthonormal eigenvectors of R, $R\mathbf{e}_i = \lambda_i\mathbf{e}_i$, with eigenvalues arranged in increasing order. Then,

$$E[(\mathbf{e}_i^\dagger\Delta R\mathbf{e}_j)\,(\mathbf{e}_k^\dagger\Delta R\mathbf{e}_l)] = \frac{1}{N}(\mathbf{e}_i^\dagger R\mathbf{e}_l)\,(\mathbf{e}_k^\dagger R\mathbf{e}_j) = \frac{1}{N}\lambda_i\lambda_j\delta_{il}\delta_{kj}$$

where we used $(\mathbf{e}_i^\dagger R)\mathbf{e}_l = \lambda_i\mathbf{e}_i^\dagger\mathbf{e}_l = \lambda_i\delta_{il}$. Arranging the eigenvectors into the eigenvector matrix $E = [\mathbf{e}_0,\mathbf{e}_1,\dots,\mathbf{e}_M]$, we recognize that the quantities $\mathbf{e}_i\Delta R\mathbf{e}_j$, are the matrix elements of ΔR in the E basis; that is, the elements of the matrix $\Delta V = E^\dagger\Delta RE$. Thus, we obtain the *diagonalized* version of Eq. (14.11.12)

$$E[\Delta V_{ij}\,\Delta V_{kl}] = \frac{1}{N}\lambda_i\lambda_j\delta_{il}\delta_{kj} \tag{14.11.14}$$

The asymptotic properties of the eigenstructure of $\hat{R}$ are obtained by using Eq. (14.11.14) and standard first-order perturbation theory. The eigenproblems for R and $\hat{R}$ are,

$$RE = E\Lambda \quad\text{and}\quad \hat{R}\hat{E} = \hat{E}\hat{\Lambda} \tag{14.11.15}$$

where $\hat{E}, E$ are the eigenvector matrices and $\hat{\Lambda},\Lambda$ the diagonal matrices of the eigenvalues. Because the eigenvectors E form a complete set, it follows that the eigenvectors $\hat{E}$ can be expanded as linear combinations of the former; that is, $\hat{E} = EF$. The orthonormality and completeness of $\hat{E}$ and E require that F be a unitary matrix, satisfying $F^\dagger F = FF^\dagger = I$. This is easily shown; for example, $I = \hat{E}^\dagger\hat{E} = F^\dagger E^\dagger EF = F^\dagger IF = F^\dagger F$.

In carrying out the first-order perturbation analysis, we shall assume initially that all the eigenvalues of R are distinct. This corresponds to the Pisarenko case, where the noise subspace is one-dimensional and thus, $L = M$.

The assumption of distinct eigenvalues means that, under a perturbation, $\hat{R} = R + \Delta R$, each eigenvector changes by a small correction of the form $\hat{E} = E + \Delta E$. By the completeness of the basis E we may write $\Delta E = E\,\Delta C$ so that $\hat{E} = E(I + \Delta C) = EF$. The unitarity of the matrix $F = I + \Delta C$ requires that ΔC be anti-hermitian; that is, $\Delta C + \Delta C^\dagger = 0$. This follows from the first-order approximation $F^\dagger F = I + \Delta C + \Delta C^\dagger$. The perturbation changes the eigenvalues by $\hat{\lambda}_i = \lambda_i + \Delta\lambda_i$, or, $\hat{\Lambda} = \Lambda + \Delta\Lambda$. To determine the first-order corrections we use Eq. (14.11.15)

$$(R+\Delta R)(E+\Delta E) = (E+\Delta E)(\Lambda+\Delta\Lambda) \quad\Rightarrow\quad (\Delta R)E + R(\Delta E) = (\Delta E)\Lambda + E(\Delta\Lambda)$$

where we kept only the first-order terms. Multiplying both sides by $E^\dagger$ and using $E^\dagger RE = \Lambda$ and the definition $\Delta V = E^\dagger(\Delta R)E$, we obtain

$$\Delta V + \Lambda(\Delta C) = (\Delta C)\Lambda + \Delta\Lambda \quad\Rightarrow\quad \Delta\Lambda + (\Delta C)\Lambda - \Lambda(\Delta C) = \Delta V$$

or, component-wise

$$\Delta\lambda_i\delta_{ij} + (\lambda_j - \lambda_i)\Delta C_{ij} = \Delta V_{ij}$$

Setting $i = j$ and then $i \neq j$, we find

$$\Delta\lambda_i = \Delta V_{ii}\,,\quad \Delta C_{ij} = -\frac{\Delta V_{ij}}{\lambda_i - \lambda_j}\,,\quad \text{for}\quad i\neq j \tag{14.11.16}$$

Using Eq. (14.11.14), we obtain the asymptotic variances of the eigenvalues

$$E[(\Delta\lambda_i)^2] = E[\Delta V_{ii}\,\Delta V_{ii}] = \frac{\lambda_i^2}{N} \tag{14.11.17}$$

For the eigenvectors, we write

$$\Delta\mathbf{e}_i = \hat{\mathbf{e}}_i - \mathbf{e}_i = \sum_{j\neq i}\mathbf{e}_j\Delta C_{ji}$$

and their covariances are

$$E[\Delta\mathbf{e}_i\Delta\mathbf{e}_i^\dagger] = \sum_{j\neq i}\sum_{k\neq i}\mathbf{e}_j\mathbf{e}_k^\dagger E[\Delta C_{ji}\Delta C_{ki}^*]$$

Using the anti-hermiticity of ΔC and Eq. (14.11.14), we find

$$E[\Delta C_{ji}\Delta C_{ki}^*] = -\frac{E[\Delta V_{ji}\Delta V_{ik}]}{(\lambda_j-\lambda_i)(\lambda_i-\lambda_k)} = \frac{1}{N}\frac{\lambda_i\lambda_j}{(\lambda_i-\lambda_j)^2}\delta_{jk}$$

which gives

$$E[\Delta\mathbf{e}_i\Delta\mathbf{e}_i^\dagger] = \frac{1}{N}\sum_{j\neq i}\frac{\lambda_i\lambda_j}{(\lambda_i-\lambda_j)^2}\mathbf{e}_j\mathbf{e}_j^\dagger \tag{14.11.18}$$

Separating out the minimum eigenvalue λ_0 and eigenvector $\mathbf{e}_0$, and denoting the remaining signal subspace eigenvectors and eigenvalues by $E_S = [\mathbf{e}_1, \dots, \mathbf{e}_M]$ and $\Lambda_S = \text{diag}\{\lambda_1, \dots, \lambda_M\}$, we may write Eq. (14.11.18) compactly

$$E[\Delta\mathbf{e}_0 \Delta\mathbf{e}_0^\dagger] = \frac{\lambda_0}{N} E_S \Lambda_S (\Lambda_S - \lambda_0 I_M)^{-2} E_S^\dagger \tag{14.11.19}$$

where I_M is the M-dimensional unit matrix. The zeros of the polynomial $\mathbf{e}_0$ contain the desired frequency information. The asymptotic variances for the zeros can be obtained by writing

$$\Delta z_i = \left(\frac{\partial z_i}{\partial \mathbf{e}_0}\right)^T \Delta\mathbf{e}_0$$

which gives

$$E\big[|\Delta z_i|^2\big] = \left(\frac{\partial z_i}{\partial \mathbf{e}_0}\right)^T E[\Delta\mathbf{e}_0 \Delta\mathbf{e}_0^\dagger] \left(\frac{\partial z_i}{\partial \mathbf{e}_0}\right)^* \tag{14.11.20}$$

Example 14.11.2: In the $L = M = 1$ Example 14.3.1, we have for the eigenvalues and orthonormal eigenvectors of R

$$\lambda_0 = \sigma_v^2\,, \quad \lambda_1 = \sigma_v^2 + 2P_1\,, \quad \mathbf{e}_0 = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ -e^{jk_1} \end{bmatrix}, \quad \mathbf{e}_1 = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ e^{jk_1} \end{bmatrix}$$

It follows from Eq. (14.11.19) that

$$E[\Delta\mathbf{e}_0 \Delta\mathbf{e}_0^\dagger] = \frac{1}{N}\mathbf{e}_1\mathbf{e}_1^\dagger \frac{\lambda_1\lambda_0}{(\lambda_1 - \lambda_0)^2}$$

Using the general formula for the sensitivities of zeros with respect to the coefficients of a polynomial [12].

$$\frac{\partial z_i}{\partial a_m} = -\frac{1}{a_0}\frac{z_i^{M-m}}{\prod_{j\neq i}(z_i - z_j)}$$

we find for the zero $z_1 = e^{jk_1}$ of $\mathbf{e}_0$

$$\frac{\partial z_1}{\partial \mathbf{e}_0} = -\sqrt{2}\begin{bmatrix} z_1 \\ 1 \end{bmatrix}$$

Using this into Eq. (14.11.20), we find

$$E\big[|\Delta z_i|^2\big] = \frac{1}{N}\frac{4\lambda_1\lambda_0}{(\lambda_1 - \lambda_0)^2} = \frac{1}{N}\frac{1 + 2SNR}{SNR^2}\,, \quad SNR = \frac{P_1}{\sigma_v^2}$$

This implies that the quality of the estimated zero improves either by increasing the number of snapshots N or the signal to noise ratio. For low SNR, the denominator $(\lambda_1 - \lambda_0)^2$ becomes small and the variance of z_1 increases, resulting in degradation of performance. For a given level of quality there is a tradeoff between the number of snapshots and SNR. In general, the signal subspace eigenvalues Λ_S will be separated from $\lambda_0 = \sigma_v^2$ by a term that depends on the signal powers, say, $\Lambda_S = \lambda_0 I_M + P_S$. Then,

$$\lambda_0 \Lambda_S (\Lambda_S - \lambda_0 I_M)^{-2} = (I_M + P_S/\sigma_v^2)(P_S/\sigma_v^2)^{-2}$$

and Eq. (14.11.19) implies that the estimate of $\mathbf{e}_0$ becomes better for higher SNRs. □

When the noise subspace has dimension $K = M + 1 - L$ and the minimum eigenvalue λ_0 has K-fold degeneracy, the first-order perturbation analysis becomes somewhat more complicated. The eigenproblem for R is divided into its noise and signal subspace parts

$$RE_N = \lambda_0 E_N\,, \quad RE_S = E_S\Lambda_S$$

where E_N consists of the K degenerate eigenvectors belonging to the minimum eigenvalue $\lambda_0 = \sigma_v^2$ and E_S consists of the remaining L signal subspace eigenvectors. Under a perturbation $\hat{R} = R + \Delta R$, the degeneracy of E_N is lifted and the noise subspace eigenvalues become unequal $\hat{\lambda}_i = \lambda_0 + \Delta\lambda_i$, $i = 0, 1, \dots, K-1$, or, $\hat{\Lambda}_N = \lambda_0 I_K + \Delta\Lambda_N$. Similarly, the signal subspace eigenvalues change to $\hat{\Lambda}_S = \Lambda_S + \Delta\Lambda_S$.

The signal subspace eigenvectors, belonging to distinct eigenvalues, change in the usual way; namely, each eigenvector changes by receiving small contributions from all other eigenvectors. The noise subspace eigenvectors, however, being degenerate, are mixed up by the perturbation into linear combinations of themselves, and in addition, they receive small corrections from the signal subspace eigenvectors. Thus, the eigenproblem for the perturbed matrix $\hat{R}$ is

$$\hat{R}\hat{E}_N = \hat{E}_N\hat{\Lambda}_N\,, \quad \hat{R}\hat{E}_S = \hat{E}_S\hat{\Lambda}_S \tag{14.11.21}$$

where the corrections of the eigenvectors are of the form

$$\hat{E}_N = E_N C + E_S\Delta C\,, \quad \hat{E}_S = E_S + E_S\Delta B + E_N\Delta D \tag{14.11.22}$$

In absence of the perturbation ΔR, the choice of the degenerate basis E_N is arbitrary and can be replaced by any linear combination $E_N C$. The presence of the perturbation fixes this particular linear combination by the requirement that the change in the eigenvectors be small. Combining the two equations into the full eigenvector matrices, we have

$$\hat{E} = [\hat{E}_N, \hat{E}_S] = [E_N, E_S]\begin{bmatrix} C & \Delta D \\ \Delta C & I_L + \Delta B \end{bmatrix} = EF$$

The orthonormality and completeness requirements for $\hat{E}$ imply that $F^\dagger F = FF^\dagger = I$. To first order, these conditions are equivalent to

$$C^\dagger C = I_K\,, \quad \Delta C + \Delta D^\dagger C = 0\,, \quad \Delta B + \Delta B^\dagger = 0 \tag{14.11.23}$$

Thus, C must be unitary. Inserting Eq. (14.11.22) into the first term of (14.11.21) and using (14.11.23), we find

$$(R + \Delta R)(E_N C - E_S\Delta D^\dagger C) = (E_N C - E_S\Delta D^\dagger C)(\lambda_0 I_K + \Delta\Lambda_N)$$

and equating first-order terms,

$$\Delta R E_N C - E_S\Lambda_S\Delta D^\dagger C = E_N C\Delta\Lambda_N - E_S\Delta D^\dagger C\lambda_0$$

Multiplying both sides first by $E_N^\dagger$ and then by $E_S^\dagger$ and using the orthonormality properties (14.4.3), we obtain

$$\Delta V_{NN} C = C\Delta\Lambda_N \tag{14.11.24}$$

where $\Delta V_{NN} = E_N^\dagger \Delta R E_N$, and

$$\Delta V_{SN} C - \Lambda_S \Delta D^\dagger C = -\Delta D^\dagger C \lambda_0$$

where $\Delta V_{SN} = E_S^\dagger \Delta R E_N$, and solving for $\Delta D^\dagger$

$$\Delta D^\dagger = (\Lambda_S - \lambda_0 I_L)^{-1} \Delta V_{SN} \tag{14.11.25}$$

Similarly, from the second term of Eq. (14.11.21), we find for ΔB

$$\Delta\Lambda_S + \Delta B \Lambda_s - \Lambda_s \Delta B = \Delta V_{SS}\,, \quad \Delta V_{SS} = E_S^\dagger \Delta R E_S \tag{14.11.26}$$

which can be solved as in Eq. (14.11.16). To summarize, the corrections to the noise subspace eigenvalues $\Delta\Lambda_N$ and the unitary matrix C are obtained from the solution of the $K\times K$ eigenproblem (14.11.24), ΔD constructed by (14.11.25), then ΔC is constructed by (14.11.23), and ΔB by (14.11.26).

Because the corrections to the signal subspace eigenvectors are obtained from the non-degenerate part of the perturbation analysis, it follows that (14.11.18) is still valid for the signal eigenvectors. More specifically, because we index the noise subspace eigenvectors for $0 \le i \le K-1$ and the signal subspace eigenvectors for $K \le i \le M$, we may split the sum over the noise and signal subspace parts

$$E[\Delta\mathbf{e}_i \Delta\mathbf{e}_i^\dagger] = \frac{1}{N}\frac{\lambda_0\lambda_i}{(\lambda_0-\lambda_i)^2}\sum_{j=0}^{K-1}\mathbf{e}_j\mathbf{e}_j^\dagger + \frac{1}{N}\sum_{\substack{j\neq i\\ j=K}}^{M}\frac{\lambda_i\lambda_j}{(\lambda_i-\lambda_j)^2}\mathbf{e}_j\mathbf{e}_j^\dagger$$

where we used the fact that all noise subspace eigenvalues are equal to λ_0. The first term is recognized as the projector onto the noise subspace. Thus, for $K \le i \le M$,

$$E[\Delta\mathbf{e}_i \Delta\mathbf{e}_i^\dagger] = \frac{1}{N}\frac{\lambda_0\lambda_i}{(\lambda_0-\lambda_i)^2}E_N E_N^\dagger + \frac{1}{N}\sum_{\substack{j\neq i\\ j=K}}^{M}\frac{\lambda_i\lambda_j}{(\lambda_i-\lambda_j)^2}\mathbf{e}_j\mathbf{e}_j^\dagger \tag{14.11.27}$$

Because most eigenvector methods can also be formulated in terms of the signal subspace eigenvectors, it is enough to consider only the asymptotic covariances of these eigenvectors. For example, in the reduced-order method of Sec. 14.7, the reduced-order polynomials $\mathbf{a}_i$ may alternatively be computed by requiring that the corresponding shifted vectors $\mathbf{b}_i$ be orthogonal to the signal subspace [1139]; namely, $E_S^\dagger \mathbf{b}_i = 0$, $i = 0, 1, \dots, K-1$, and similarly, for the empirical quantities $\hat{E}_S^\dagger \hat{\mathbf{b}}_i = 0$. If we denote by G_i the part of E_S consisting of $L+1$ rows starting with the ith row, then, these conditions become $G_i^\dagger \mathbf{a}_i = 0$. Because the first coefficient of $\mathbf{a}_i$ is unity, these give rise to L linear equations for the L last coefficients $\mathbf{a}_i$. It follows that $\mathbf{a}_i$ can be constructed as a function of the signal eigenvectors, and thus, one can obtain the corresponding covariance of $\mathbf{a}_i$ using Eq. (14.11.27). An example will illustrate this remark.

Example 14.11.3: Consider the case of one plane wave ($L = 1$) and arbitrary M. The covariance matrix $R = \sigma_v^2 I + P_1 \mathbf{s}_{k_1}\mathbf{s}_{k_1}^\dagger$ has a one-dimensional signal subspace so that $E_S = [\mathbf{e}_M]$, Its eigenvalue is $\lambda_M = \sigma_v^2 + (M+1)P_1$. The matrix G_i is formed by row i to row $i+L = i+1$, that is,

$$G_i = \begin{bmatrix} e_{M,i} \\ e_{M,i+1} \end{bmatrix} = \frac{1}{\sqrt{M+1}}\begin{bmatrix} e^{jk_1 i} \\ e^{jk_1(i+1)} \end{bmatrix}$$

The equation $G_i^\dagger \mathbf{a}_i = 0$ becomes for the first-order filters $\mathbf{a}_i$,

$$G_i^\dagger \mathbf{a}_i = \frac{1}{\sqrt{M+1}}[e^{-jk_1 i},\, e^{-jk_1(i+1)}]\begin{bmatrix} 1 \\ a_{i1} \end{bmatrix} = 0 \quad \Rightarrow \quad a_{i1} = -e^{jk_1}$$

and hence, all the reduced-order polynomials are equal to the theoretical one, $A_i(z) = 1 - e^{jk_1}z^{-1}$. Now, if the empirical $\hat{\mathbf{e}}_M$ is used , then a similar calculation gives $a_{i1} = -e_{M,i}^*/e_{M,i+1}^*$, and therefore, the estimated zero will be $\hat{z}_1 = e_{M,i}^*/e_{M,i+1}^*$. Differentiating, we obtain $d\hat{z}_1 = de_{M,i}^*/e_{M,i+1}^* - e_{M,i}^* de_{M,i+1}^*/e_{M,i+1}^{*2}$; therefore, its covariance will be

$$\begin{aligned} E[|\Delta z_1|^2] = {} & \frac{1}{|e_{M,i+1}|^2}E[|\Delta e_{M,i}|^2] + \frac{|e_{M,i}|^2}{|e_{M,i+1}|^4}E[|\Delta e_{M,i+1}|^2] \\ & - 2\,\mathrm{Re}\left[\frac{e_{M,i}^*}{e_{M,i+1}e_{M,i+1}^{*2}}\,E[\Delta e_{M,i}\Delta e_{M,i+1}^*]\right] \end{aligned}$$

This simplifies to

$$E[|\Delta z_1|^2] = (M+1)\Big[E[|\Delta e_{M,i}|^2] + E[|\Delta e_{M,i+1}|^2] - 2\,\mathrm{Re}\big(e^{jk_1}E[\Delta e_{M,i}\Delta e_{M,i+1}^*]\big)\Big]$$

Because the signal subspace is one-dimensional, the second term in Eq. (14.11.27) is absent. The noise-subspace projector can be expressed in terms of the signal-subspace projector $E_N E_N^\dagger = I - E_S E_S^\dagger$. Thus, Eq. (14.11.27) gives

$$E[\Delta\mathbf{e}_0 \Delta\mathbf{e}_0^\dagger] = \frac{1}{N}\frac{\lambda_M\lambda_0}{(\lambda_M-\lambda_0)^2}\left(I - \frac{1}{M+1}\mathbf{s}_{k_1}\mathbf{s}_{k_1}^\dagger\right)$$

Extracting the ith and $(i+1)$st components, we get for the variance of the estimated zero

$$E[|\Delta z_1|^2] = \frac{1}{N}\frac{2(M+1)\lambda_M\lambda_0}{(\lambda_M-\lambda_0)^2} = \frac{1}{N}\frac{2[1+(M+1)SNR]}{(M+1)SNR^2}$$

where $SNR = P_1/\sigma_v^2$. Setting $M = 1$, we recover the result of Example 14.11.2. □

14.12 Computer Project – LCMV Beamforming and GSC

This computer project, divided into separate parts, deals with the theory of linearly-constrained minimum-variance (LCMV) beamforming and its equivalence to the generalized sidelobe canceler [1209–1221]. The problem also has application in linearly-constrained time-series Wiener filtering, and other applications, such as optimum minimum-variance Markowitz portfolios in finance (those are discussed in the following project). The following topics are included,

- Linearly-constrained Wiener filtering problem.
- Linearly-constrained minimum-variance beamforming.

- Retrodirective beams towards multiple interferers.
- Quiescent pattern control with linear constraints.
- Equivalence of LCMV and the generalized sidelobe canceler.

1. *Linearly-constrained Wiener filtering problem.* Let R be a given $M\times M$ positive-definite Hermitian matrix and $\mathbf{r}$ be a given $M\times 1$ complex-valued vector. It is desired to find the weight vector $\mathbf{a}$ that minimizes:

$$\mathcal{E} = \mathbf{a}^\dagger R\mathbf{a} - \mathbf{r}^\dagger \mathbf{a} - \mathbf{a}^\dagger \mathbf{r} = \min \tag{14.12.1}$$

subject to the K linear constraints:

$$C^\dagger \mathbf{a} = \mathbf{g} \tag{14.12.2}$$

where $K < M$, and C is a $M\times K$ matrix with linearly independent columns, and $\mathbf{g}$ is a given $K\times 1$ vector of "gains". Component-wise, Eq. (14.12.2) reads $\mathbf{c}_i^\dagger \mathbf{a} = g_i$, $i = 1,2,\dots,K$, where $\mathbf{c}_i$ is the ith column of C, and g_i the ith component of $\mathbf{g}$.

(a) Show that the unconstrained minimization of Eq. (14.12.1) gives the solution:

$$\mathbf{a}_u = R^{-1}\mathbf{r} \tag{14.12.3}$$

(b) Introduce a K-dimensional complex-valued vector of Lagrange multipliers $\boldsymbol{\lambda}$ and minimize the modified performance index:

$$J = \mathbf{a}^\dagger R\mathbf{a} - \mathbf{r}^\dagger \mathbf{a} - \mathbf{a}^\dagger \mathbf{r} + \boldsymbol{\lambda}^\dagger(\mathbf{g} - C^\dagger\mathbf{a}) + (\mathbf{g}^\dagger - \mathbf{a}^\dagger C)\boldsymbol{\lambda} = \min$$

Show that the solution the solution of this problem is the solution of the constrained problem of Eqs. (14.12.1) and (14.12.2) can be expressed in terms of $\mathbf{a}_u$ as follows:

$$\mathbf{a} = \mathbf{a}_u + R^{-1}C(C^\dagger R^{-1}C)^{-1}(\mathbf{g} - C^\dagger \mathbf{a}_u) \tag{14.12.4}$$

Many of the subsequent questions are special cases of this result.

2. *LCMV Beamforming.* Consider an array of M antennas equally-spaced at distance d (in units of the wavelength λ) along the x-axis. The array response is to be designed to achieve K prescribed gain values g_i at the directions θ_i corresponding to the steering vectors:

$$\mathbf{s}_i = \mathbf{s}_{k_i}\,, \quad k_i = 2\pi d\,\sin\theta_i\,, \quad i = 1,2,\dots,K \tag{14.12.5}$$

where

$$\mathbf{s}_k = \begin{bmatrix} 1 \\ e^{jk} \\ e^{2jk} \\ \vdots \\ e^{(M-1)jk} \end{bmatrix}$$

Thus, the constraint matrix C and gain vector are:

$$C = [\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_K]\,, \quad \mathbf{g} = \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_K \end{bmatrix}$$

We assume that L plane waves are incident on the array from L directions which may or may not coincide with the above constraint angles. Let S be the $M\times L$ steering matrix of the incident plane waves and let P be their $L\times L$ power matrix, assumed to have full rank. The $M\times M$ autocorrelation matrix of the array output in the presence of uncorrelated noise is:

$$R = \sigma_v^2 I + SPS^\dagger \tag{14.12.6}$$

(a) Using the results of the previous question, show that the optimum array weights that minimize the output power subject to the K constraints:

$$\boxed{\mathcal{E} = \mathbf{a}^\dagger R\mathbf{a} = \min\,, \quad \text{subject to} \quad C^\dagger\mathbf{a} = \mathbf{g}} \tag{14.12.7}$$

are given by:

$$\boxed{\mathbf{a} = R^{-1}C(C^\dagger R^{-1}C)^{-1}\mathbf{g}} \quad \text{(LCMV)} \tag{14.12.8}$$

This is known as the *linearly-constrained minimum variance* (LCMV) beamformer. See Frost [1209] for its LMS adaptive implementation. The corresponding array response towards an angle θ is defined as follows in absolute units and in dB

$$A(\theta) = |\mathbf{s}_k^\dagger \mathbf{a}|\,, \quad A_{\text{dB}}(\theta) = 20\log_{10} A(\theta)\,, \quad k = 2\pi d\sin\theta \tag{14.12.9}$$

(b) With R given by Eq. (14.12.6), show that if $C = S$, then, Eq. (14.12.8) reduces to

$$\mathbf{a} = S(S^\dagger S)^{-1}\mathbf{g} \tag{14.12.10}$$

which is recognized as the minimum-norm solution of the equation $S^\dagger\mathbf{a} = \mathbf{g}$. Being the minimum-norm solution implies that it minimizes $\mathbf{a}^\dagger\mathbf{a}$. How is this reconciled with the fact that $\mathbf{a}$ minimizes Eq. (14.12.7)?

(c) *Retrodirective Beamforming towards Multiple Interferers.* An example of retrodirective beamforming was given in Eq. (14.3.27). In the present notation, this corresponds to the case $C = [\mathbf{s}_1]$, $\mathbf{g} = [1]$, and $S = [\mathbf{s}_1, \mathbf{s}_2]$ with $\mathbf{s}_1$ being the desired look-direction and $\mathbf{s}_2$ representing a jammer.

Suppose that the incident signals are divided into two groups, the desired signals S_1 and the interferers S_2, so that $S = [S_1, S_2]$, where S_1 and S_2 have L_1 and L_2 columns, respectively, such that $L_1 + L_2 = L$, and let us assume that the corresponding power matrices are P_1 and P_2 and that the S_1 and S_2 are spatially uncorrelated so that the full power matrix is block-diagonal so that R is expressed in the form:

$$R = \sigma_v^2 I + S_2P_2S_2^\dagger + S_1P_1S_1^\dagger \equiv R_n + S_1P_1S_1^\dagger$$

where $R_n = \sigma_v^2 I + S_2 P_2 S_2^\dagger$ is the noise-plus-interference covariance matrix. Using the matrix inversion lemma, show the identity:

$$R^{-1}S_1(S_1^\dagger R^{-1}S_1)^{-1} = R_n^{-1}S_1(S_1^\dagger R_n^{-1}S_1)^{-1}$$

Thus, if we choose $C = S_1$ and $\mathbf{g}$ being an arbitrary vector of responses towards the desired look directions, the optimum weights will be given by:

$$\mathbf{a} = R^{-1}C(C^\dagger R^{-1}C)^{-1}\mathbf{g} = R_n^{-1}S_1(S_1^\dagger R_n^{-1}S_1)^{-1}\mathbf{g}$$

Using the matrix inversion lemma on R_n, show the following results:

$$R_n^{-1}S_1 = \frac{1}{\sigma_v^2}\big[S_1 - S_2(\sigma_v^2 P_2^{-1} + S_2^\dagger S_2)^{-1}S_2^\dagger S_1\big]$$

$$S_2^\dagger R_n^{-1}S_1 = P_2^{-1}(\sigma_v^2 P_2^{-1} + S_2^\dagger S_2)^{-1}S_2^\dagger S_1$$

Thus, the array gain steered towards the interferers, $S_2^\dagger \mathbf{a}$, becomes smaller with increasing interferer power P_2.

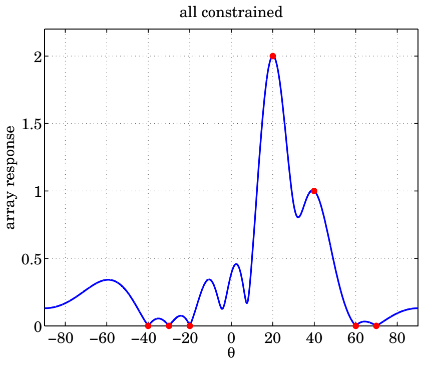
(d) Consider an array of 10 antennas equally-spaced at half-wavelength spacings ($d = 0.5$). The array is designed to receive two desired signals from angles $\theta_1 = 20^{\rm o}$ and $\theta_2 = 40^{\rm o}$ and reject five interferers coming in from the directions:

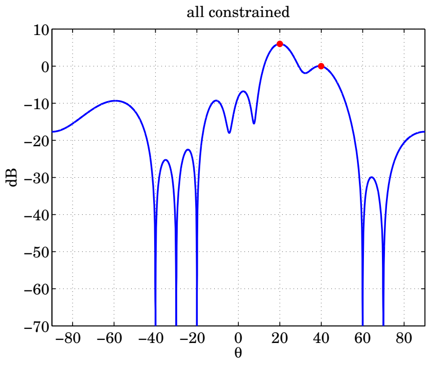
$$\{\theta_3, \theta_4, \theta_5, \theta_6, \theta_7\} = \{-40^{\rm o},\ -30^{\rm o},\ -20^{\rm o},\ 60^{\rm o},\ 70^{\rm o}\}$$

The response towards θ_1 is to be double that towards θ_2, while the responses towards the interferers must be zero. Thus, the constraint matrix and gain vector must be chosen as:

$$C = [\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3, \mathbf{s}_4, \mathbf{s}_5, \mathbf{s}_6, \mathbf{s}_7]\,, \quad \mathbf{g} = [2, 1, 0, 0, 0, 0, 0]^T$$

Assume that all plane waves have 10-dB power levels (and P is diagonal). Design the optimum weight vector and calculate and plot the array response of Eq. (14.12.9) in absolute units over the angle range $-90^{\rm o} \le \theta \le 90^{\rm o}$. Indicate on the graph the values of the gain vector $\mathbf{g}$. Plot it also in dB with a vertical scales ranging from $[-70, 10]$ dB.



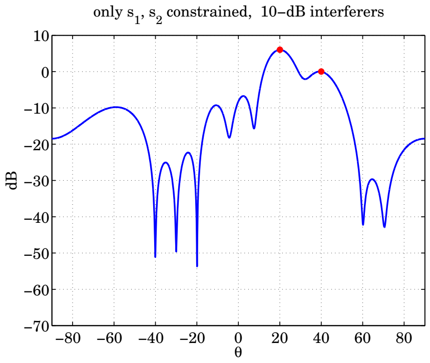


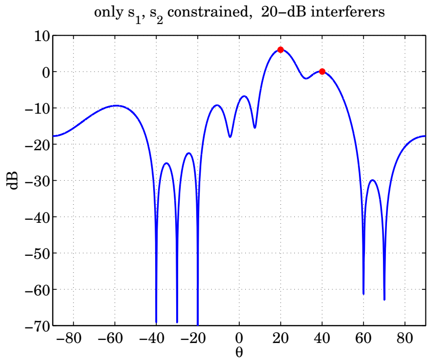
(e) To see the retrodirective action of the optimum beamformer, redesign the optimum weights based only on the desired signal constraints:

$$C = [\mathbf{s}_1, \mathbf{s}_2]\,, \quad \mathbf{g} = [2, 1]^T$$

and plot the array response in dB using the same vertical dB scales as before. Note the nulls at the interferer directions.

Repeat when the interferer powers are increased to 20 dB. In this case, you should expect the interferer nulls to be deeper.

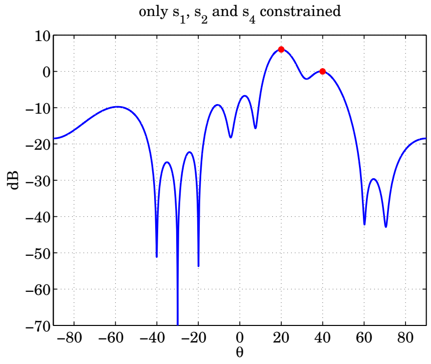




(f) Repeat when the interferer at θ_4 is to be nulled exactly and the other interferers nulled approximately by the retrodirective action of the beamformer, that is, choose

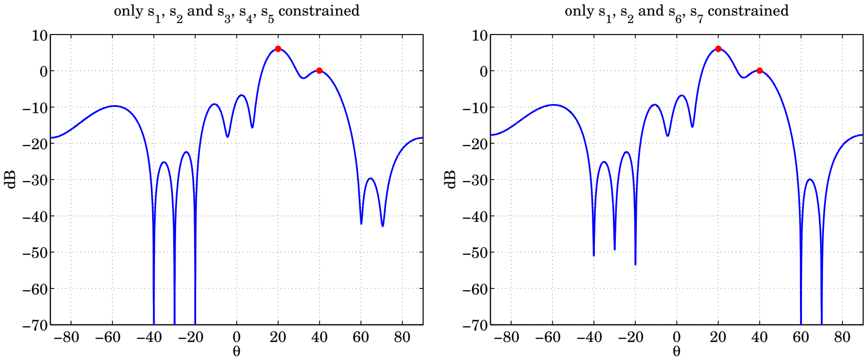
$$C = [\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_4]\,, \quad \mathbf{g} = [2, 1, 0]^T$$

Plot the array gain in dB using the same vertical scales as before. Assume 10-dB power levels for all signals.



(g) Repeat the design and the gain plot for the following two cases. When interferes 3,4,5 are to be nulled exactly, and when 6,7 are to be nulled, that is, for the two cases defined by:

$$C = [\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3, \mathbf{s}_4, \mathbf{s}_5]\,, \qquad \mathbf{g} = [2, 1, 0, 0, 0]^T$$
$$C = [\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_6, \mathbf{s}_7]\,, \qquad \mathbf{g} = [2, 1, 0, 0]^T$$



Note that the construction of the steering matrix S and constraint matrix C can be done with the help of the MATLAB function `steermat`, and the array gain can be calculated with the help of the `dtft` function, both of which may be found in the collection `osp-toolbox`. For example, in the very last case, we may define,

```
Pdb = [10 10 10 10 10 10 10]; P = diag(10.^(Pdb/10));
S = steermat(M-1, exp(j*[k1, k2, k3, k4, k5, k6, k7]));
C = steermat(M-1, exp(j*[k1, k2, k6, k7]));
R = eye(M) + S*P*S';    % assume unit-variance noise
```

where $k_i = 2\pi d \sin\theta_i$. For the gain computation, you may use:

```
th = linspace(-90,90,1001); thr = pi*th/180; d=0.5;
k = 2*pi*d*sin(thr);
A = abs(dtft(a,k));          % array response, |A(theta)|
plot(th,A);
```

3. *Quiescent Pattern Control.* Next, we consider the proper design of the quiescent response of an array that achieves a desired shape and respects the beam constraints. The method is discussed by Griffiths and Buckley [1213].

Consider an antenna array defined by Eqs. (14.12.6)–(14.12.9). The quiescent weights $\mathbf{a}_q$ correspond to the case when the incident signals are absent and only noise is present, that is, when $R = \sigma_v^2 I$. In this case, show that the optimum weights (14.12.8) are given by

$$\mathbf{a}_q = C(C^\dagger C)^{-1}\mathbf{g} \qquad (14.12.11)$$

They correspond to the minimum-norm solution of the constraints, $C^\dagger\mathbf{a} = \mathbf{g}$.

(a) Suppose that the array is to be steered towards a desired look-direction θ_1 corresponding to a steering vector $\mathbf{s}_1$. If we choose $C = \mathbf{s}_1$ for the constraint matrix and $\mathbf{g} = [1]$, that is, $\mathbf{s}_1^\dagger\mathbf{a} = 1$ for the constraint, then show that the quiescent weights are:

$$\mathbf{a}_q = \frac{1}{M}\mathbf{s}_1 \qquad (14.12.12)$$

and that the array response becomes

$$A(\theta) = |\mathbf{s}_k^\dagger\mathbf{a}_q| = \frac{1}{M}\,|W(k - k_1)| \qquad (14.12.13)$$

where $k = 2\pi d\sin\theta$, $k_1 = 2\pi d\sin\theta_1$, and $W(k)$ is the response of the rectangular window of length M.

(b) If $C = \mathbf{s}_1$ as above, but only the desired signal is present, that is, $R = \sigma_v^2 I + P_1\mathbf{s}_1\mathbf{s}_1^\dagger$, then show that the optimum weights are again given by Eq. (14.12.12).

(c) Consider an array of 21 antennas spaced at half-wavelength intervals. The desired signal comes from the direction $\theta_1 = 30^{\rm o}$ and has 0-dB signal to noise ratio.

Plot the quiescent response (14.12.13) versus angle using dB scales with a vertical range of $[-80, 5]$ dB. Notice the usual 13-dB level of the highest sidelobes. This corresponds to a steered uniform array.

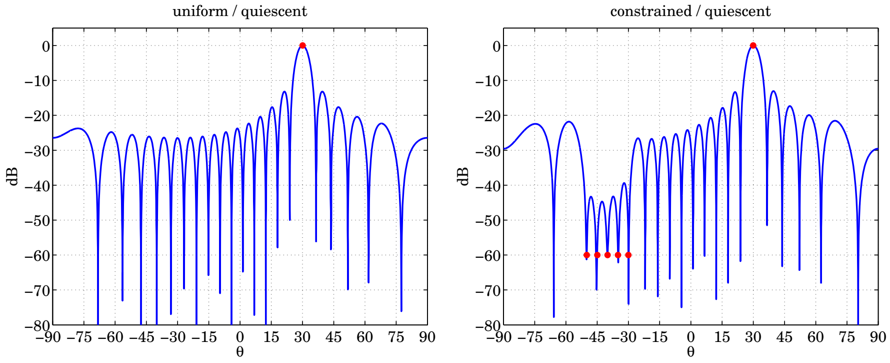
(d) It is anticipated that one or more interferers may come on in the angular range of $-50^{\rm o} \le \theta \le -30^{\rm o}$. To mitigate their effect, we try to force the array gain to be -60 dB at the following angles:

$$\{\theta_2, \theta_3, \theta_4, \theta_5, \theta_6\} = \{-50^{\rm o}, -45^{\rm o}, -40^{\rm o}, -35^{\rm o}, -30^{\rm o}\}$$

Thus, we may choose the constraint matrix and gain vector to be:

$$C = [\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3, \mathbf{s}_4, \mathbf{s}_5, \mathbf{s}_6]\,, \quad \mathbf{g} = [1, g, g, g, g, g]^T \qquad (14.12.14)$$

where $g = 10^{-60/20} = 10^{-3}$. Compute the corresponding quiescent weights using Eq. (14.12.11) and plot the corresponding array angular pattern in dB as above. Place the points θ_i on the graph.



(e) The quiescent responses of the uniform array considered above are not very good because of their substantial sidelobe level. As an example of a better response, consider the design of a 21-element Dolph-Chebyshev array that is steered towards the angle $\theta_1 = 30^{\mathrm{o}}$ and has a sidelobe level of -40 dB. Such an array can be designed with the MATLAB function `dolph`. The MATLAB code is as follows:

```
d = 0.5; th1 = 30; k1 = 2*pi*d*sin(pi*th1/180);
s1 = steermat(M-1, exp(j*k1));
a = dolph(d, 90-th1, M, 40)';
ad = a/(s1'*a);
```

where the calling convention of `dolph`, and the conjugation implied by the prime operation, have to do with the conventions used in that toolbox. The last line normalizes the designed array vector $\mathbf{a}_d$ so that $\mathbf{s}_1^\dagger\mathbf{a}_d = 1$ and have unity gain towards $\mathbf{s}_1$.

Plot the array response of the desired weights $\mathbf{a}_d$ using the same scales as the above two graphs. Note the -40 dB sidelobe level and the gains at the five constraint points θ_2-θ_6, with the constraints yet to be enforced.

(f) The main idea of Ref. [1213] is to find that weight vector $\bar{\mathbf{a}}$ that satisfies the constraints $C^\dagger\bar{\mathbf{a}} = \mathbf{g}$ and is closest to the desired weight $\mathbf{a}_d$ with respect to the Euclidean norm, that is, find $\bar{\mathbf{a}}$ that is the solution of the minimization problem:

$$J = (\bar{\mathbf{a}} - \mathbf{a}_d)^\dagger(\bar{\mathbf{a}} - \mathbf{a}_d) = \min\,, \quad \text{subject to} \quad C^\dagger\bar{\mathbf{a}} = \mathbf{g} \tag{14.12.15}$$

Using the results of Eq. (14.12.4), show that the optimum solution is

$$\bar{\mathbf{a}} = \mathbf{a}_d + C(C^\dagger C)^{-1}(\mathbf{g} - C^\dagger\mathbf{a}_d) \tag{14.12.16}$$

which can be written in the form:

$$\bar{\mathbf{a}} = [I - C(C^\dagger C)^{-1}C^\dagger]\mathbf{a}_d + C(C^\dagger C)^{-1}\mathbf{g} \equiv \mathbf{a}_\perp + \mathbf{a}_q \tag{14.12.17}$$

Note that $C^\dagger\mathbf{a}_\perp = 0$ and that $\mathbf{a}_q$ and $\mathbf{a}_\perp$ are orthogonal, $\mathbf{a}_q^\dagger\mathbf{a}_\perp = 0$. Show that $\bar{\mathbf{a}}$ can also be written in the form:

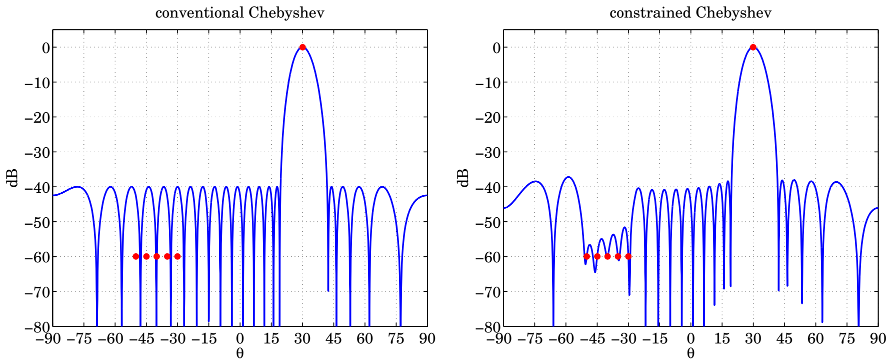
$$\bar{\mathbf{a}} = \mathbf{a}_\perp + C(C^\dagger C)^{-1}\mathbf{g} = [C, \mathbf{a}_\perp]\begin{bmatrix}(C^\dagger C)^{-1} & 0\\ 0 & (\mathbf{a}_\perp^\dagger\mathbf{a}_\perp)^{-1}\end{bmatrix}\begin{bmatrix}\mathbf{g}\\ \mathbf{a}_\perp^\dagger\mathbf{a}_\perp\end{bmatrix}$$

Using the orthogonality property $C^\dagger\mathbf{a}_\perp = 0$, show that the above expression can be written in the form:

$$\bar{\mathbf{a}} = \bar{C}(\bar{C}^\dagger\bar{C})^{-1}\bar{\mathbf{g}}\,, \quad \bar{C} = [C, \mathbf{a}_\perp]\,, \quad \bar{\mathbf{g}} = \begin{bmatrix}\mathbf{g}\\ \mathbf{a}_\perp^\dagger\mathbf{a}_\perp\end{bmatrix} \tag{14.12.18}$$

Therefore, the modified weights $\bar{\mathbf{a}}$ may be thought of as the quiescent weights with respect to the constraints $\bar{C}^\dagger\bar{\mathbf{a}} = \bar{\mathbf{g}}$, which involve one more constraint equation appended to the old ones.

(g) Using the constraints defined by Eq. (14.12.14) and the conventional Chebyshev weights $\mathbf{a}_d$ computed in part (e), construct the new quiescent “constrained Chebyshev” weights according to Eq. (14.12.16) and plot the corresponding array pattern using the same scales as before. Place the constrained points on the graph.



(h) The previous question dealt with the quiescent pattern. Here, assume that there are actually four incident plane waves, one from the desired look-direction θ_1 and three interferers from the directions $\theta_2, \theta_3, \theta_4$ as defined in part (d). Thus, the steering matrix used to construct the covariance matrix R will be $S = [\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3, \mathbf{s}_4]$. All four SNRs are assumed to be 20 dB.

However in this part, we wish to clamp down the three interferers at the -70 dB level. The constraint matrix C remains the same as in Eq. (14.12.14), but the gain vector $\mathbf{g}$ needs to be modified appropriately so that its entries 2–4 reflect the -70 dB requirement.

Construct the extended constraint set $\bar{C}, \bar{\mathbf{g}}$ as in Eq. (14.12.18) and then construct the corresponding optimum weights using Eq. (14.12.8) (with $\bar{C}, \bar{\mathbf{g}}$ in place of $C, \mathbf{g}$). Plot the corresponding angular pattern using the same scales as before, and place the constraint points on the graph.

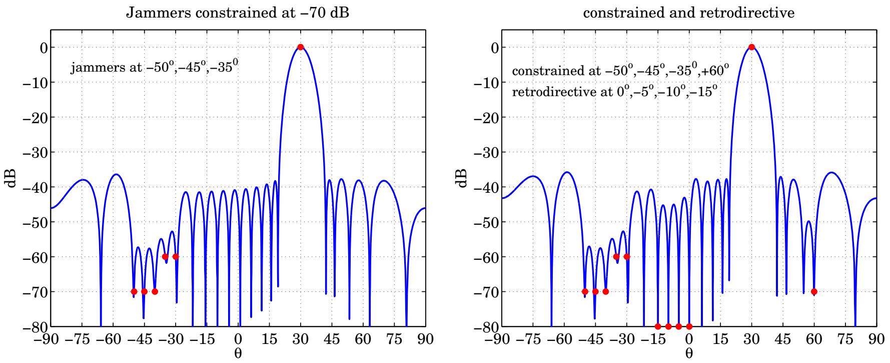
(i) In this part, in addition to the four plane waves of the part (h), there are also incident the following interferers:

$$\{\theta_7, \theta_8, \theta_9, \theta_{10}, \theta_{11}\} = \{-15^{\mathrm{o}}, -10^{\mathrm{o}}, -5^{\mathrm{o}}, 0^{\mathrm{o}}, +60^{\mathrm{o}}\}$$

Again, all SNRs are 20 dB. We wish to meet the following requirements: (1) $\mathbf{s}_1$ is the desired incident signal with unity gain, (2) $\mathbf{s}_2, \mathbf{s}_3, \mathbf{s}_4$ are incident waves to be clamped down at -70 dB, (3) $\mathbf{s}_5, \mathbf{s}_6$ whose angles were defined in part (d) are constraint directions clamped at -60 dB but they do not correspond to incident waves, (4) $\mathbf{s}_7, \mathbf{s}_8, \mathbf{s}_9, \mathbf{s}_{10}$ are incident and will be nulled by the retrodirective action of the array, and (5) $\mathbf{s}_{11}$ is incident and also to be clamped at -70 dB.

Construct the covariance matrix R and the new constraints $C, \mathbf{g}$. Then, construct the extended constraint set $\bar{C}, \bar{\mathbf{g}}$, calculate the optimum weights from

Eq. (14.12.8), and plot the corresponding array pattern using the same scales as before. Indicate the constraint points on the graph. Label the four retrodirective beam points at the bottom of the graph, that is, at the −80 dB level.



4. *Equivalence of LCMV and the Generalized Sidelobe Canceler.* Finally, we look at the equivalence of the LCMV beamformer and the generalized sidelobe canceler (GSC). Consider the LCMV problem defined by Eq. (14.12.7) and its solution, Eq. (14.12.8). The constraints $C^\dagger\mathbf{a} = \mathbf{g}$ can be regarded as a full-rank under-determined system of equations.

   (a) Show that the pseudoinverse of the constraint matrix $C^\dagger$ is:

   $$(C^\dagger)^+ = C(C^\dagger C)^{-1}$$

   (b) Show that the most general solution of $C^\dagger\mathbf{a} = \mathbf{g}$ can be expressed in the form:

   $$\mathbf{a} = \mathbf{a}_q - B\mathbf{c} \tag{14.12.19}$$

   where $\mathbf{a}_q = (C^\dagger)^+\mathbf{g} = C(C^\dagger C)^{-1}\mathbf{g}$ is the minimum-norm solution and B is an $M\times(M-K)$ matrix that forms a basis for the $(M-K)$-dimensional null space of $C^\dagger$, that is, the space $N(C^\dagger)$, and $\mathbf{c}$ is an arbitrary $(M-K)$-dimensional vector of coefficients. Thus, B must satisfy

   $$C^\dagger B = 0 \tag{14.12.20}$$

   For example, B may be constructed from the full SVD, $C = U\Sigma V^\dagger$, where the $M\times M$ unitary matrix U can be decomposed into its $M\times K$ and $M\times(M-K)$ parts, $U = [U_1, U_2]$. One can choose then $B = U_2$, or more generally, $B = U_2F$, where F is any $(M-K)\times(M-K)$ invertible matrix.

   (c) Because $\mathbf{a} = \mathbf{a}_q - B\mathbf{c}$ already satisfies the constraints, the LCMV problem (14.12.7) can be replaced by the following *unconstrained* minimization problem for the determination of the coefficients $\mathbf{c}$:

   $$J = \mathbf{a}^\dagger R\mathbf{a} = (\mathbf{a}_q - B\mathbf{c})^\dagger R(\mathbf{a}_q - B\mathbf{c}) = \min \tag{14.12.21}$$

   Show that the optimum $\mathbf{c}$ is given by

   $$\mathbf{c} = (B^\dagger RB)^{-1}B^\dagger R\mathbf{a}_q \tag{14.12.22}$$

   and hence the optimum $\mathbf{a}$ can be written in the form:

   $$\mathbf{a} = \mathbf{a}_q - B\mathbf{c} = \big[I - B(B^\dagger RB)^{-1}B^\dagger R\big]\mathbf{a}_q \tag{14.12.23}$$

   The solution (14.12.23) must be the same as that of Eq. (14.12.8).

   (d) Show that the orthogonality condition (14.12.20) and the fact that the two matrices C, B together form a basis for $\mathbb{C}^M$, imply that B and C must satisfy the following relationships:

   $$C(C^\dagger C)^{-1}C^\dagger + B(B^\dagger B)^{-1}B^\dagger = I \tag{14.12.24}$$

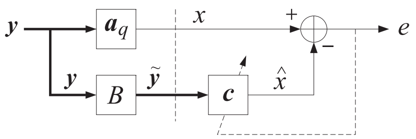
   and

   $$R^{-1}C(C^\dagger R^{-1}C)^{-1}C^\dagger + B(B^\dagger RB)^{-1}B^\dagger R = I \tag{14.12.25}$$

   Hints: For the first one, use the full SVD of C and the fact that $B = U_2F$. For the second one, work with the matrices $\bar{C} = R^{-1/2}C$ and $\bar{B} = R^{1/2}B$, where $R^{1/2}$ is a hermitian positive square root of R, and then apply the same argument as for the first part.

   (e) Using Eq. (14.12.25) show the equivalence of Eqs. (14.12.8) and (14.12.23).

   (f) The above results lead to the following block diagram realization of the LCMV, known as the generalized sidelobe canceler:



   where $\mathbf{y}$ is the overall input vector at the M antennas, and the other variables are defined as follows:

   $$\boxed{\begin{aligned} &x = \mathbf{a}_q^T\mathbf{y} \\ &\tilde{\mathbf{y}} = B^T\mathbf{y} \\ &\hat{x} = \mathbf{c}^T\tilde{\mathbf{y}} \\ &e = x - \hat{x} = \mathbf{a}_q^T\mathbf{y} - \mathbf{c}^T B^T\mathbf{y} = (\mathbf{a}_q - B\mathbf{c})^T\mathbf{y} = \mathbf{a}^T\mathbf{y} \end{aligned}} \tag{14.12.26}$$

   The portion of the block diagram to the right of the vertical dividing line may be thought of as an ordinary Wiener filtering problem of estimating the signal x from the vector $\tilde{\mathbf{y}}$. This follows by noting that the output power minimization of e is equivalent to the estimation problem:

   $$J = \mathbf{a}^T R\mathbf{a} = E[|e|^2] = E[|x - \hat{x}|^2] = \min$$

where $R = E[\mathbf{y}^*\mathbf{y}^T]$. Let $\tilde{R} = E[\tilde{\mathbf{y}}^*\tilde{\mathbf{y}}^T]$ and $\tilde{\mathbf{r}} = E[x\tilde{\mathbf{y}}^*]$. Then, show that the optimum $\mathbf{c}$ of Eq. (14.12.22) is indeed the Wiener solution:

$$\mathbf{c} = \tilde{R}^{-1}\tilde{\mathbf{r}} \tag{14.12.27}$$

The great advantage of the GSC is that this unconstrained Wiener filtering part can be implemented adaptively using any adaptive method such as LMS or RLS, applied to the signals $x(n), \tilde{\mathbf{y}}(n)$. For example, the complete LMS algorithm would be as follows. At each time n, the input $\mathbf{y}(n)$ and weights $\mathbf{c}(n)$ are available, then,

$$\boxed{\begin{aligned} &x(n) = \mathbf{a}_q^T\mathbf{y}(n)\\ &\tilde{\mathbf{y}}(n) = B^T\mathbf{y}(n)\\ &\hat{x}(n) = \mathbf{c}^T(n)\tilde{\mathbf{y}}(n)\\ &e(n) = x(n) - \hat{x}(n)\\ &\mathbf{c}(n+1) = \mathbf{c}(n) + \mu e(n)\tilde{\mathbf{y}}^*(n) \end{aligned}} \qquad \text{(adaptive GSC)}$$

with the vector $\mathbf{a}(n) = \mathbf{a}_q - B\mathbf{c}(n)$ converging to the desired optimum constrained solution of Eq. (14.12.8).

14.13 *Computer Project – Markowitz Portfolio Theory*

This project, divided into separate questions, deals with Markowitz's optimum mean-variance portfolio theory. [1222–1233]. It can be considered to be a special case of the linearly-constrained quadratic optimization problem of the previous project. The project develops the following topics from financial engineering:

- optimum mean-variance Markowitz portfolios
- efficient frontier between risk and return
- quantifying risk aversion
- two mutual fund theorem
- inequality-constrained portfolios without short selling
- risk-free assets and tangency portfolio
- capital asset line and the Sharp ratio
- market portfolios and capital market line
- stock's beta, security market line, risk premium
- capital asset pricing model (CAPM)
- minimum-variance with multiple constraints

1. *Mean-Variance Portfolio Theory.* The following constrained optimization problem finds application in investment analysis and optimum portfolio selection in which one tries to balance return versus risk.[†]

[†] Harry Markowitz received the Nobel prize in economics for this work.

Suppose one has identified M stocks or assets $\mathbf{y} = [y_1, y_2, \dots, y_M]^T$ into which to invest. From historical data, the expected returns of the individual stocks are assumed to be known, $E[\mathbf{y}] = \mathbf{m} = [m_1, m_2, \dots, m_M]^T$, as are the cross-correlations between the assets, $R_{ij} = E\big[(y_i - m_i)(y_j - m_j)\big]$, or, $R = E\big[(\mathbf{y}-\mathbf{m})(\mathbf{y}-\mathbf{m})^T\big]$, assumed to have full rank. The variance $\sigma_i^2 = R_{ii}$ is a measure of the volatility, or risk, of the ith asset.

A portfolio is selected by choosing the percentage a_i to invest in the ith asset y_i. The portfolio is defined by the random variable:

$$y = \sum_{i=1}^{M} a_i y_i = [a_1, a_2, \dots, a_M]\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_M \end{bmatrix} = \mathbf{a}^T\mathbf{y}$$

where the weights a_i must add up to unity (negative weights are allowed, describing so-called "short sells"):

$$\sum_{i=1}^{M} a_i = [a_1, a_2, \dots, a_M]\begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} = \mathbf{a}^T\mathbf{u} = 1$$

The expected return of the portfolio and its variance or risk are given by:

$$\begin{aligned} \mu &= E[y] = E[\mathbf{a}^T\mathbf{y}] = \mathbf{a}^T\mathbf{m} \\ \sigma^2 &= E\big[(y-\mu)^2\big] = \mathbf{a}^T R\mathbf{a} \end{aligned}$$

An optimum portfolio may be defined by finding the weights that minimize the risk σ^2 for a given value of the return μ, that is,

$$\sigma^2 = \mathbf{a}^T R\,\mathbf{a} = \min\,, \quad \text{subject to} \quad \mathbf{a}^T\mathbf{m} = \mu\,, \quad \mathbf{a}^T\mathbf{u} = 1 \tag{14.13.1}$$

(a) Incorporate the constraints by means of two Lagrange multipliers, λ_1, λ_2, and minimize the modified performance index:

$$J = \frac{1}{2}\mathbf{a}^T R\,\mathbf{a} + \lambda_1(\mu - \mathbf{a}^T\mathbf{m}) + \lambda_2(1 - \mathbf{a}^T\mathbf{u}) = \min$$

Show that the quantities $\{\mathbf{a}, \lambda_1, \lambda_2\}$ can be obtained from the solution of the $(M+2)\times(M+2)$ linear system of equations:

$$\begin{bmatrix} R & -\mathbf{m} & -\mathbf{u} \\ \mathbf{m}^T & 0 & 0 \\ \mathbf{u}^T & 0 & 0 \end{bmatrix}\begin{bmatrix} \mathbf{a} \\ \lambda_1 \\ \lambda_2 \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mu \\ 1 \end{bmatrix}$$

where the invertibility of this matrix requires that the vectors $\mathbf{m}$ and $\mathbf{u}$ be not collinear.

(b) Show that the solution of this system for **a** takes the form:

$$\mathbf{a} = \lambda_1 R^{-1}\mathbf{m} + \lambda_2 R^{-1}\mathbf{u} \qquad (14.13.2)$$

and that λ_1, λ_2 can be obtained from the reduced 2×2 linear system:

$$\begin{bmatrix} A & B \\ B & C \end{bmatrix}\begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix} = \begin{bmatrix} \mu \\ 1 \end{bmatrix} \quad \Rightarrow \quad \begin{array}{l} \lambda_1 = \dfrac{\mu C - B}{D} \\[2ex] \lambda_2 = \dfrac{A - \mu B}{D} \end{array}$$

where A, B, C, D are defined in terms of $\mathbf{m}, R$ by

$$\begin{aligned} A &= \mathbf{m}^T R^{-1}\mathbf{m} \\ B &= \mathbf{m}^T R^{-1}\mathbf{u} \qquad \text{and} \quad D = AC - B^2 \\ C &= \mathbf{u}^T R^{-1}\mathbf{u} \end{aligned}$$

(c) Show that the quantities A, C, D are non-negative.

(d) Show that the minimized value of the risk $\sigma^2 = \mathbf{a}^T R\mathbf{a}$ can be written in the form:

$$\sigma^2 = \mu\lambda_1 + \lambda_2 = \frac{C\mu^2 - 2B\mu + A}{D} \qquad (14.13.3)$$

Thus, the dependence of the variance σ^2 on the return μ has a parabolic shape, referred to as the *efficient frontier*.

(e) The apex of this parabola is obtained by minimizing Eq. (14.13.3) with respect to μ. Setting $\partial\sigma^2/\partial\mu = 0$, show that the absolute minimum is reached for the following values of the return, risk, and weights:

$$\mu_0 = \frac{B}{C}, \quad \sigma_0^2 = \frac{1}{C}, \quad \mathbf{a}_0 = \frac{R^{-1}\mathbf{u}}{\mathbf{u}^T R^{-1}\mathbf{u}} \qquad (14.13.4)$$

(f) Show that Eq. (14.13.3) can be re-expressed as

$$\sigma^2 = \sigma_0^2 + \frac{C}{D}(\mu - \mu_0)^2 \qquad (14.13.5)$$

which can be solved for μ in terms of σ^2, as is common in practice:

$$\mu = \mu_0 \pm \sqrt{\frac{D}{C}}\sqrt{\sigma^2 - \sigma_0^2} \qquad (14.13.6)$$

Of course, only the upper sign corresponds to the efficient frontier because it yields higher return for the same risk. (See some example graphs below.)

(g) Show that the optimum portfolio of Eq. (14.13.2) can be written in the form:

$$\mathbf{a} = \mathbf{a}_0 + \frac{C}{D}(\mu - \mu_0)\, R^{-1}(\mathbf{m} - \mu_0\,\mathbf{u}) \qquad (14.13.7)$$

where $\mathbf{a}_0$ was defined in Eq. (14.13.4). Thus, as expected, $\mathbf{a} = \mathbf{a}_0$, if $\mu = \mu_0$.

(h) Show that the optimum portfolio of Eq. (14.13.2) can be written in the form

$$\mathbf{a} = \mu\,\mathbf{g} + \mathbf{h}$$

where $\mathbf{g}, \mathbf{h}$ depend only on the asset statistics $\mathbf{m}, R$ and are independent of μ. Moreover, show that $\mathbf{m}^T\mathbf{g} = 1$ and $\mathbf{m}^T\mathbf{h} = 0$, and that $\mathbf{u}^T\mathbf{g} = 0$ and $\mathbf{u}^T\mathbf{h} = 1$. In particular, show that $\mathbf{g}, \mathbf{h}$ are given by,

$$\begin{aligned} \mathbf{g} &= \frac{C}{D}R^{-1}\mathbf{m} - \frac{B}{D}R^{-1}\mathbf{u} \\ \mathbf{h} &= \frac{A}{D}R^{-1}\mathbf{u} - \frac{B}{D}R^{-1}\mathbf{m} \end{aligned}$$

(i) Consider two optimal portfolios $\mathbf{a}_1$ and $\mathbf{a}_2$ having return-risk values that lie on the efficient frontier, μ_1, σ_1 and μ_2, σ_2, satisfying Eq. (14.13.5), and assume that $\mu_1 < \mu_2$. Using the results of the previous question, show that any other optimum portfolio with return-risk pair μ, σ, such that $\mu_1 < \mu < \mu_2$, can be constructed as a linear combination of $\mathbf{a}_1, \mathbf{a}_2$ as follows, with positive weights p_1, p_2, such that, $p_1 + p_2 = 1$,

$$\mathbf{a} = p_1\mathbf{a}_1 + p_2\mathbf{a}_2\,, \quad p_1 = \frac{\mu_2 - \mu}{\mu_2 - \mu_1}\,, \quad p_2 = \frac{\mu - \mu_1}{\mu_2 - \mu_1}$$

Thus, the investor need only invest in the two standard portfolios $\mathbf{a}_1$ and $\mathbf{a}_2$ in the proportions p_1 and p_2, respectively. This is known as the *two mutual fund theorem*. The restriction $\mu_1 < \mu < \mu_2$ can be relaxed if short selling of the funds is allowed.

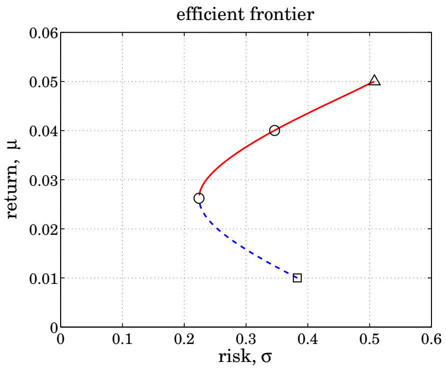
(j) Consider a portfolio of four assets having return (given as annual rate) and covariance matrix:

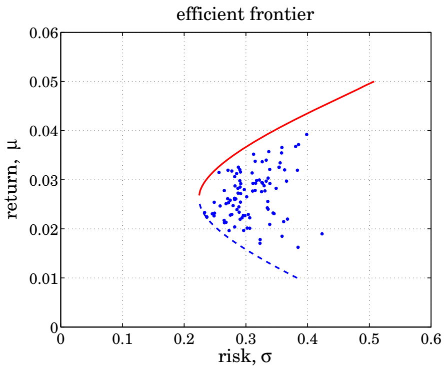
$$\mathbf{m} = \begin{bmatrix} 0.02 \\ 0.03 \\ 0.01 \\ 0.05 \end{bmatrix}, \quad R = \begin{bmatrix} 0.10 & -0.02 & 0.04 & -0.01 \\ -0.02 & 0.20 & 0.05 & 0.02 \\ 0.04 & 0.05 & 0.30 & 0.03 \\ -0.01 & 0.02 & 0.03 & 0.40 \end{bmatrix}$$

Make a plot of the efficient frontier of μ versus σ according to Eq. (14.13.6). To do so, choose 51 equally-spaced μs in the interval $[\min(\mathbf{m}), \max(\mathbf{m})]$ and calculate the corresponding σs.

Compute the risk σ and weight vector $\mathbf{a}$ that would achieve a return of $\mu = 0.04$ and indicate that point on the efficient frontier. Why wouldn't an investor want to put all his/her money in stock y_4 since it has a higher return of $m_4 = 0.05$?

(k) Generate 100 weight vectors $\mathbf{a}$ randomly (but such that $\mathbf{a}^T\mathbf{u} = 1$), compute the values of the quantities $\mu = \mathbf{a}^T\mathbf{m}$ and $\sigma^2 = \mathbf{a}^T R\mathbf{a}$ and make a scatterplot of the points (μ, σ) to see that they lie on or below the efficient frontier.





2. *Risk aversion.* A somewhat different minimization criterion is often chosen for the portfolio selection problem, that is,

$$J = \frac{1}{2}\mathbf{a}^T R\mathbf{a} - \gamma\mathbf{a}^T\mathbf{m} = \min\,, \quad \text{subject to } \mathbf{u}^T\mathbf{a} = 1 \tag{14.13.8}$$

where γ is a positive parameter that quantifies the investor's aversion for risk versus return—small γ emphasizes low risk, larger γ, high return.

For various values of γ, the optimum weights $\mathbf{a}$ are determined and then the corresponding return $\mu = \mathbf{a}^T\mathbf{m}$ and risk $\sigma^2 = \mathbf{a}^T\Sigma\mathbf{a}$ are calculated. The resulting plot of the pairs (μ, σ) is the efficient frontier in this case.

Given the parameters $\gamma, \mathbf{m}, R$, incorporate the constraint, $\mathbf{a}^T\mathbf{u} = 1$, with a Lagrange multiplier and work with the modified performance index:

$$J = \frac{1}{2}\mathbf{a}^T R\mathbf{a} - \gamma\mathbf{a}^T\mathbf{m} + \lambda_2(1 - \mathbf{u}^T\mathbf{a}) = \min$$

Show that one obtains exactly the same solution for $\mathbf{a}$ as in the previous problem and that the correspondence between the assumed γ and the realized return μ is given by

$$\mu = \mu_0 + \frac{D}{C}\gamma$$

3. *Portfolio with inequality constraints.* If short sales are not allowed, the portfolio weights a_i must be restricted to be in the interval $0 \le a_i \le 1$. In this case, the following optimization problem must be solved:

$$\sigma^2 = \mathbf{a}^T R\mathbf{a} = \min\,, \quad \text{subject to } \begin{cases} \mathbf{a}^T\mathbf{m} = \mu \\ \mathbf{a}^T\mathbf{u} = 1 \\ 0 \le a_i \le 1\,, \quad i = 1, 2, \dots, M \end{cases} \tag{14.13.9}$$

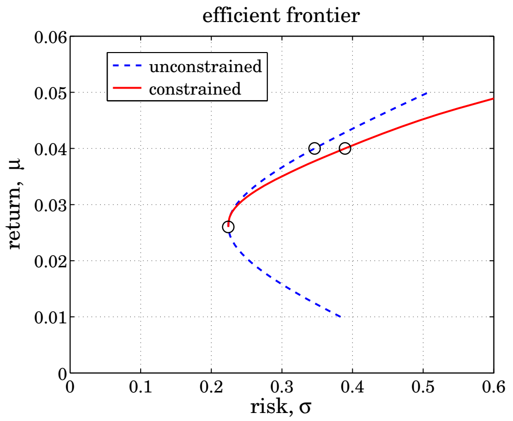
This is a convex optimization problem that can be solved conveniently using the CVX package.[†] For example, given a desired value for μ, the following CVX code will solve for $\mathbf{a}$:

[†] http://cvxr.com/cvx

```
% define M, m, R, mu
cvx_begin
   variable a(M)
   minimize( a'*R*a );
   subject to
      a'*u == 1;
      a'*m == mu;
      a <= ones(M,1);
     -a <= zeros(M,1);
cvx_end
```

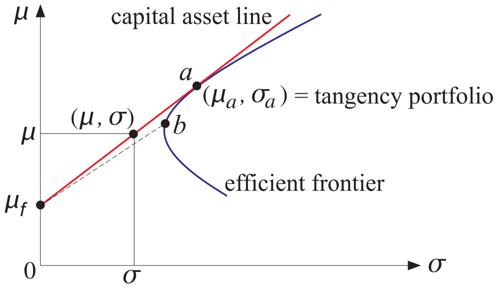
(a) For the numerical example of Question (1.j), choose 51 equally-spaced μs in the interval $[\min(\mathbf{m}), \max(\mathbf{m})]$ and calculate the corresponding $\mathbf{a}$ and σs, and plot the efficient frontier, that is the pairs (μ, σ). Superimpose on this graph the efficient frontier of the unconstrained case from Question 1.

(b) Compute the risk σ and weight vector $\mathbf{a}$ that would achieve a return of $\mu = 0.04$ and indicate that point on the efficient frontier of part (a). Place on the graph also the unconstrained solution for the same μ, and explain why the inequality-constrained case is slightly worse.



4. *Capital Asset Line.* A variation of the mean-variance portfolio theory was considered by W. F. Sharpe, who in addition to the collection of risky assets, allowed the presence of a *risk-free* asset, such as a Treasury T-bill, that has a fixed guaranteed return, say $\mu = \mu_f$, and no risk, $\sigma_f = 0$.

Let us assume again that we also have M risky assets $\mathbf{y} = [y_1, y_2, \dots, y_M]^T$ with expected returns $E[\mathbf{y}] = \mathbf{m} = [m_1, m_2, \dots, m_M]^T$, and known covariance matrix $R = E[(\mathbf{y} - \mathbf{m})(\mathbf{y} - \mathbf{m})^T]$. Clearly, we must assume that $\mu_f < m_i$, $i = 1, 2, \dots, M$, otherwise, we would put all our money into the risk-free asset. We form an optimum portfolio of risky assets $y = \mathbf{a}^T\mathbf{y}$ by choosing a point on the efficient frontier, for example, the indicated point b on the figure below.



Then, we join with a straight line the point b to the risk-free point μ_f (the dashed line on the figure), and we allocate a fraction w_f of our total funds to the risk-free asset and hence a fraction $(1-w_f)$ to the portfolio y, that is, the combined portfolio will be:

$$y_{\rm tot} = w_f\mu_f + (1-w_f)y = w_f\mu_f + (1-w_f)\mathbf{a}^T\mathbf{y}$$

with mean and variance:

$$\begin{aligned}\mu &= E[y_{\rm tot}] = w_f\mu_f + (1-w_f)\mathbf{a}^T\mathbf{m}\\ \sigma^2 &= E\big[(y_{\rm tot}-\mu)^2\big] = (1-w_f)^2\,\mathbf{a}^T R\mathbf{a}\end{aligned} \tag{14.13.10}$$

The lowest location for b is at the apex of the frontier, for which we have $\mathbf{a} = \mathbf{a}_0$. It should be evident from the figure that if we move the point b upwards along the efficient frontier, increasing the slope of the straight line, we would obtain a better portfolio having larger μ.

We may increase the slope until the line becomes tangent to the efficient frontier, say at the point a, which would correspond to an optimum portfolio $\mathbf{a}$ with mean and variance μ_a, σ_a.

This portfolio is referred to as the *tangency portfolio* and the tangent line (red line) is referred to as the *capital asset line* and its maximum slope, say β, as the *Sharpe ratio*. Eqs. (14.13.10) become now:

$$\begin{aligned}\mu &= w_f\mu_f + (1-w_f)\mu_a\\ \sigma &= (1-w_f)\sigma_a\end{aligned} \tag{14.13.11}$$

where $\mu_a = \mathbf{a}^T\mathbf{m}$ and $\sigma_a^2 = \mathbf{a}^T R\mathbf{a}$. Solving the second of Eq. (14.13.11) for w_f, we find $1-w_f = \sigma/\sigma_a$, and substituting in the first, we obtain the equation for the straight line on the (μ,σ) plane:

$$\mu = \mu_f + \beta\sigma\,, \quad \beta = \frac{\mu_a-\mu_f}{\sigma_a} = \text{slope} \tag{14.13.12}$$

This line is the *efficient frontier* for the combined portfolio. Next we impose the condition that the point a be a tangency point on the efficient frontier. This will fix μ_a, σ_a. We saw that the frontier is characterized by the parabolic curve of Eq. (14.13.5) with the optimum weight vector given by (14.13.7). Applying these to the pair (μ_a, σ_a), we have:

$$\begin{aligned}\sigma_a^2 &= \sigma_0^2 + \frac{C}{D}(\mu_a-\mu_0)^2\\ \mathbf{a} &= \mathbf{a}_0 + \frac{C}{D}(\mu_a-\mu_0)\,R^{-1}(\mathbf{m}-\mu_0\mathbf{u})\end{aligned} \tag{14.13.13}$$

The slope of the tangent at the point a is given by the derivative $d\mu_a/d\sigma_a$, and it must agree with the slope β of the line (14.13.12):

$$\frac{d\mu_a}{d\sigma_a} = \beta = \frac{\mu_a-\mu_f}{\sigma_a} \tag{14.13.14}$$

(a) Using condition (14.13.14) and Eq. (14.13.13), show the following relationships:

$$\begin{aligned}&\frac{C}{D}(\mu_a-\mu_0)(\mu_0-\mu_f) = \frac{1}{C}\\ &\sigma_a^2 = \frac{C}{D}(\mu_a-\mu_0)(\mu_a-\mu_f)\\ &\beta = \sigma_a C(\mu_0-\mu_f)\end{aligned} \tag{14.13.15}$$

The first can be solved for μ_a, and show that it can be expressed as:

$$\mu_a = \frac{A-\mu_f B}{C(\mu_0-\mu_f)} \tag{14.13.16}$$

(b) Using Eqs. (14.13.13) and (14.13.15), show that the optimum weights are given by

$$\mathbf{a} = \frac{1}{C(\mu_0-\mu_f)}\,R^{-1}(\mathbf{m}-\mu_f\mathbf{u}) \tag{14.13.17}$$

and verify that they satisfy the constraints $\mathbf{a}^T\mathbf{m} = \mu_a$ and $\mathbf{a}^T\mathbf{u} = 1$.

(c) Show that the slope β can also be expressed by:

$$\beta^2 = \left(\frac{\mu_a-\mu_f}{\sigma_a}\right)^2 = (\mathbf{m}-\mu_f\mathbf{u})^T R^{-1}(\mathbf{m}-\mu_f\mathbf{u}) \tag{14.13.18}$$

(d) Define $\mathbf{w} = (1-w_f)\mathbf{a}$ to be the effective weight for the total portfolio:

$$y_{\rm tot} = w_f\mu_f + (1-w_f)\mathbf{a}^T\mathbf{y} = w_f\mu_f + \mathbf{w}^T\mathbf{y} \tag{14.13.19}$$

Show that $\mathbf{w}$ is given in terms of the return $\mu = w_f\mu_f + (1-w_f)\mu_a$ as follows:

$$\mathbf{w} = (1-w_f)\mathbf{a} = \frac{\mu-\mu_f}{\beta^2}\,R^{-1}(\mathbf{m}-\mu_f\mathbf{u}) \tag{14.13.20}$$

(e) The optimality of the capital asset line and the tangency portfolio can also be derived directly by considering the following optimization problem. Let w_f and $\mathbf{w}$ be the weights to be assigned to the risk-free asset μ_f and the risky assets $\mathbf{y}$. Then the total portfolio, its mean μ, and variance σ^2 are given by:

$$\begin{aligned} y_{\text{tot}} &= w_f\mu_f + \mathbf{w}^T\mathbf{y} \\ \mu &= w_f\mu_f + \mathbf{w}^T\mathbf{m} \\ \sigma^2 &= \mathbf{w}^T R\mathbf{w} \end{aligned} \tag{14.13.21}$$

where the weights must add up to unity: $w_f + \mathbf{w}^T\mathbf{u} = 1$. Given μ, we wish to determine the weights $w_f, \mathbf{w}$ to minimize σ^2. Incorporating the constraints by two Lagrange multipliers, we obtain the performance index:

$$J = \frac{1}{2}\mathbf{w}^T R\mathbf{w} + \lambda_1(\mu - w_f\mu_f - \mathbf{w}^T\mathbf{m}) + \lambda_2(1 - w_f - \mathbf{w}^T\mathbf{u}) = \min$$

Show that the minimization of J with respect to $w_f, \mathbf{w}$ results in:

$$\mathbf{w} = \lambda_1 R^{-1}(\mathbf{m} - \mu_f\mathbf{u}), \quad \lambda_2 = -\lambda_1\mu_f$$

(f) Imposing the constraints show that,

$$\mathbf{w}^T(\mathbf{m} - \mu_f\mathbf{u}) = \mu - \mu_f, \quad \lambda_1 = \frac{\mu - \mu_f}{\beta^2}$$

where β^2 is given as in Eq. (14.13.18),

$$\beta^2 = (\mathbf{m} - \mu_f\mathbf{u})^T R^{-1}(\mathbf{m} - \mu_f\mathbf{u})$$

and hence, show that $\mathbf{w}$ is given by Eq. (14.13.20)

$$\mathbf{w} = \frac{\mu - \mu_f}{\beta^2} R^{-1}(\mathbf{m} - \mu_f\mathbf{u})$$

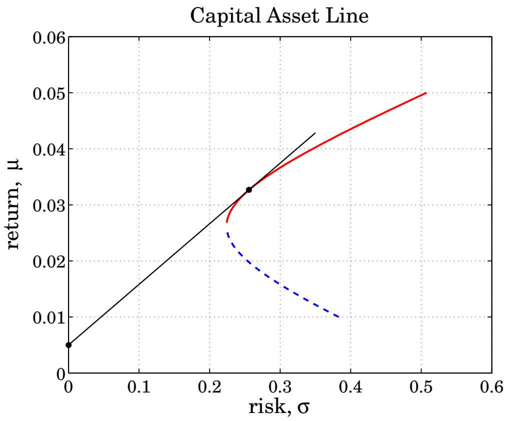
(g) Show that $\sigma^2 = \mathbf{w}^T R\mathbf{w}$ is given by,

$$\sigma^2 = \left(\frac{\mu - \mu_f}{\beta}\right)^2$$

which corresponds to the straight-line frontier on the μ, σ plane:

$$\mu = \mu_f + \beta\sigma$$

(h) For the numerical example of Question (1.j) and for a fixed-asset return of $\mu_f = 0.005$, connect the points (μ_a, σ_a) and $(\mu_f, 0)$ by a straight line and plot it together with the parabolic efficient frontier of the risky assets, and verify the tangency of the line.



5. *Security market line and CAPM.* When the collection of risky stocks of the previous problem are taken to be representative of the market, such as for example, the stocks in the S&P 500 index, the tangency portfolio at point *a* is referred to as the *market portfolio*, and the capital asset line, as the *capital market line.*

Let $y_a = \mathbf{a}^T\mathbf{y}$ be the linear combination of stocks for the market portfolio, and consider another stock y_i with return and risk μ_i, σ_i. The stock y_i is arbitrary and not necessarily belonging to those that make up the representative market. Define the *beta* for the stock as the ratio of the following covariances relative to the market portfolio:

$$\beta_i = \frac{R_{ia}}{R_{aa}}$$

where $R_{ia} = E[(y_i - \mu_i)(y_a - \mu_a)]$ and $R_{aa} = \sigma_a^2 = E[(y_a - \mu_a)^2]$.

Let us now make a portfolio y consisting of a percentage w of the stock y_i and a percentage $(1 - w)$ of the market y_a. Then, y and its mean and variance will be given as follows, where $R_{ii} = \sigma_i^2$.

$$\begin{aligned} y &= wy_i + (1 - w)y_a \\ \mu &= w\mu_i + (1 - w)\mu_a \\ \sigma^2 &= w^2R_{ii} + 2w(1 - w)R_{ia} + (1 - w)^2R_{aa} \end{aligned} \tag{14.13.22}$$

As w varies, the pair (μ, σ) varies over its own parabolic efficient frontier.

The *capital asset pricing model* (CAPM) asserts that the tangent line at the market point $y = y_a$ on this frontier obtained when $w = 0$, coincides with the capital market line between the risk-free asset μ_f and the market portfolio. This establishes the following relationship to be proved below:

$$\mu_i - \mu_f = \beta_i(\mu_a - \mu_f) \tag{14.13.23}$$

so that the excess return above μ_f, called the *risk premium*, is proportional to the stock's beta. The straight line of μ_i vs. β_i is referred to as the *security market line.*

(a) Using the differentiation rule,

$$\frac{d\sigma}{d\mu}=\frac{d\sigma}{dw}\cdot\frac{dw}{d\mu}$$

show that the slope at the market point, i.e., at $w = 0$, is given by

$$\left.\frac{d\mu}{d\sigma}\right|_{w=0}=\frac{\mu_i-\mu_a}{(\beta_i-1)\sigma_a} \tag{14.13.24}$$

(b) Then, derive Eq. (14.13.23) by equating (14.13.24) to the slope (14.13.12).

6. *Minimum-variance with multiple constraints.* A generalization of the portfolio constrained minimization problem involves more that two constraints:

$$J=\frac{1}{2}\mathbf{a}^TR\mathbf{a}-\mathbf{b}^T\mathbf{a}=\min$$

subject to K linear constraints:

$$\mathbf{c}_i^T\mathbf{a}=\mu_i\,,\quad i=1,2,\dots,K$$

where R is an $M{\times}M$ positive definite symmetric matrix, $\mathbf{b}$ is a given $M{\times}1$ vector, and we assume that $K < M$ and that the $M{\times}1$ vectors $\mathbf{c}_i$ are linearly independent. Defining the $M{\times}K$ matrix of constraints C and the K-dimensional vector of "returns" $\boldsymbol{\mu}$,

$$C=[\mathbf{c}_1,\mathbf{c}_2,\dots,\mathbf{c}_K]\,,\quad \boldsymbol{\mu}=\begin{bmatrix}\mu_1\\ \mu_2\\ \vdots\\ \mu_K\end{bmatrix}$$

the above minimization problem can be cast compactly in the form:

$$J=\frac{1}{2}\mathbf{a}^TR\mathbf{a}-\mathbf{b}^T\mathbf{a}=\min\,,\quad\text{subject to}\quad C^T\mathbf{a}=\boldsymbol{\mu}$$

(a) Introduce a K-dimensional vector of Lagrange multipliers $\boldsymbol{\lambda}$ and replace the performance index by:

$$J=\frac{1}{2}\mathbf{a}^TR\mathbf{a}-\mathbf{b}^T\mathbf{a}+\boldsymbol{\lambda}^T(\boldsymbol{\mu}-C^T\mathbf{a})=\min$$

Show that the quantities $\mathbf{a},\boldsymbol{\lambda}$ may be obtained as the solution of the $(M{+}K)\times(M{+}K)$ linear system of equations:

$$\begin{bmatrix}R & -C\\ C^T & \mathbf{0}\end{bmatrix}\begin{bmatrix}\mathbf{a}\\ \boldsymbol{\lambda}\end{bmatrix}=\begin{bmatrix}\mathbf{b}\\ \boldsymbol{\mu}\end{bmatrix}$$

(b) Show that the above matrix has the following inverse:

$$\left[\begin{array}{c|c} R^{-1}-R^{-1}C(C^TR^{-1}C)^{-1}C^TR^{-1} & R^{-1}C(C^TR^{-1}C)^{-1}\\ \hline -(C^TR^{-1}C)^{-1}C^TR^{-1} & (C^TR^{-1}C)^{-1}\end{array}\right]$$

and explain why the assumed full rank of C guarantees the existence of the matrix inverse $(C^TR^{-1}C)^{-1}$.

(c) Show that the solution for $\mathbf{a}$ and $\boldsymbol{\lambda}$ can be obtained by:

$$\begin{aligned}\boldsymbol{\lambda}&=(C^TR^{-1}C)^{-1}[\boldsymbol{\mu}-C^TR^{-1}\mathbf{b}]\\ \mathbf{a}&=R^{-1}[\mathbf{b}+C\boldsymbol{\lambda}]\end{aligned}$$

(d) Show that the "variance" $\sigma^2=\mathbf{a}^TR\mathbf{a}$ is parabolic in the "returns", like Eq. (14.13.3), thus defining a generalized "efficient frontier":

$$\sigma^2=\sigma_0^2+\boldsymbol{\mu}^T(C^TR^{-1}C)^{-1}\boldsymbol{\mu}$$

where the constant σ_0^2 is defined by:

$$\sigma_0^2=\mathbf{b}^T[R^{-1}-R^{-1}C(C^TR^{-1}C)^{-1}C^TR^{-1}]\mathbf{b}$$

Note that if additional inequality constraints are included, such as for example $a_i > 0$ for the weights, then this becomes a much harder problem that must be solved with *quadratic programming* techniques. The CVX package or MATLAB's function `quadprog` from the optimization toolbox solves such problems. The antenna or sensor array version of this problem is LCMV beamforming.

14.14 Problems

14.1 *Computer Experiment.* A fourth-order autoregressive process is defined by the difference equation

$$y_n+a_1y_{n-1}+a_2y_{n-2}+a_3y_{n-3}+a_4y_{n-4}=\epsilon_n$$

where ϵ_n is zero-mean, unit-variance, white gaussian noise. The filter parameters $\{a_1,a_2,a_3,a_4\}$ are chosen such that the prediction error filter

$$A(z)=1+a_1z^{-1}+a_2z^{-2}+a_3z^{-3}+a_4z^{-4}$$

has zeros at the locations

$$0.99\exp(\pm 0.2\pi j)\quad\text{and}\quad 0.99\exp(\pm 0.4\pi j)$$

(a) Determine $\{a_1,a_2,a_3,a_4\}$.

(b) Using a random number generator for ϵ_n, generate a realization of y_n consisting of 50 samples. To avoid transient effects, be sure to let the filter run for a while. For instance, discard the first 500 or 1000 outputs and keep the last 50.

(c) Compute the sample autocorrelation of y_n based on the above block of data.

(d) Solve the normal equations by means of Levinson's algorithm to determine the Yule-Walker estimates of the model parameters $\{a_1, a_2, a_3, a_4; \sigma_\epsilon^2\}$ and compare them with the exact values.

(e) Compute the corresponding Yule-Walker spectrum and plot it together with the exact autoregressive spectrum versus frequency. Be sure to allow for a sufficiently dense grid of frequencies to be able to resolve the narrow peaks of this example. Plot all spectra in decibels.

(f) Using the same finite block of y_n data, determine estimates of the model parameters $\{a_1, a_2, a_3, a_4; \sigma_\epsilon^2\}$ using Burg's method, and compare them with the Yule-Walker estimates and with the exact values.

(g) Compute the corresponding Burg spectrum and plot it together with the exact spectrum versus frequency.

(h) Using the same block of y_n data, compute the ordinary periodogram spectrum and plot it together with the exact spectrum.

(i) Window the y_n data with a Hamming window and then compute the corresponding periodogram spectrum and plot it together with the exact spectrum.

(j) Repeat parts (b) through (i) using a longer realization of length 100.

(k) Repeat parts (b) through (i) using a length-200 realization of y_n.

(l) Evaluate the various results of this experiment.

14.2 Show that the classical Bartlett spectrum of Eq. (14.2.6) can be written in the compact matrix form of Eq. (14.2.7).

14.3 Show that in the limit of large M, the first sidelobe of the smearing function $W\omega)$ of Eq. (14.2.10) is approximately 13 dB down from the main lobe.

14.4 *Computer Experiment.* (a) Reproduce the spectra shown in Figs. 14.2.1 and 14.2.2.

(b) For the AR case, let $M = 6$, and take the SNRs of both sinusoids to be 6 dB, but change the sinusoid frequencies to

$$\omega_1 = 0.5 + \Delta\omega\,, \quad \omega_2 = 0.5 - \Delta\omega$$

where $\Delta\omega$ is variable. Study the dependence of bias of the spectral peaks on the frequency separation $\Delta\omega$ by computing and plotting the spectra for various values of $\Delta\omega$. (Normalize all spectra to 0 dB at the sinusoid frequency ω_1).

14.5 Derive Equation (14.2.30).

14.6 Let

$$R = \sigma_v^2 I + \sum_{i=1}^{L} P_i \mathbf{s}_{\omega_i}\mathbf{s}_{\omega_i}^\dagger$$

be the autocorrelation matrix of Eq. (14.2.8). Show that the inverse R^{-1} can be computed recursively as follows:

$$R_k^{-1} = R_{k-1}^{-1} - \frac{R_{k-1}^{-1}\mathbf{s}_{\omega_k}\mathbf{s}_{\omega_k}^\dagger R_{k-1}^{-1}}{\mathbf{s}_{\omega_k}^\dagger R_{k-1}^{-1}\mathbf{s}_{\omega_k} + P_k^{-1}}$$

for $k = 1, 2, \dots, L$, initialized by $R_0 = \sigma_v^2 I$.

14.7 Consider the case of one sinusoid ($L = 1$) in noise and arbitrary filter order $M > 2$, so that the $(M+1)\times(M+1)$ autocorrelation matrix is

$$R = \sigma_v^2 I + P_1 \mathbf{s}_{\omega_1}\mathbf{s}_{\omega_1}^\dagger$$

(a) Show that the $(L = 1)$-dimensional signal subspace is spanned by the eigenvector

$$\mathbf{e}_M = \mathbf{s}_{\omega_1}$$

and determine the corresponding eigenvalue.

(b) Show that the $M + 1 - L = M$ dimensional noise subspace is spanned by the M linearly independent eigenvectors, all belonging to the minimum eigenvalue σ_v^2:

$$\mathbf{e}_0 = \begin{bmatrix} 1 \\ -e^{j\omega_1} \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{e}_1 = \begin{bmatrix} 0 \\ 1 \\ -e^{j\omega_1} \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{e}_2 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ -e^{j\omega_1} \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad \dots, \quad \mathbf{e}_{M-1} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 1 \\ -e^{j\omega_1} \end{bmatrix}$$

(c) Show that the eigenpolynomial $A(z)$ corresponding to an arbitrary linear combination of the M noise eigenvectors

$$\mathbf{a} = \mathbf{e}_0 + c_1\mathbf{e}_1 + c_2\mathbf{e}_2 + \cdots + c_{M-1}\mathbf{e}_{M-1}$$

can be factored in the form

$$A(z) = (1 - e^{j\omega_1}z^{-1})(1 + c_1 z^{-1} + c_2 z^{-2} + \cdots + c_{M-1}z^{-(M-1)})$$

exhibiting one zero at the desired sinusoid frequency $e^{j\omega_1}$ on the unit circle, and $M - 1$ additional spurious zeros with arbitrary locations that depend on the particular choice of the coefficients c_i.

14.8 The constraint (14.2.31) can be incorporated into the performance index (14.2.32) by means of a Lagrange multiplier

$$\mathcal{E} = \mathbf{a}^\dagger R\mathbf{a} + \lambda(1 - \mathbf{a}^\dagger\mathbf{a})$$

Show that the minimization of $\mathcal{E}$ is equivalent to the Pisarenko eigenvalue problem of Eq. (14.2.29), with the multiplier λ playing the role of the eigenvalue. Show that the minimum of $\mathcal{E}$ is the minimum eigenvalue.

14.9 Show Eq. (14.3.11).

14.10 Consider a singular $(M+1)\times(M+1)$ autocorrelation matrix R having non-singular principal submatrices, and let $\mathbf{a}$ be the symmetric or antisymmetric order-M prediction filter satisfying $R\mathbf{a} = 0$, as discussed in Sec. 12.5. First, argue that the M zeros of this filter lie on the unit circle $z_i = e^{j\omega_i}$, $i = 1, 2, \dots, M$. Then, consider the eigenvalue decomposition of this matrix in the form $R = E\Lambda E^\dagger$, where Λ is the diagonal matrix of the M nonzero eigenvalues of R and E is the $(M+1)\times M$ matrix whose columns are the M corresponding eigenvectors. Let $S = [\mathbf{s}_{\omega_1}, \mathbf{s}_{\omega_2}, \dots, \mathbf{s}_{\omega_M}]$ be the matrix of phasing vectors defined by the zeros of $\mathbf{a}$. Argue that E is linearly related to S and that R can be written in the form $R = SPS^\dagger$, where P is an $M\times M$ positive-definite matrix. Finally, show that the requirement that R be Toeplitz implies that P must be diagonal, and therefore, R admits the sinusoidal representation

$$R = \sum_{i=1}^{M} P_i \mathbf{s}_{\omega_i}\mathbf{s}_{\omega_i}^\dagger\,, \quad \text{with } P_i > 0$$

14.11 *Computer Experiment.* To simulate Eq. (14.3.7), the amplitudes $A_i(n)$ may be generated by

$$A_i(n) = A_i e^{j\phi_{in}}$$

where ϕ_{in} are independent random phases distributed uniformly over the interval $[0, 2\pi]$, and A_i are deterministic amplitudes related to the assumed signal to noise ratios (SNR) in units of decibels by

$$SNR_i = 10\log_{10}\left[\frac{|A_i|^2}{\sigma_v^2}\right]$$

(a) Consider one plane wave incident on an array of seven sensors from an angle $\theta_1 = 30^{\text{o}}$. The sensors are equally spaced at half-wavelength spacings; i.e., $d = \lambda/2$. For each of the following values of the SNR of the wave

$$SNR = 0\text{ dB},\quad 10\text{ dB},\quad 20\text{ dB}$$

generate $N = 1000$ snapshots of Eq. (14.3.7) and compute the empirical spatial correlation matrix across the array by

$$\hat{R} = \frac{1}{N}\sum_{n=0}^{N-1}\mathbf{y}(n)^*\mathbf{y}(n)^T$$

Compute and plot on the same graph the three spatial spectra: Bartlett, autoregressive (AR), and maximum likelihood (ML), versus wavenumber k.

(b) Repeat for two plane waves incident from angles $\theta_1 = 25^{\text{o}}$ and $\theta_2 = 35^{\text{o}}$, and with equal powers of 30 dB.

(c) Repeat part (b) for angles $\theta_1 = 28^{\text{o}}$ and $\theta_2 = 32^{\text{o}}$.

(d) Repeat part (c) by gradually decreasing the (common) SNR of the two plane waves to the values of 20 dB, 10 dB, and 0 dB.

(e) For parts (a) through (d), also plot all the theoretical spectra.

14.12 Consider L plane waves incident on a linear array of $M+1$ sensors $(L \le M)$ in the presence of spatially coherent noise. As discussed in Sec. 14.3, the corresponding covariance matrix is given by

$$R = \sigma_v^2 Q + \sum_{i=1}^{L} P_i \mathbf{s}_{k_i}\mathbf{s}_{k_i}^\dagger$$

where the waves are assumed to be mutually uncorrelated.

(a) Show that the generalized eigenvalue problem

$$R\mathbf{a} = \lambda Q\mathbf{a}$$

has (1) an $(M+1-L)$-dimensional noise subspace spanned by $M+1-L$ linearly independent degenerate eigenvectors, all belonging to the eigenvalue $\lambda = \sigma_v^2$, and (2) an L-dimensional signal subspace with L eigenvalues greater than σ_v^2.

(b) Show that any two eigenvectors $\mathbf{a}_1$ and $\mathbf{a}_2$ belonging to distinct eigenvalues λ_1 and λ_2 are orthogonal to each other with respect to the inner product defined by the matrix Q, that is, show that $\mathbf{a}_1^\dagger Q\mathbf{a}_2 = 0$.

(c) Show that the L-dimensional signal subspace is spanned by the L vectors

$$Q^{-1}\mathbf{s}_{k_i},\quad i = 1, 2, \dots, L$$

(d) Show that any vector $\mathbf{a}$ in the noise subspace corresponds to a polynomial $A(z)$ that has L of its M zeros on the unit circle at locations

$$z_i = e^{jk_i},\quad i = 1, 2, \dots, L$$

The remaining $M-L$ zeros can have arbitrary locations.

14.13 The previous problem suggests the following approach to the problem of “selectively nulling” some of the sources and not nulling others. Suppose L_1 of the sources are not to be nulled and have known SNRs and directions of arrival, and L_2 of the sources are to be nulled. The total number of sources is then $L = L_1 + L_2$, and assuming incoherent background noise, the incident field will have covariance matrix

$$R = \sigma_v^2 I + \sum_{i=1}^{L_1} P_i\mathbf{s}_{k_i}\mathbf{s}_{k_i}^\dagger + \sum_{i=L_1+1}^{L_1+L_2} P_i\mathbf{s}_{k_i}\mathbf{s}_{k_i}^\dagger$$

Define Q by

$$\sigma_v^2 Q = \sigma_v^2 I + \sum_{i=1}^{L_1} P_i\mathbf{s}_{k_i}\mathbf{s}_{k_i}^\dagger$$

so that we may write R as follows

$$R = \sigma_v^2 Q + \sum_{i=L_1+1}^{L_1+L_2} P_i\mathbf{s}_{k_i}\mathbf{s}_{k_i}^\dagger$$

Then, the nulling of the L_2 sources at wavenumbers k_i, $i = L_1+1, \dots, L_1+L_2$, can be effected by the $(M+1-L_2)$-dimensional noise subspace of the generalized eigenvalue problem

$$R\mathbf{a} = \lambda Q\mathbf{a}$$

having minimum eigenvalue equal to σ_v^2.

(a) As an example, consider the case $M = 2$, $L_1 = L_2 = 1$. Then,

$$R = \sigma_v^2 Q + P_2\mathbf{s}_{k_2}\mathbf{s}_{k_2}^\dagger,\quad \sigma_v^2 Q = \sigma_v^2 I + P_1\mathbf{s}_{k_1}\mathbf{s}_{k_1}^\dagger$$

Show that the $(M+1-L_2 = 2)$-dimensional noise subspace is spanned by the two eigenvectors

$$\mathbf{e}_1 = \begin{bmatrix} 1 \\ -e^{jk_2} \\ 0 \end{bmatrix},\quad \mathbf{e}_2 = \begin{bmatrix} 0 \\ 1 \\ -e^{jk_2} \end{bmatrix}$$

(b) Show that an arbitrary linear combination

$$\mathbf{a} = \mathbf{e}_1 + \rho\mathbf{e}_2$$

corresponds to a filter $A(z)$ having one zero at the desired location $z_2 = e^{jk_2}$, and a spurious zero with arbitrary location.

(c) Show that the $(L_2 = 1)$-dimensional signal subspace is spanned by the vector

$$\mathbf{e}_3 = Q^{-1}\mathbf{s}_{k_2}$$

and that the corresponding generalized eigenvalue is

$$\lambda = \sigma_v^2 + P_2\mathbf{s}_{k_2}^\dagger Q^{-1}\mathbf{s}_{k_2}$$

(d) Verify the orthogonality properties $\mathbf{e}_i^\dagger Q\mathbf{e}_3 = 0$, $i = 1,2$, for the three eigenvectors $\mathbf{e}_1,\mathbf{e}_2,\mathbf{e}_3$ defined in parts (a) and (c).

(e) As another example, consider the case $M = 3$ and $L_1 = L_2 = 1$. Show that the $(M+1-L_2 = 3)$-dimensional noise subspace is spanned by the three eigenvectors

$$\mathbf{e}_1 = \begin{bmatrix} 1 \\ -e^{jk_2} \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{e}_2 = \begin{bmatrix} 0 \\ 1 \\ -e^{jk_2} \\ 0 \end{bmatrix}, \quad \mathbf{e}_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ -e^{jk_2} \end{bmatrix}$$

and the signal eigenvector is $\mathbf{e}_4 = Q^{-1}\mathbf{s}_{k_2}$. Generalize this part and part (a), to the case of arbitrary M and $L_1 = L_2 = 1$.

(f) As a final example that corresponds to a unique noise eigenvector, consider the case $M = 2$, $L_1 = 1$, and $L_2 = 2$, so that

$$R = \sigma_v^2 Q + P_2\mathbf{s}_{k_2}\mathbf{s}_{k_2}^\dagger + P_3\mathbf{s}_{k_3}\mathbf{s}_{k_3}^\dagger\,, \quad \sigma_v^2 Q = \sigma_v^2 I + P_1\mathbf{s}_{k_1}\mathbf{s}_{k_1}^\dagger$$

with k_2 and k_3 to be nulled. Show that the $(M+1-L_2 = 1)$-dimensional noise subspace is spanned by

$$\mathbf{a} = \mathbf{e}_1 = \begin{bmatrix} 1 \\ -(e^{jk_2} + e^{jk_3}) \\ e^{jk_2}e^{jk_3} \end{bmatrix}$$

and that the corresponding polynomial $A(z)$ factors into the two desired zeros

$$A(z) = (1 - e^{jk_2}z^{-1})(1 - e^{jk_3}z^{-1})$$

14.14 *Computer Experiment.* Consider a nine-element $(M = 8)$ linear array with half-wavelength spacing and two mutually uncorrelated incident plane waves with wavenumbers $k_1 = 0.3\pi$, $k_2 = 0.5\pi$ and equal powers of 20 dB. The background noise is incoherent with variance $\sigma_v^2 = 1$.

(a) Construct the theoretical matrix R of Eq. (14.3.13) and solve its eigenproblem determining the nine eigenvectors and eigenvalues. Using a root finder (see e.g., [1206]), compute the eight zeros of each of the seven noise subspace eigenvectors and verify that the desired zeros lie on the unit circle.

(b) Generate $N = 100$ snapshots, construct the sample covariance matrix R of Eq. (14.4.14), solve its eigenproblem, use the AIC and MDL criteria to check the dimension of the noise subspace, but regardless of these criteria take that dimension to be seven. Compare the empirical eigenvalues with the theoretical ones found above. Compute the zeros of the noise subspace eigenvectors and decide if the desired zeros are among them and if any spurious ones lie close to the unit circle. Also, compute the zeros of the Min-Norm vector $\mathbf{d}$.

(c) On the same graph, plot in dB the pseudospectra of a few of the noise subspace eigenvectors, say, the first three. On a separate graph, but using the same vertical scales as the previous one, plot the MUSIC and Min-Norm spectra.

(d) Using the same set of snapshots, repeat parts (b,c) for the symmetrized sample covariance matrix of Eq. (14.4.15).

(e) For fixed SNR, repeat parts (b,c,d) for the following choices of number of snapshots: $N = 20, 50, 150, 200, 500$.

(f) With the number of snapshots fixed at $N = 100$, repeat parts (a,b,c,d) for the following values of the signal to noise ratio: $SNR = -10, -5, 0, 5, 10, 30$ dB.

(g) Repeat parts (a–f) for three 20-dB plane waves with $k_1 = 0.3\pi$, $k_2 = 0.4\pi$, $k_3 = 0.5\pi$.

14.15 Show Eqs. (14.11.9) and (14.11.10).

14.16 Consider an M-dimensional complex random vector $\mathbf{y}$ with real and imaginary parts $\boldsymbol{\xi}$ and $\boldsymbol{\eta}$, so that $\mathbf{y} = \boldsymbol{\xi} + j\boldsymbol{\eta}$. With the complex vector $\mathbf{y}$ we associate a $(2M)$-dimensional real random vector $\bar{\mathbf{y}} = \begin{bmatrix} \boldsymbol{\xi} \\ \boldsymbol{\eta} \end{bmatrix}$. The corresponding covariance matrices are defined by

$$R = E[\mathbf{y}^*\mathbf{y}^T]\,, \quad \bar{R} = E[\bar{\mathbf{y}}\bar{\mathbf{y}}^T]$$

(a) Show that the conditions $E[\boldsymbol{\xi}\boldsymbol{\xi}^T] = E[\boldsymbol{\eta}\boldsymbol{\eta}^T]$ and $E[\boldsymbol{\xi}\boldsymbol{\eta}^T] = -E[\boldsymbol{\eta}\boldsymbol{\xi}^T]$ are equivalent to the condition $E[\mathbf{y}\mathbf{y}^T] = 0$, and that in this case the covariance matrices can be written as follows:

$$R = 2(A + jB)\,, \quad \bar{R} = \begin{bmatrix} A & B \\ -B & A \end{bmatrix}, \quad A = E[\boldsymbol{\xi}\boldsymbol{\xi}^T]\,, \quad B = E[\boldsymbol{\xi}\boldsymbol{\eta}^T]$$

The matrix A is symmetric and B antisymmetric. Show the equality of the quadratic forms

$$\mathbf{y}^T R^{-1}\mathbf{y}^* = \frac{1}{2}\bar{\mathbf{y}}^T\bar{R}^{-1}\bar{\mathbf{y}}$$

Also, show the relationship between the determinants $\det R = 2^M(\det\bar{R})^{1/2}$.
Hint: Apply a correlation canceling transformation on $\bar{R}$ and use the matrix identity $A + BA^{-1}B = (A + jB)A^{-1}(A - jB)$.

(b) A complex gaussian random vector $\mathbf{y}$ is defined by the requirement that the corresponding real vector $\bar{\mathbf{y}}$ be gaussian [1207,1208]. Equating the elemental probabilities $p(\mathbf{y})d^{2M}\mathbf{y} = p(\bar{\mathbf{y}})d^{2M}\bar{\mathbf{y}}$ and using the results of part (a), show that if $p(\bar{\mathbf{y}})$ is an ordinary (zero-mean) gaussian with covariance $\bar{R}$, then the density of $\mathbf{y}$ is

$$p(\bar{\mathbf{y}}) = \frac{1}{(2\pi)^M(\det\bar{R})^{1/2}}\exp\left(-\frac{1}{2}\bar{\mathbf{y}}^T\bar{R}^{-1}\bar{\mathbf{y}}\right) \quad \Rightarrow \quad p(\mathbf{y}) = \frac{1}{\pi^M\det R}\exp\left(-\mathbf{y}^T R^{-1}\mathbf{y}^*\right)$$

(c) Using this density show for any four components of $\mathbf{y}$

$$E[y_i^* y_j y_k^* y_l] = R_{ij}R_{kl} + R_{il}R_{kj}$$

(d) Use this result to prove Eq. (14.11.12)

14.17 Show that the log-likelihood function based on N independent complex gaussian snapshots is given by (up to a constant)

$$\ln p = -N\,\mathrm{tr}[\ln R + R^{-1}\hat{R}]$$

where $\hat{R}$ is given byEq. (14.4.14). Note that it differs by a factor of two from the real-valued case. From the discussion of Sec. 1.18, it follows that $\hat{R}$ is the maximum likelihood estimate of R. Moreover, the trace formula for the Fisher information matrix also differs by a factor of two, namely,

$$J_{ij} = N\,\mathrm{tr}\left[R^{-1}\frac{\partial R}{\partial\lambda_i}R^{-1}\frac{\partial R}{\partial\lambda_j}\right]$$

14.18 Using Eq. (14.11.12), show that the covariances of the LP parameters E and $\mathbf{a}$ are in the complex-valued case:

$$E\big[(\Delta E)^2\big] = \frac{E^2}{N}\,, \quad E[\Delta\mathbf{a}\,\Delta E] = 0\,, \quad E[\Delta\mathbf{a}\,\Delta\mathbf{a}^\dagger] = \frac{E}{N}(R^{-1} - E^{-1}\mathbf{a}\,\mathbf{a}^\dagger)$$

14.19 Let $S(k)= \mathbf{s}_k^\dagger R\mathbf{s}_k$ be the Bartlett spectrum. Using Eq. (14.11.13), show that its variance is

$$E\big[\big(\Delta S(k)\big)^2\big] = \frac{1}{N}S(k)^2$$

Show that the variance of the ML spectrum $S(k)= 1/\mathbf{s}_k^\dagger R^{-1}\mathbf{s}_k$ is also given by a similar formula.

14.20 (a) Let $A(k)= \mathbf{s}_k^\dagger\mathbf{a}$ be the frequency response of the LP polynomial in the complex-valued case. Using the results of Problem 14.18, show that its variance is

$$E\big[|\Delta A(k)|^2\big] = \frac{E}{N}\big[\mathbf{s}_k^\dagger R^{-1}\mathbf{s}_k - E^{-1}|A(k)|^2\big]$$

Use the kernel representation of Problem 12.17 to argue that the right-hand side is positive. Alternatively, show that it is positive by writing $A(k)= E(\mathbf{s}_k^\dagger R^{-1}\mathbf{u}_0)$ and $E = (\mathbf{u}_0^\dagger R^{-1}\mathbf{u}_0)^{-1}$, and using the Schwarz inequality.

(b) In the complex case, show that $E[\Delta\mathbf{a}\,\Delta\mathbf{a}^T]= 0$. Then, show that the variance of the AR spectrum $S(k)= E/|A(k)|^2$ is given by

$$E\big[\big(\Delta S(k)\big)^2\big] = \frac{1}{N}S(k)^2\big[2S(k)\,(\mathbf{s}_k^\dagger R^{-1}\mathbf{s}_k)-1\big]$$

and show again that the right-hand side is positive.

15

SVD and Signal Processing

15.1 Vector and Matrix Norms

The three most widely used vector norms [1234,1235] are the L_2 or Euclidean norm, the L_1 and the L_∞ norms, defined for a vector $\mathbf{x}\in\mathbb{R}^N$ by:

$$\begin{aligned} \|\mathbf{x}\|_2 &= \sqrt{|x_1|^2 + |x_2|^2 + \cdots + |x_N|^2} = \sqrt{\mathbf{x}^T\mathbf{x}} \\ \|\mathbf{x}\|_1 &= |x_1| + |x_2| + \cdots + |x_N| \\ \|\mathbf{x}\|_\infty &= \max\big(|x_1|, |x_2|, \dots, |x_N|\big) \end{aligned} \qquad \text{where} \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} \tag{15.1.1}$$

All vector norms satisfy the *triangle inequality*:

$$\|\mathbf{x}+\mathbf{y}\| \le \|\mathbf{x}\| + \|\mathbf{y}\|\,, \quad \text{for} \quad \mathbf{x},\mathbf{y}\in\mathbb{R}^N \tag{15.1.2}$$

Unless otherwise specified, from now on the notation $\|\mathbf{x}\|$ will denote the Euclidean norm. The *Cauchy-Schwarz inequality* for the Euclidean norm reads:

$$\big|\mathbf{x}^T\mathbf{y}\big| \le \|\mathbf{x}\|\,\|\mathbf{y}\| \tag{15.1.3}$$

where equality is achieved when $\mathbf{y}$ is any scalar multiple of $\mathbf{x}$, that is, $\mathbf{y} = c\mathbf{x}$. The “angle” between the two vectors $\mathbf{x},\mathbf{y}$ is defined through:

$$\cos\theta = \frac{\mathbf{x}^T\mathbf{y}}{\|\mathbf{x}\|\,\|\mathbf{y}\|} \tag{15.1.4}$$

An $N{\times}M$ matrix A is a linear mapping from $\mathbb{R}^M$ to $\mathbb{R}^N$, that is, for each $\mathbf{x}\in\mathbb{R}^M$, the vector $\mathbf{y} = A\mathbf{x}$ is in $\mathbb{R}^N$. For each vector norm, one can define a corresponding *matrix norm* through the definition:

$$\|A\| = \sup_{\|\mathbf{x}\|\neq 0}\frac{\|A\mathbf{x}\|}{\|\mathbf{x}\|} = \sup_{\|\mathbf{x}\|=1}\|A\mathbf{x}\| \tag{15.1.5}$$

We will see later that the Euclidean matrix norm $\|A\|_2$ is equal to the *largest singular value* of the SVD decomposition of A, or equivalently, the square-root of the largest *eigenvalue* of the matrix A^TA or the matrix AA^T. The L_1 and L_∞ matrix norms can be expressed directly in terms of the matrix elements A_{ij} of A:

$$\begin{aligned} \|A\|_1 &= \max_j \sum_i |A_{ij}| = \text{maximum of column-wise sums} \\ \|A\|_\infty &= \max_i \sum_j |A_{ij}| = \text{maximum of row-wise sums} \end{aligned} \tag{15.1.6}$$

Another useful matrix norm—not derivable from a vector norm—is the *Frobenius* norm defined to be the sum of the squares of all the matrix elements:

$$\|A\|_F = \sqrt{\sum_{i,j} |A_{ij}|^2} = \sqrt{\operatorname{tr}(A^TA)} \qquad \text{(Frobenius norm)} \tag{15.1.7}$$

The L_2, L_1, L_∞, and the Frobenius matrix norms satisfy the matrix versions of the triangle and Cauchy-Schwarz inequalities:

$$\begin{aligned} \|A+B\| &\le \|A\| + \|B\| \\ \|AB\| &\le \|A\|\|B\| \end{aligned} \tag{15.1.8}$$

The *distance* between two vectors, or between two matrices, may be defined with respect to any norm:

$$d(\mathbf{x},\mathbf{y}) = \|\mathbf{x}-\mathbf{y}\|\,, \quad d(A,B) = \|A-B\| \tag{15.1.9}$$

15.2 Subspaces, Bases, and Projections

A subset $Y \subseteq \mathbb{R}^N$ is a linear *subspace* if every linear combination of vectors from Y also lies in Y. The dimension of the subspace Y is the *maximum number* of linearly independent vectors in Y.

If the dimension of Y is M, then, any set of M linearly independent vectors, say $\{\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_M\}$, forms a *basis* for Y. Each basis vector $\mathbf{b}_i$ is an N-dimensional vector, that is, it lies in $\mathbb{R}^N$. Because Y is a subset of $\mathbb{R}^N$, we must necessarily have $M \le N$. Any vector in Y can be expanded *uniquely* as a linear combination of the basis vectors, that is, for $\mathbf{b} \in Y$:

$$\mathbf{b} = \sum_{i=1}^{M} c_i \mathbf{b}_i = c_1\mathbf{b}_1 + c_2\mathbf{b}_2 + \cdots c_M\mathbf{b}_M = [\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_M]\begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_M \end{bmatrix} = B\mathbf{c} \tag{15.2.1}$$

where we defined the $N\times M$ basis matrix $B = [\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_M]$ and the $M\times 1$ vector of expansion coefficients $\mathbf{c} = [c_1, c_2 \dots, c_M]^T$.

Because the columns of B are linearly independent, B will have *full rank* equal to M. It follows that the $M\times M$ matrix B^TB will also have full rank[†] and, therefore, it will be invertible. This allows us to compute the expansion coefficients $\mathbf{c}$. Multiplying both sides of (15.2.1) by B^T, we may solve for $\mathbf{c}$:

$$B^T\mathbf{b} = B^TB\mathbf{c} \quad \Rightarrow \quad \mathbf{c} = (B^TB)^{-1}B^T\mathbf{b} = B^+\mathbf{b}\,, \qquad B^+ \equiv (B^TB)^{-1}B^T \tag{15.2.2}$$

The space spanned by the linear combinations of the columns of the matrix B is called the *column space* or *range space* of B and is denoted by $R(B)$. Because B is a basis for Y, we will have $Y = R(B)$. The matrix equation $B\mathbf{c} = \mathbf{b}$ given in (15.2.1) is an overdetermined system of N equations in M unknowns that has a solution because we assumed that $\mathbf{b}$ lies in the range space of B.

The quantity $B^+ = (B^TB)^{-1}B^T$ is a special case of the Moore-Penrose *pseudoinverse* (for the case of a full rank matrix B with $N \ge M$.) In MATLAB notation, the solution (15.2.2) is obtained via the *backslash* or the *pseudoinverse* operators (which produce the same answer in the full-rank case):

$$\mathbf{c} = B\backslash\mathbf{b} = \operatorname{pinv}(B)*\mathbf{b} = B^+\mathbf{b} \tag{15.2.3}$$

The matrix $B^TB \in \mathbb{R}^{M\times M}$ is called the Grammian. Its matrix elements are the mutual dot products of the basis vectors $(B^TB)_{ij} = \mathbf{b}_i^T\mathbf{b}_j$, $i,j = 1,2,\dots,M$.

The quantity $\mathcal{P} = BB^+ = B(B^TB)^{-1}B^T$ is the *projection* matrix onto the subspace Y. As a projection matrix, it is idempotent and symmetric, that is, $\mathcal{P}^2 = \mathcal{P}$ and $\mathcal{P}^T = \mathcal{P}$. The matrix $\mathcal{Q} = I_N - \mathcal{P}$ is also a projection matrix, projecting onto the *orthogonal complement* of Y, that is, the space $Y^\perp$ of vectors in $\mathbb{R}^N$ that are orthogonal to each vector in Y. Thus, we have:

$$\begin{aligned} \mathcal{P} &= BB^+ = B(B^TB)^{-1}B^T = \text{projector onto } Y \\ \mathcal{Q} &= I_N - BB^+ = I_N - B(B^TB)^{-1}B^T = \text{projector onto } Y^\perp \end{aligned} \tag{15.2.4}$$

They satisfy the properties $B^T\mathcal{Q} = 0$, $\mathcal{P}\mathcal{Q} = \mathcal{Q}\mathcal{P} = 0$, and $\mathcal{P} + \mathcal{Q} = I_N$. These imply that the full space $\mathbb{R}^N$ is the direct sum of Y and $Y^\perp$. Moreover, the subspace $Y^\perp$ is the same as the *null* space $N(B^T)$ of B^T. This follows from the property that $\mathbf{b}_\perp \in Y^\perp$ if and only if $B^T\mathbf{b}_\perp = 0$. Thus, we have the decomposition:

$$Y \oplus Y^\perp = R(B) \oplus N(B^T) = \mathbb{R}^N \tag{15.2.5}$$

The *orthogonal decomposition theorem* follows from (15.2.5). It states that a given vector in $\mathbb{R}^N$ can be decomposed uniquely with respect to a subspace Y into the sum of a vector that lies in Y and a vector that lies in $Y^\perp$, that is, for $\mathbf{b} \in \mathbb{R}^N$:

$$\mathbf{b} = \mathbf{b}_\parallel + \mathbf{b}_\perp\,, \quad \text{where} \quad \mathbf{b}_\parallel \in Y\,, \quad \mathbf{b}_\perp \in Y^\perp \tag{15.2.6}$$

so that $\mathbf{b}_\perp^T\mathbf{b}_\parallel = 0$. The proof is trivial; defining $\mathbf{b}_\parallel = \mathcal{P}\mathbf{b}$ and $\mathbf{b}_\perp = \mathcal{Q}\mathbf{b}$, we have:

$$\mathbf{b} = I_N\mathbf{b} = (\mathcal{P} + \mathcal{Q})\mathbf{b} = \mathcal{P}\mathbf{b} + \mathcal{Q}\mathbf{b} = \mathbf{b}_\parallel + \mathbf{b}_\perp$$

The uniqueness is argued as follows: setting $\mathbf{b}_\parallel + \mathbf{b}_\perp = \mathbf{b}'_\parallel + \mathbf{b}'_\perp$ for a different pair $\mathbf{b}'_\parallel \in Y$, $\mathbf{b}'_\perp \in Y^\perp$, we have $\mathbf{b}_\parallel - \mathbf{b}'_\parallel = \mathbf{b}'_\perp - \mathbf{b}_\perp$, which implies that both difference vectors lie in $Y \cap Y^\perp = \{0\}$, and therefore, they must be the zero vector.

[†] Indeed, $B^TB\mathbf{c} = 0 \;\Rightarrow\; \mathbf{c}^TB^TB\mathbf{c} = \|B\mathbf{c}\|^2 = 0 \;\Rightarrow\; B\mathbf{c} = 0 \;\Rightarrow\; \mathbf{c} = 0$, because B has full rank.

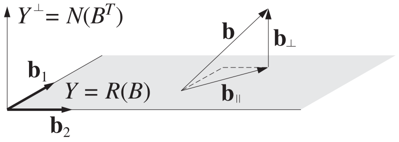


Fig. 15.2.1 Projection of $\mathbf{b}$ onto the subspace $Y = R(B)$ spanned by $B = [\mathbf{b}_1, \mathbf{b}_2]$.

Fig. 15.2.1 illustrates this theorem. An alternative proof is to expand $\mathbf{b}_{\parallel}$ in the B-basis, that is, $\mathbf{b}_{\parallel} = B\mathbf{c}$, and require that $\mathbf{b}_{\perp} = \mathbf{b} - \mathbf{b}_{\parallel}$ be perpendicular to Y, that is, $B^T\mathbf{b}_{\perp} = 0$. Thus, we get the conditions:

$$\mathbf{b} = B\mathbf{c} + \mathbf{b}_{\perp} \quad \Rightarrow \quad B^T\mathbf{b} = B^TB\mathbf{c} + B^T\mathbf{b}_{\perp} = B^TB\mathbf{c}\,, \quad \text{or,}$$

$$\mathbf{c} = (B^TB)^{-1}B^T\mathbf{b}\,, \qquad \mathbf{b}_{\parallel} = B\mathbf{c} = B(B^TB)^{-1}B^T\mathbf{b} = \mathcal{P}\mathbf{b} \tag{15.2.7}$$

A variation of the orthogonal decomposition theorem is the *orthogonal projection theorem*, which states that the projection $\mathbf{b}_{\parallel}$ is that vector in Y that lies closest to $\mathbf{b}$ with respect to the Euclidean distance, that is, as the vector $\mathbf{y} \in Y$ varies over Y, the distance $\|\mathbf{b} - \mathbf{y}\|$ is minimized when $\mathbf{y} = \mathbf{b}_{\parallel}$.

Fig. 15.2.2 illustrates the theorem. The proof is straightforward. We have $\mathbf{b} - \mathbf{y} = \mathbf{b}_{\parallel} + \mathbf{b}_{\perp} - \mathbf{y} = (\mathbf{b}_{\parallel} - \mathbf{y}) + \mathbf{b}_{\perp}$, but since both $\mathbf{b}_{\parallel}$ and $\mathbf{y}$ lie in Y, so does $(\mathbf{b}_{\parallel} - \mathbf{y})$ and therefore, $(\mathbf{b}_{\parallel} - \mathbf{y}) \perp \mathbf{b}_{\perp}$. It follows from the Pythagorean theorem that:

$$\|\mathbf{b} - \mathbf{y}\|^2 = \|(\mathbf{b}_{\parallel} - \mathbf{y}) + \mathbf{b}_{\perp}\|^2 = \|\mathbf{b}_{\parallel} - \mathbf{y}\|^2 + \|\mathbf{b}_{\perp}\|^2$$

which is minimized when $\mathbf{y} = \mathbf{b}_{\parallel}$. The minimized value of the distance is $\|\mathbf{b} - \mathbf{b}_{\parallel}\| = \|\mathbf{b}_{\perp}\|$. The orthogonal projection theorem provides an intuitive interpretation of linear estimation problems and of least-squares solutions of linear equations.

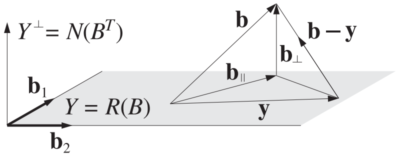


Fig. 15.2.2 The projection $\mathbf{b}_{\parallel}$ minimizes the distance $\|\mathbf{b} - \mathbf{y}\|$ to the subspace Y.

The basis B for the subspace Y is not unique. Any other set of M linearly independent vectors in Y would do. The projector $\mathcal{P}$ remains *invariant* under a change of basis. Indeed, suppose that another basis is defined by the basis matrix $U = [\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_M]$ whose M columns $\mathbf{u}_i$ are assumed to be linearly independent. Then, each $\mathbf{b}_j$ can be expanded as a linear combination of the new basis vectors $\mathbf{u}_i$:

$$\mathbf{b}_j = \sum_{i=1}^{M} \mathbf{u}_i c_{ij}\,, \quad j = 1, 2, \dots, M \tag{15.2.8}$$

These relationships may be expressed compactly in the matrix form:

$$B = UC \qquad \text{(base change)} \tag{15.2.9}$$

where C is the $M \times M$ matrix of expansion coefficients c_{ij}. Because U and B have full rank, the matrix C will be invertible (the $\mathbf{u}_i$'s can just as well be expressed in terms of the $\mathbf{b}_j$'s.) It follows that $B^TB = C^T(U^TU)C$ and:

$$\begin{aligned}\mathcal{P} = B(B^TB)^{-1}B^T &= UC\big(C^T(U^TU)C\big)^{-1}C^TU^T \\ &= UC\big(C^{-1}(U^TU)^{-1}C^{-T}\big)C^TU^T = U(U^TU)^{-1}U^T\end{aligned}$$

where C^{-T} denotes the inverse of the transposed matrix C^T. Among the possible bases for Y, a convenient one is to choose the M vectors $\mathbf{u}_i$ to have unit norm and be mutually orthogonal, that is, $\mathbf{u}_i^T\mathbf{u}_j = \delta_{ij}$, for $i, j = 1, 2, \dots, M$. Compactly, we may express this condition in terms of the basis matrix $U = [\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_M]$:

$$U^TU = I_M \qquad \text{(orthonormal basis)} \tag{15.2.10}$$

When U is orthonormal, the projection matrix $\mathcal{P}$ can be expressed simply as:

$$\mathcal{P} = B(B^TB)^{-1}B^T = U(U^TU)^{-1}U^T = UU^T \tag{15.2.11}$$

There are many ways to construct the orthonormal basis U starting with B. One is through the SVD implemented into the function **orth**. Another is through the QR-factorization, which is equivalent to the Gram-Schmidt orthogonalization process. The two alternatives are:

```
U = orth(B);          % SVD-based
U = qr(B,0);          % QR-factorization
```

Example 15.2.1: A three-dimensional subspace Y of $\mathbb{R}^4$ is spanned by the basis matrix B:

$$B = \begin{bmatrix} 1.52 & 2.11 & 4.30 \\ -1.60 & -2.05 & -4.30 \\ 2.08 & 2.69 & 3.70 \\ -2.00 & -2.75 & -3.70 \end{bmatrix}$$

The matrix B has rank 3, but non-orthogonal columns. The two orthogonal bases obtained via the SVD and via the QR factorization are as follows:

$$B = \begin{bmatrix} -0.5 & 0.5 & 0.5 \\ 0.5 & -0.5 & 0.5 \\ -0.5 & -0.5 & -0.5 \\ 0.5 & 0.5 & -0.5 \end{bmatrix} \begin{bmatrix} -3.60 & -4.80 & -8.00 \\ -0.48 & -0.64 & 0.60 \\ -0.08 & 0.06 & 0.00 \end{bmatrix} = U_1C_1$$

$$B = \begin{bmatrix} -0.4184 & -0.5093 & 0.5617 \\ 0.4404 & -0.4904 & -0.5617 \\ -0.5726 & 0.4875 & -0.4295 \\ 0.5505 & 0.5122 & 0.4295 \end{bmatrix} \begin{bmatrix} -3.6327 & -4.8400 & -7.8486 \\ 0.0000 & -0.1666 & -0.1729 \\ 0.0000 & 0.0000 & 1.6520 \end{bmatrix} = U_2C_2$$

The bases were constructed by the MATLAB commands:

```
[U1,S1,V1] = svd(B,0);  C1 = S1*V1';       % alternatively, U1 = orth(B);
[U2,C2] = qr(B,0);
```

The orthogonal bases satisfy $U_1^TU_1 = U_2^TU_2 = I_3$, and C_2 is upper triangular. The projection matrices onto Y and $Y^\perp$ are:

$$\mathcal{P} = U_1U_1^T = \frac{1}{4}\begin{bmatrix} 3 & -1 & -1 & -1 \\ -1 & 3 & -1 & -1 \\ -1 & -1 & 3 & -1 \\ -1 & -1 & -1 & 3 \end{bmatrix}, \quad \mathcal{Q} = I_4 - \mathcal{P} = \frac{1}{4}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

The ranks of $\mathcal{P}, \mathcal{Q}$ are the dimensions of the subspaces $Y, Y^\perp$, that is, 3 and 1. □

15.3 The Fundamental Theorem of Linear Algebra

An $N\times M$ matrix $A \in \mathbb{R}^{N\times M}$ of rank $r \le \min\{M, N\}$ is characterized by four fundamental subspaces: the two range subspaces $R(A)$ and $R(A^T)$ and the two null subspaces $N(A)$ and $N(A^T)$. These subspaces play a fundamental role in the SVD of A and in the least-squares solution of the equation $A\mathbf{x} = \mathbf{b}$.

The *fundamental theorem of linear algebra* [1234,1254] states that their dimensions and orthogonality properties are as follows:

$$\begin{array}{llll} R(A), & \text{subspace of } \mathbb{R}^N, & \dim = r, & R(A)^\perp = N(A^T) \\ N(A^T), & \text{subspace of } \mathbb{R}^N, & \dim = N - r, & N(A^T)^\perp = R(A) \\ R(A^T), & \text{subspace of } \mathbb{R}^M, & \dim = r, & R(A^T)^\perp = N(A) \\ N(A), & \text{subspace of } \mathbb{R}^M, & \dim = M - r, & N(A)^\perp = R(A^T) \end{array} \qquad (15.3.1)$$

The dimensions of the two range subspaces are equal to the rank of A. The dimensions of the null subspaces are called the *nullity* of A and A^T. It follows that the spaces $\mathbb{R}^M$ and $\mathbb{R}^N$ are the direct sums:

$$\begin{aligned} \mathbb{R}^N &= R(A) \oplus N(A^T) = R(A) \oplus R(A)^\perp \\ \mathbb{R}^M &= R(A^T) \oplus N(A) = R(A^T) \oplus R(A^T)^\perp \end{aligned} \qquad (15.3.2)$$

Their intersections are: $R(A)\cap N(A^T) = \{0\}$ and $R(A^T)\cap N(A) = \{0\}$, that is, the zero vector. Fig. 15.3.1 depicts these subspaces and the action of the matrices A and A^T. The fundamental theorem of linear algebra, moreover, states that the singular value decomposition of A provides *orthonormal bases* for these four subspaces and that A and A^T become diagonal with respect to these bases.

15.4 Solving Linear Equations

Given an $N\times M$ matrix $A \in \mathbb{R}^{N\times M}$ of rank $r \le \min(N, M)$ and a vector $\mathbf{b} \in \mathbb{R}^N$, the linear system $A\mathbf{x} = \mathbf{b}$ may or may not have a solution $\mathbf{x} \in \mathbb{R}^M$. A solution exists only if the vector $\mathbf{b}$ lies in the range space $R(A)$ of the matrix A.

However, there is always a solution in the *least-squares* sense. That solution may not be unique. The properties of the four fundamental subspaces of A determine the nature of the least-squares solutions [1234].

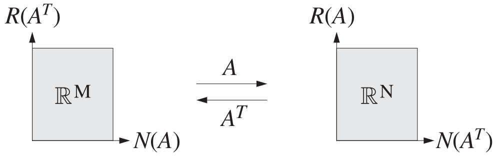


Fig. 15.3.1 The four fundamental subspaces associated with an $N\times M$ matrix A.

Defining the error vector $\mathbf{e} = \mathbf{b} - A\mathbf{x}$, a least-squares solution is a vector $\mathbf{x} \in \mathbb{R}^M$ that minimizes the Euclidean norm $\|\mathbf{e}\|$, that is,

$$\mathcal{J} = \|\mathbf{e}\|^2 = \mathbf{e}^T\mathbf{e} = \|\mathbf{b} - A\mathbf{x}\|^2 = (\mathbf{b} - A\mathbf{x})^T(\mathbf{b} - A\mathbf{x}) = \min \qquad (15.4.1)$$

The solution is obtained by setting the gradient of the performance index to zero:

$$\frac{\partial \mathcal{J}}{\partial \mathbf{x}} = -2A^T\mathbf{e} = -2A^T(\mathbf{b} - A\mathbf{x}) = 0$$

Thus, we obtain the *orthogonality* and *normal* equations:

$$\begin{array}{ll} A^T\mathbf{e} = 0 & \text{(orthogonality equations)} \\ A^TA\mathbf{x} = A^T\mathbf{b} & \text{(normal equations)} \end{array} \qquad (15.4.2)$$

If the $M\times M$ matrix A^TA has *full rank*, then it is invertible and the solution of the normal equations is unique and is given by

$$\mathbf{x} = (A^TA)^{-1}A^T\mathbf{b} \qquad \text{(full-rank overdetermined case)} \qquad (15.4.3)$$

This happens, for example, if $N \ge M$ and $r = M$. In the special case of a *square full-rank* matrix A (that is, $r = N = M$), this solution reduces to $\mathbf{x} = A^{-1}\mathbf{b}$.

For the rank-defective case, A^TA is not invertible, but Eq. (15.4.2) does have solutions. They can be characterized with the help of the four fundamental subspaces of A, as shown in Fig. 15.4.1.

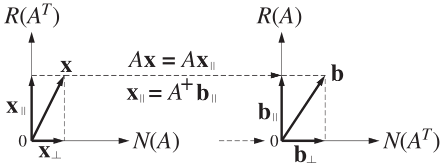


Fig. 15.4.1 Role of the fundamental subspaces in the least-squares solution of $A\mathbf{x} = \mathbf{b}$.

Using the direct-sum decompositions (15.3.2), we resolve both $\mathbf{b}$ and $\mathbf{x}$ into their unique orthogonal components:

$$\begin{array}{llll} \mathbf{b} = \mathbf{b}_\parallel + \mathbf{b}_\perp, & \mathbf{b}_\parallel \in R(A), & \mathbf{b}_\perp \in N(A^T), & \mathbf{b} \in \mathbb{R}^N \\ \mathbf{x} = \mathbf{x}_\parallel + \mathbf{x}_\perp, & \mathbf{x}_\parallel \in R(A^T), & \mathbf{x}_\perp \in N(A), & \mathbf{x} \in \mathbb{R}^M \end{array} \qquad (15.4.4)$$

Because $\mathbf{x}_\perp$ lies in the null space of A, we have $A\mathbf{x}_\perp = 0$, and therefore, $A\mathbf{x} = A(\mathbf{x}_\| + \mathbf{x}_\perp) = A\mathbf{x}_\|$. Then, the error vector becomes:

$$\mathbf{e} = \mathbf{b} - A\mathbf{x} = \big(\mathbf{b}_\| - A\mathbf{x}_\|\big) + \mathbf{b}_\perp \equiv \mathbf{e}_\| + \mathbf{e}_\perp \tag{15.4.5}$$

Because both $\mathbf{b}_\|$ and $A\mathbf{x}_\|$ lie in $R(A)$, so does $\mathbf{e}_\| = \mathbf{b}_\| - A\mathbf{x}_\|$, and therefore, it will be orthogonal to $\mathbf{e}_\perp = \mathbf{b}_\perp$. Thus, Eq. (15.4.5) represents the orthogonal decomposition of the error vector $\mathbf{e}$. But from the orthogonality equations (15.4.2), we have $A^T\mathbf{e} = 0$, which means that $\mathbf{e} \in N(A^T)$, and therefore, $\mathbf{e} = \mathbf{e}_\perp$. This requires that $\mathbf{e}_\| = 0$, or, $A\mathbf{x}_\| = \mathbf{b}_\|$. Because $\mathbf{b}_\|$ lies in $R(A)$, this system will have a solution $\mathbf{x}_\|$.

Moreover, because $\mathbf{x}_\| \in R(A^T)$, this solution will be unique. Indeed, if $\mathbf{b}_\| = A\mathbf{x}_\| = A\mathbf{x}'_\|$, for another vector $\mathbf{x}'_\| \in R(A^T)$, then $A(\mathbf{x}_\| - \mathbf{x}'_\|) = 0$, or, $\mathbf{x}_\| - \mathbf{x}'_\|$ would lie in $N(A)$ in addition to lying in $R(A^T)$, and hence it must be the zero vector because $R(A^T)\cap N(A) = \{0\}$. In fact, this unique $\mathbf{x}_\|$ may be constructed by the pseudoinverse of A:

$$A\mathbf{x}_\| = \mathbf{b}_\| \quad\Rightarrow\quad \mathbf{x}_\| = A^+\mathbf{b}_\| = A^+\mathbf{b} \qquad \text{(minimum-norm solution)} \tag{15.4.6}$$

An explicit expression for the pseudoinverse A^+ will be given in Sec. 15.6 with the help of the SVD of A. We will also show there that $A^+\mathbf{b}_\| = A^+\mathbf{b}$. In conclusion, the *most general solution* of the least-squares problem (15.4.1) is given by:

$$\boxed{\mathbf{x} = A^+\mathbf{b} + \mathbf{x}_\perp} \tag{15.4.7}$$

where $\mathbf{x}_\perp$ is an *arbitrary* vector in $N(A)$. The arbitrariness of $\mathbf{x}_\perp$ parametrizes the non-uniqueness of the solution $\mathbf{x}$.

The pseudoinverse solution $\mathbf{x}_\|$ is also recognized to be that particular solution of the least-squares problem that has *minimum norm*. This follows from (15.4.4):

$$\|\mathbf{x}\|^2 = \|\mathbf{x}_\|\|^2 + \|\mathbf{x}_\perp\|^2 = \|A^+\mathbf{b}\|^2 + \|\mathbf{x}_\perp\|^2 \tag{15.4.8}$$

which shows that the norm $\|\mathbf{x}\|$ is minimum when $\mathbf{x}_\perp = 0$, or, when $\mathbf{x} = \mathbf{x}_\|$. Fig. 15.4.1 illustrates this property.

The minimum-norm solution is computed by MATLAB's built-in function **pinv**, $\mathbf{x}_\| = \text{pinv}(A)*\mathbf{b}$. The solution obtained by MATLAB's backslash operator, $\mathbf{x} = A\backslash\mathbf{b}$, does not, in general, coincide with $\mathbf{x}_\|$. It has a term $\mathbf{x}_\perp$ chosen such that the resulting vector $\mathbf{x}$ has *at most* r non-zero (and $M - r$ zero) entries, where r is the rank of A.

We obtained the general least-squares solution by purely geometric means using the orthogonality equation (15.4.2) and the orthogonal decompositions of $\mathbf{x}$ and $\mathbf{b}$. An alternative approach is to substitute (15.4.5) directly into the performance index and use the fact that $\mathbf{e}_\|$ and $\mathbf{e}_\perp$ are orthogonal:

$$\mathcal{J} = \|\mathbf{e}\|^2 = \|\mathbf{e}_\|\|^2 + \|\mathbf{e}_\perp\|^2 = \|\mathbf{b}_\| - A\mathbf{x}_\|\|^2 + \|\mathbf{b}_\perp\|^2 \tag{15.4.9}$$

This expression is minimized when $A\mathbf{x}_\| = \mathbf{b}_\|$, leading to the same general solution (15.4.7). The minimized value of the mean-square error is $\|\mathbf{b}_\perp\|^2$.

The *full-rank* case deserves special mention. There are three possibilities depending on whether the system $A\mathbf{x} = \mathbf{b}$ is over-determined, under-determined, or square. Then, one or both of the null subspaces consist only of the zero vector:

1. $N > M$, $r = M$, $N(A) = \{0\}$, $R(A^T) = \mathbb{R}^M$, (over-determined)
2. $M > N$, $r = N$, $N(A^T) = \{0\}$, $R(A) = \mathbb{R}^N$, (under-determined)
3. $N = M$, $r = N$, $N(A) = \{0\}$, $N(A^T) = \{0\}$, (square, invertible)

The three cases are depicted in Fig. 15.4.2. In the over-determined case, $N(A) = \{0\}$ and therefore, the least-squares solution is unique $\mathbf{x} = \mathbf{x}_\|$ and, as we saw earlier, it is given by $\mathbf{x} = (A^TA)^{-1}A^T\mathbf{b}$. Comparing with (15.4.6), it follows that the pseudoinverse is in this case $A^+ = (A^TA)^{-1}A^T$.

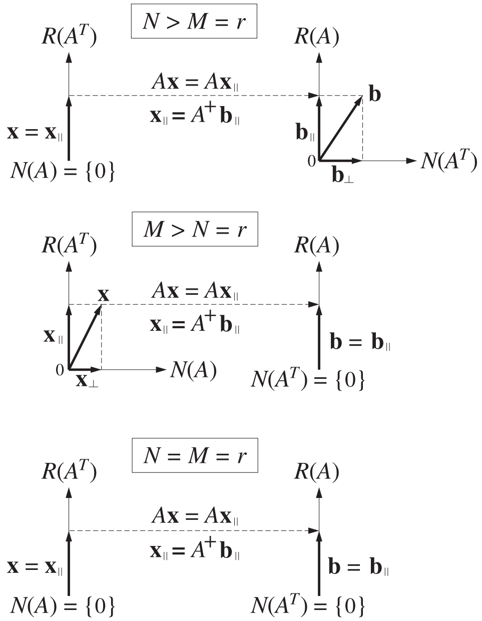


Fig. 15.4.2 Subspaces in the full-rank least-squares solutions of $A\mathbf{x} = \mathbf{b}$.

In the under-determined case, we have $\mathbf{b} = \mathbf{b}_\|$, that is, $\mathbf{b}$ is in the range of A, and therefore, $A\mathbf{x} = \mathbf{b}$ does have a solution. There are more unknowns than equations, and therefore, there is an infinity of solutions $\mathbf{x} = \mathbf{x}_\| + \mathbf{x}_\perp$. The minimum norm solution can be constructed as follows.

Because $\mathbf{x}_\| \in R(A^T)$, there is a coefficient vector $\mathbf{c} \in \mathbb{R}^N$ such that $\mathbf{x}_\| = A^T\mathbf{c}$. Then, the system reads $\mathbf{b} = A\mathbf{x} = A\mathbf{x}_\| = AA^T\mathbf{c}$. The $N\times N$ matrix AA^T is invertible because it has full rank $r = N$. Thus, we find $\mathbf{c} = (AA^T)^{-1}\mathbf{b}$ and hence, $\mathbf{x}_\| = A^T\mathbf{c} = A^T(AA^T)^{-1}\mathbf{b}$. It follows that $A^+ = A^T(AA^T)^{-1}$.

Finally, in the square invertible case, we have $\mathbf{x} = A^{-1}\mathbf{b}$. The three *full-rank* cases may be summarized as follows:

$$\begin{array}{llll} 1. & N > M = r, & \mathbf{x} = A^+\mathbf{b}, & A^+ = (A^TA)^{-1}A^T \\ 2. & M > N = r, & \mathbf{x} = A^+\mathbf{b} + \mathbf{x}_\perp, & A^+ = A^T(AA^T)^{-1} \\ 3. & N = M = r, & \mathbf{x} = A^{-1}\mathbf{b}, & A^+ = A^{-1} \end{array} \tag{15.4.10}$$

In the last two cases, the equation $A\mathbf{x} = \mathbf{b}$ is satisfied exactly. In the first case, it is satisfied only in the least-squares sense.

Example 15.4.1: Solve the two systems of equations:

$$\begin{cases} x = 1 \\ x = 2 \end{cases} \quad \text{and} \quad \begin{cases} 2x = 2 \\ x = 2 \end{cases}$$

Solution: The least-squares minimization problems and their solutions are in the two cases:

$$\mathcal{J} = (x-1)^2 + (x-2)^2 = \min \quad \Rightarrow \quad \frac{\partial \mathcal{J}}{\partial x} = 2(x-1) + 2(x-2) = 0 \quad \Rightarrow \quad x = 1.5$$

$$\mathcal{J} = (2x-2)^2 + (x-2)^2 = \min \quad \Rightarrow \quad \frac{\partial \mathcal{J}}{\partial x} = 4(2x-2) + 2(x-2) = 0 \quad \Rightarrow \quad x = 1.2$$

It may be surprising that the solutions are different since the first equations of the two systems are the same, differing only by an overall scale factor. The presence of the scale factor introduces an effective weighting of the performance index which alters the relative importance of the squared terms. Indeed, the second performance index may be rewritten as:

$$\mathcal{J} = 4(x-1)^2 + (x-2)^2$$

which assigns the weights 4:1 to the two terms, as opposed to the original 1:1. Generalizing this example, we may express the systems in the form $A\mathbf{x} = \mathbf{b}$:

$$\begin{array}{l} a_1 x = b_1 \\ a_2 x = b_2 \end{array} \quad \Rightarrow \quad \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} x = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}, \quad A = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \tag{15.4.11}$$

This is recognized as an overdetermined full-rank case, which can be solved by the pseudoinverse $A^+ = (A^TA)^{-1}A^T$. Noting that $A^TA = a_1^2 + a_2^2$, we have:

$$\begin{bmatrix} a_1 \\ a_2 \end{bmatrix}^+ = \frac{[a_1, a_2]}{a_1^2 + a_2^2} \quad \Rightarrow \quad x = A^+\mathbf{b} = \frac{1}{a_1^2 + a_2^2}[a_1, a_2]\begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = \frac{a_1 b_1 + a_2 b_2}{a_1^2 + a_2^2}$$

The first system has $[a_1, a_2] = [1, 1]$ and $[b_1, b_2] = [1, 2]$, and the second system, $[a_1, a_2] = [2, 1]$ and $[b_1, b_2] = [2, 2]$. If we multiply both sides of Eq. (15.4.11) by the weights w_1, w_2, we get the system and solution:

$$\begin{bmatrix} w_1 a_1 \\ w_2 a_2 \end{bmatrix} x = \begin{bmatrix} w_1 b_1 \\ w_2 b_2 \end{bmatrix} \quad \Rightarrow \quad x = \frac{w_1^2 a_1 b_1 + w_2^2 a_2 b_2}{w_1^2 a_1^2 + w_2^2 a_2^2} \tag{15.4.12}$$

The differences between (15.4.11) and (15.4.12) can be explained by inspecting the corresponding performance indices that are being minimized:

$$\mathcal{J} = (a_1 x - b_1)^2 + (a_2 x - b_2)^2, \qquad \mathcal{J} = w_1^2(a_1 x - b_1)^2 + w_2^2(a_2 x - b_2)^2$$

The scale factors w_1, w_2 alter the relative weighting of the terms in $\mathcal{J}$. □

Example 15.4.2: Find the minimum norm solution, as well as the most general least-squares solution of the system:

$$x_1 + x_2 = 2 \quad \Leftrightarrow \quad [1, 1]\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = [2], \quad A = [1, 1], \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad \mathbf{b} = [2]$$

Solution: This is an under-determined full-rank case. The minimum norm solution is computed using the pseudoinverse $A^+ = A^T(AA^T)^{-1}$. We have, $AA^T = 2$, therefore,

$$A^+ = \frac{1}{2}\begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} \quad \Rightarrow \quad \mathbf{x}_\parallel = A^+\mathbf{b} = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}[2] = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

The most general vector in the one-dimensional null space of A has the form:

$$\mathbf{x}_\perp = \begin{bmatrix} z \\ -z \end{bmatrix} \quad \Leftrightarrow \quad [1, 1]\begin{bmatrix} z \\ -z \end{bmatrix} = 0 \quad \Leftrightarrow \quad A\mathbf{x}_\perp = 0$$

Therefore, the most general least-squares solution will have the form:

$$\mathbf{x} = \mathbf{x}_\parallel + \mathbf{x}_\perp = \begin{bmatrix} 1 \\ 1 \end{bmatrix} + \begin{bmatrix} z \\ -z \end{bmatrix} = \begin{bmatrix} 1 + z \\ 1 - z \end{bmatrix}$$

It is evident that the norm-square $\|\mathbf{x}\|^2 = (z+1)^2 + (z-1)^2 = 2 + 2z^2$ is minimized when $z = 0$. MATLAB's backslash solution $\mathbf{x} = A\backslash\mathbf{b} = \begin{bmatrix} 2 \\ 0 \end{bmatrix}$ is obtained when $z = 1$ and corresponds to the point of intersection of the line $x_1 + x_2 = 2$ with the x_1 axis. A geometrical picture of the general solution is shown in Fig. 15.4.3.

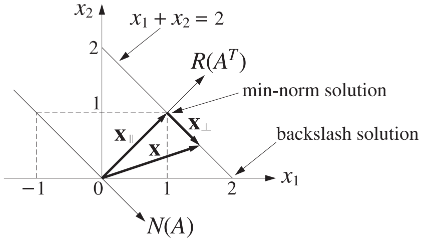


Fig. 15.4.3 The minimum norm solution is perpendicular to the straight line $x_1 + x_2 = 2$.

The equation $x_1 + x_2 = 2$ is represented by a straight line on the x_1x_2 plane. Any point on the line is a solution. In particular, the minimum-norm solution $\mathbf{x}_\parallel$ is obtained by drawing the perpendicular from the origin to the line.

The direction of $\mathbf{x}_\parallel$ defines the 1-dimensional range space $R(A^T)$. The orthogonal direction to $R(A^T)$, which is parallel to the line, is the direction of the 1-dimensional null subspace $N(A)$. In the more general case, we may replace the given equation by:

$$a_1 x_1 + a_2 x_2 = b_1 \quad \Leftrightarrow \quad [a_1, a_2]\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = [b_1], \quad A = [a_1, a_2], \quad \mathbf{b} = [b_1]$$

The pseudoinverse of A and the min-norm solution are:

$$A^+ = A^T(AA^T)^{-1} = \frac{1}{a_1^2+a_2^2}\begin{bmatrix} a_1 \\ a_2 \end{bmatrix}, \quad \mathbf{x}_\parallel = A^+\mathbf{b} = \frac{1}{a_1^2+a_2^2}\begin{bmatrix} a_1 b_1 \\ a_2 b_1 \end{bmatrix}$$

Vectors in $N(A)$ and the most general least-squares solution are given by:

$$\mathbf{x}_\perp = \frac{1}{a_1^2+a_2^2}\begin{bmatrix} a_2 z \\ -a_1 z \end{bmatrix}, \quad \mathbf{x} = \mathbf{x}_\parallel + \mathbf{x}_\perp = \frac{1}{a_1^2+a_2^2}\begin{bmatrix} a_1 b_1 + a_2 z \\ a_2 b_1 - a_1 z \end{bmatrix}$$

It is easily verified that $A\mathbf{x}_\perp = 0$ and that $\|\mathbf{x}\|^2$ is minimized when $z = 0$. □

Example 15.4.3: The pseudoinverses of N-dimensional column and row vectors are:

$$\mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_N \end{bmatrix} \quad \Rightarrow \quad \mathbf{a}^+ = \frac{\mathbf{a}^T}{\|\mathbf{a}\|^2} \quad \text{and} \quad \mathbf{a}^T = [a_1, a_2, \dots, a_N] \quad \Rightarrow \quad (\mathbf{a}^T)^+ = \frac{\mathbf{a}}{\|\mathbf{a}\|^2}$$

where $\|\mathbf{a}\|^2 = \mathbf{a}^T\mathbf{a} = a_1^2 + a_2^2 + \cdots + a_N^2$. Thus, we obtain the minimum-norm solutions:

$$\begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_N \end{bmatrix} x = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{bmatrix} \quad \Rightarrow \quad x = \mathbf{a}^+\mathbf{b} = \frac{\mathbf{a}^T\mathbf{b}}{\mathbf{a}^T\mathbf{a}} = \frac{a_1 b_1 + a_2 b_2 + \cdots + a_N b_N}{a_1^2 + a_2^2 + \cdots + a_N^2}$$

$$a_1 x_1 + a_2 x_2 + \cdots + a_N x_N = b \quad \Rightarrow \quad \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} = \frac{1}{a_1^2 + a_2^2 + \cdots + a_N^2}\begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_N \end{bmatrix} b$$

15.5 The Singular Value Decomposition

Given an $N\times M$ matrix $A \in \mathbb{R}^{N\times M}$ of rank $r \le \min(N, M)$, the *singular value decomposition theorem* [1234] states that there exist *orthogonal* matrices $U \in \mathbb{R}^{N\times N}$ and $V \in \mathbb{R}^{M\times M}$ such that A is factored in the form:

$$\boxed{A = U\Sigma V^T} \qquad \text{(SVD)} \tag{15.5.1}$$

where $\Sigma \in \mathbb{R}^{N\times M}$ is an $N\times M$ diagonal matrix, partitioned in the form:

$$\Sigma = \left[\begin{array}{c|c} \Sigma_r & 0 \\ \hline 0 & 0 \end{array}\right] \tag{15.5.2}$$

with Σ_r a square diagonal matrix in $\mathbb{R}^{r\times r}$:

$$\boxed{\Sigma_r = \mathrm{diag}(\sigma_1, \sigma_2, \dots, \sigma_r)} \tag{15.5.3}$$

with positive diagonal entries called the *singular values* of A and arranged in decreasing order:

$$\sigma_1 \ge \sigma_2 \ge \cdots \ge \sigma_r > 0 \tag{15.5.4}$$

The orthogonal matrices U, V are not unique, but the singular values σ_i are. To clarify the structure of Σ and the blocks of zeros bordering Σ_r, we give below the expressions for Σ for the case of a 6×4 matrix A of rank $r = 1, 2, 3, 4$:

$$\left[\begin{array}{c|ccc} \sigma_1 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right], \quad \left[\begin{array}{cc|cc} \sigma_1 & 0 & 0 & 0 \\ 0 & \sigma_2 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right], \quad \left[\begin{array}{ccc|c} \sigma_1 & 0 & 0 & 0 \\ 0 & \sigma_2 & 0 & 0 \\ 0 & 0 & \sigma_3 & 0 \\ \hline 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right], \quad \left[\begin{array}{cccc} \sigma_1 & 0 & 0 & 0 \\ 0 & \sigma_2 & 0 & 0 \\ 0 & 0 & \sigma_3 & 0 \\ 0 & 0 & 0 & \sigma_4 \\ \hline 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right]$$

The orthogonality of U, V may be expressed by $U^T U = UU^T = I_N$ and $V^T V = VV^T = I_M$. These just mean the U has N orthonormal columns that form a complete basis for $\mathbb{R}^N$, and V has M orthonormal columns that form a basis for $\mathbb{R}^M$.

Denoting the columns of U by $\mathbf{u}_i$, $i = 1, 2, \dots, N$, and the columns of V by $\mathbf{v}_i$, $i = 1, 2, \dots, M$, we may partition U, V in a compatible way as in Eq. (15.5.2):

$$\begin{aligned} U &= [\underbrace{\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_r}_{U_r}, \underbrace{\mathbf{u}_{r+1}, \dots, \mathbf{u}_N}_{\tilde{U}_r}] = [U_r \mid \tilde{U}_r] \\ V &= [\underbrace{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_r}_{V_r}, \underbrace{\mathbf{v}_{r+1}, \dots, \mathbf{v}_M}_{\tilde{V}_r}] = [V_r \mid \tilde{V}_r] \end{aligned} \tag{15.5.5}$$

Then, Eq. (15.5.1) can be written in the form:

$$A = [U_r \mid \tilde{U}_r]\left[\begin{array}{c|c} \Sigma_r & 0 \\ \hline 0 & 0 \end{array}\right]\left[\begin{array}{c} V_r^T \\ \hline \tilde{V}_r^T \end{array}\right] = U_r \Sigma_r V_r^T \tag{15.5.6}$$

or, as a sum of r rank-1 matrices:

$$A = \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^T = \sigma_1 \mathbf{u}_1 \mathbf{v}_2^T + \sigma_2 \mathbf{u}_2 \mathbf{v}_2^T + \cdots + \sigma_r \mathbf{u}_r \mathbf{v}_r^T \tag{15.5.7}$$

The submatrices have dimensions $U_r \in \mathbb{R}^{N\times r}$, $\tilde{U}_r \in \mathbb{R}^{N\times(N-r)}$, $V_r \in \mathbb{R}^{M\times r}$, and $\tilde{V}_r \in \mathbb{R}^{M\times(M-r)}$. The orthogonality and completeness properties of U, V may be expressed equivalently in terms of these submatrices:

$$\begin{aligned} &U_r^T U_r = I_r\,, \quad \tilde{U}_r^T \tilde{U}_r = I_{N-r}\,, \quad U_r^T \tilde{U}_r = 0\,, \quad U_r U_r^T + \tilde{U}_r \tilde{U}_r^T = I_N \\ &V_r^T V_r = I_r\,, \quad \tilde{V}_r^T \tilde{V}_r = I_{M-r}\,, \quad V_r^T \tilde{V}_r = 0\,, \quad V_r V_r^T + \tilde{V}_r \tilde{V}_r^T = I_M \end{aligned} \tag{15.5.8}$$

For example, we have:

$$U^T U = \left[\begin{array}{c|c} U_r^T U_r & U_r^T \tilde{U}_r \\ \hline \tilde{U}_r^T U_r & \tilde{U}_r^T \tilde{U}_r \end{array}\right] = \left[\begin{array}{c|c} I_r & 0 \\ \hline 0 & I_{N-r} \end{array}\right] = I_N\,, \quad U_r U_r^T + \tilde{U}_r \tilde{U}_r^T = UU^T = I_N$$

The SVD of A provides also the SVD of A^T, that is, $A^T = V\Sigma^T U^T$. The singular values of A^T coincide with those of A. The matrix Σ^T has dimension $M\times N$, but since $\Sigma_r^T = \Sigma_r$, we have:

$$A^T = V\Sigma^T U^T = [V_r \mid \tilde{V}_r]\left[\begin{array}{c|c}\Sigma_r & 0\\ \hline 0 & 0\end{array}\right]\left[\begin{array}{c}U_r^T\\ \hline \tilde{U}_r^T\end{array}\right] = V_r\Sigma_r U_r^T \tag{15.5.9}$$

Although A and A^T can be constructed only from U_r, Σ_r, V_r, the other submatrices $\tilde{U}_r, \tilde{V}_r$ are needed in order to characterize the four fundamental subspaces of A, and are needed also in the least-squares solutions.

Multiplying (15.5.6) from the right by V_r and $\tilde{V}_r$ and multiplying (15.5.9) by U_r and $\tilde{U}_r$ and using (15.5.8), we obtain:

$$\begin{aligned} &AV_r = U_r\Sigma_r V_r^T V_r = U_r\Sigma_r\,, && A\tilde{V}_r = U_r\Sigma_r V_r^T\tilde{V}_r = 0\\ &A^T U_r = V_r\Sigma_r U_r^T U_r = V_r\Sigma_r\,, && A^T\tilde{U}_r = V_r\Sigma_r U_r^T\tilde{U}_r = 0\end{aligned}$$

or, explicitly in terms of the basis vectors $\mathbf{u}_i, \mathbf{v}_i$:

$$\begin{array}{lclll} AV_r = U_r\Sigma_r & & A\mathbf{v}_i = \sigma_i\mathbf{u}_i\,, & i = 1,2,\dots,r\\ A\tilde{V}_r = 0 & \Leftrightarrow & A\mathbf{v}_i = 0\,, & i = r+1,\dots,M\\ A^T U_r = V_r\Sigma_r & & A^T\mathbf{u}_i = \sigma_i\mathbf{v}_i\,, & i = 1,2,\dots,r\\ A^T\tilde{U}_r = 0 & & A^T\mathbf{u}_i = 0\,, & i = r+1,\dots,N\end{array} \tag{15.5.10}$$

These equations show that $\mathbf{u}_i$ and $\mathbf{v}_i$, $i = 1,2,\dots,r$, lie in the range spaces $R(A)$ and $R(A^T)$, respectively. Moreover, they provide orthonormal bases for these two subspaces. Similarly, $\mathbf{v}_i$, $i = r+1,\dots,M$, and $\mathbf{u}_i$, $i = r+1,\dots,N$, are bases for the null subspaces $N(A)$ and $N(A^T)$, respectively.

Thus, a second part of the *fundamental theorem of linear algebra* is that the matrices $U_r, \tilde{U}_r, V_r, \tilde{V}_r$ provide orthonormal bases for the four fundamental subspaces of A, and with respect to these bases, A has a diagonal form (the Σ). The subspaces, their bases, and the corresponding projectors onto them are:

$$\begin{array}{llll} R(A) = \text{span}\{U_r\}\,, & \dim = r\,, & U_r^T U_r = I_r\,, & \mathcal{P}_{R(A)} = U_r U_r^T\\ N(A^T) = \text{span}\{\tilde{U}_r\}\,, & \dim = N-r\,, & \tilde{U}_r^T\tilde{U}_r = I_{N-r}\,, & \mathcal{P}_{N(A^T)} = \tilde{U}_r\tilde{U}_r^T\\ R(A^T) = \text{span}\{V_r\}\,, & \dim = r\,, & V_r^T V_r = I_r\,, & \mathcal{P}_{R(A^T)} = V_r V_r^T\\ N(A) = \text{span}\{\tilde{V}_r\}\,, & \dim = M-r\,, & \tilde{V}_r^T V_r = I_{M-r}\,, & \mathcal{P}_{N(A)} = \tilde{V}_r\tilde{V}_r^T\end{array} \tag{15.5.11}$$

The vectors $\mathbf{u}_i$ and $\mathbf{v}_i$ are referred to as the *left* and *right* singular vectors of A and are the eigenvectors of the matrices AA^T and A^TA, respectively. Indeed, it follows from the orthogonality of U and V that:

$$\begin{aligned} &A^TA = V\Sigma^T U^T U\Sigma V^T = V(\Sigma^T\Sigma)V^T\,, && \Sigma^T\Sigma = \left[\begin{array}{c|c}\Sigma_r^2 & 0\\ \hline 0 & 0\end{array}\right] \in \mathbb{R}^{M\times M}\\ &AA^T = U\Sigma V^T V\Sigma^T U^T = U(\Sigma\Sigma^T)U^T\,, && \Sigma\Sigma^T = \left[\begin{array}{c|c}\Sigma_r^2 & 0\\ \hline 0 & 0\end{array}\right] \in \mathbb{R}^{N\times N}\end{aligned} \tag{15.5.12}$$

It is evident from these that V and U are the matrices of eigenvectors of A^TA and AA^T and that the corresponding non-zero eigenvalues are $\lambda_i = \sigma_i^2$, $i = 1,2,\dots,r$. The ranks of A^TA and AA^T are equal to the rank r of A.

The SVD factors V, U could, in principle, be obtained by solving the eigenvalue problems of A^TA and AA^T. However, in practice, loss of accuracy can occur in squaring the matrix A. Methods of computing the SVD directly from A are available.

A simplified proof of the SVD is as follows. We assume that $N \geq M$ and that A has full rank $r = M$ (the proof can easily be modified for the general case.) First, we solve the eigenvalue problem of the matrix A^TA:

$$A^TA = V\Lambda V^T\,,\quad \Lambda = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_M) \in \mathbb{R}^{M\times M}$$

Because A^TA has full rank, it will be strictly positive definite, its eigenvalues will be positive, and the corresponding eigenvectors may be chosen to be orthonormal, that is, $V^TV = VV^T = I_M$. Arranging the eigenvalues in decreasing order, we define $\sigma_i = \sqrt{\lambda_i}$, $i = 1,2,\dots,M$, and $\Sigma_1 = \Lambda^{1/2} = \text{diag}(\sigma_1,\dots,\sigma_M) \in \mathbb{R}^{M\times M}$. Then, we define $U_1 = AV\Sigma_1^{-1}$, which is an $N\times M$ matrix with orthonormal columns:

$$U_1^T U_1 = \Sigma_1^{-1}V^T(A^TA)V\Sigma_1^{-1} = \Sigma_1^{-1}V^T(V\Sigma_1^2V^T)V\Sigma_1^{-1} = I_M$$

Next, we solve for A. We have $U_1\Sigma_1 = AV$, and $U_1\Sigma_1V^T = AVV^T = A$, or

$$A = U_1\Sigma_1V^T \qquad \text{(economy SVD)} \tag{15.5.13}$$

The $N\times M$ matrix U_1 may be enlarged into an $N\times N$ orthogonal matrix by adjoining to it $(N-M)$ orthonormal columns U_2 such that $U_2^TU_1 = 0$, and similarly, the $M\times M$ diagonal matrix Σ_1 may be enlarged into an $N\times M$ matrix Σ. Then, Eq. (15.5.13) may be rewritten in the standard full SVD form:

$$A = U_1\Sigma_1V^T = [U_1 \mid U_2]\left[\begin{array}{c}\Sigma_1\\ \hline 0\end{array}\right]V^T = U\Sigma V^T \tag{15.5.14}$$

Eq. (15.5.13) is called the *economy* or *thin* SVD because the U_1 matrix has the same size as A but has orthonormal columns, and Σ_1 has size $M\times M$. For many applications, such as SVD signal enhancement, the economy SVD is sufficient. In MATLAB, the full and the economy SVDs are obtained with the calls:

```
[U,S,V] = svd(A);            % full SVD
[U1,S1,V] = svd(A,0);        % economy SVD
```

Example 15.5.1: To illustrate the loss of accuracy in forming A^TA, consider the 4×3 matrix:

$$A = \begin{bmatrix} 1 & 1 & 1\\ \epsilon & 0 & 0\\ 0 & \epsilon & 0\\ 0 & 0 & \epsilon\end{bmatrix} \quad\Rightarrow\quad A^TA = \begin{bmatrix} 1+\epsilon^2 & 1 & 1\\ 1 & 1+\epsilon^2 & 1\\ 1 & 1 & 1+\epsilon^2\end{bmatrix}$$

The matrix A remains full rank to order $O(\epsilon)$, but A^TA requires that we work to order $O(\epsilon^2)$. The singular values of A are obtained from the eigenvalues of A^TA:

$$\lambda_1 = 3+\epsilon^2,\ \lambda_2 = \lambda_3 = \epsilon^2 \quad\Rightarrow\quad \sigma_1 = \sqrt{3+\epsilon^2},\ \sigma_2 = \sigma_2 = \epsilon$$

The full SVD of A can be constructed along the lines described above. Starting with the eigenproblem of A^TA, we find:

$$A = U\Sigma V^T = \begin{bmatrix} 3\alpha & 0 & 0 & -\delta\epsilon \\ \alpha\epsilon & \beta & \gamma & \delta \\ \alpha\epsilon & -\beta & \gamma & \delta \\ \alpha\epsilon & 0 & -2\gamma & \delta \end{bmatrix} \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} a & b & c \\ a & -b & c \\ a & 0 & -2c \end{bmatrix}^T$$

where $a = \dfrac{1}{\sqrt{3}}$, $\beta = b = \dfrac{1}{\sqrt{2}}$, $\gamma = c = \dfrac{1}{\sqrt{6}}$, $\alpha = \dfrac{1}{\sqrt{3(3+\epsilon^2)}}$, and $\delta = \dfrac{1}{\sqrt{(3+\epsilon^2)}}$. □

Example 15.5.2: Consider the full SVD of the 4×2 matrix A:

$$A = \begin{bmatrix} 0.5 & 1.0 \\ 1.1 & 0.2 \\ 1.1 & 0.2 \\ 0.5 & 1.0 \end{bmatrix} = \begin{bmatrix} 0.5 & 0.5 & -0.1 & -0.7 \\ 0.5 & -0.5 & -0.7 & 0.1 \\ 0.5 & -0.5 & 0.7 & -0.1 \\ 0.5 & 0.5 & 0.1 & 0.7 \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0.8 & -0.6 \\ 0.6 & 0.8 \end{bmatrix}^T = U\Sigma V^T$$

Its economy SVD is:

$$A = \begin{bmatrix} 0.5 & 1.0 \\ 1.1 & 0.2 \\ 1.1 & 0.2 \\ 0.5 & 1.0 \end{bmatrix} = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & -0.5 \\ 0.5 & -0.5 \\ 0.5 & 0.5 \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0.8 & -0.6 \\ 0.6 & 0.8 \end{bmatrix}^T$$

The choice of the last two columns of U is not unique. They can be transformed by any 2×2 orthogonal matrix without affecting the SVD. For example, v5.3 of MATLAB produces the U matrix:

$$U = \begin{bmatrix} 0.5 & 0.5 & -0.1544 & -0.6901 \\ 0.5 & -0.5 & -0.6901 & 0.1544 \\ 0.5 & -0.5 & 0.6901 & -0.1544 \\ 0.5 & 0.5 & 0.1544 & 0.6901 \end{bmatrix}$$

The last two columns of the two Us are related by the 2×2 orthogonal matrix C:

$$\begin{bmatrix} -0.1544 & -0.6901 \\ -0.6901 & 0.1544 \\ 0.6901 & -0.1544 \\ 0.1544 & 0.6901 \end{bmatrix} = \begin{bmatrix} -0.1 & -0.7 \\ -0.7 & 0.1 \\ 0.7 & -0.1 \\ 0.1 & 0.7 \end{bmatrix} C\,, \qquad C = \begin{bmatrix} 0.9969 & -0.0781 \\ 0.0781 & 0.9969 \end{bmatrix}$$

where $C^TC = I_2$. □

Complex-Valued Case

The SVD of a complex-valued matrix $A \in \mathbb{C}^{N\times M}$ takes the form:

$$A = U\Sigma V^\dagger \tag{15.5.15}$$

where † denotes the Hermitian-conjugate, or conjugate-transpose, $V^\dagger = V^{*T}$. The matrix Σ is exactly as in the real case, and U, V are *unitary* matrices $U \in \mathbb{C}^{N\times N}$ and $V \in \mathbb{C}^{M\times M}$, that is,

$$UU^\dagger = U^\dagger U = I_N\,, \quad VV^\dagger = V^\dagger V = I_M \tag{15.5.16}$$

Maximization Criterion for the SVD

The singular values and singular vectors of a matrix A of rank r can be characterized by the following maximization criterion [1321].

First, the maximum singular value σ_1 and singular vectors $\mathbf{u}_1, \mathbf{v}_1$ are the solutions of the maximization criterion:[†]

$$\sigma_1 = \max_{\|\mathbf{u}\|=1}\max_{\|\mathbf{v}\|=1} \mathbf{u}^\dagger A\mathbf{v} = \mathbf{u}_1^\dagger A\mathbf{v}_1 \tag{15.5.17}$$

Then, the remaining singular values and vectors are the solutions of the criteria:

$$\begin{aligned} &\sigma_i = \max_{\|\mathbf{u}\|=1}\max_{\|\mathbf{v}\|=1} \mathbf{u}^\dagger A\mathbf{v} = \mathbf{u}_i^\dagger A\mathbf{v}_i\,, \quad i = 2,\dots,r \\ &\qquad \text{subject to the constraints:} \quad \mathbf{u}^\dagger\mathbf{u}_j = \mathbf{v}^\dagger\mathbf{v}_j = 0\,, \quad j = 1,2,\dots,i-1 \end{aligned} \tag{15.5.18}$$

The proof is straightforward. Using the Cauchy-Schwarz inequality and the constraints $\|\mathbf{u}\| = \|\mathbf{v}\| = 1$, and that the Euclidean norm of A is σ_1, we have:

$$|\mathbf{u}^\dagger A\mathbf{v}| \le \|\mathbf{u}\|\|A\|\|\mathbf{v}\| = \|A\| = \sigma_1$$

with the equality being realized when $\mathbf{u} = \mathbf{u}_1$ and $\mathbf{v} = \mathbf{v}_1$.

For the next singular value σ_2, we must maximize $\mathbf{u}^\dagger A\mathbf{v}$ over all vectors $\mathbf{u}, \mathbf{v}$ that are orthogonal to $\mathbf{u}_1, \mathbf{v}_1$, that is, $\mathbf{u}^\dagger\mathbf{u}_1 = \mathbf{v}^\dagger\mathbf{v}_1 = 0$. Using the SVD of A, we may separate the contribution of $\mathbf{u}_1, \mathbf{v}_1$:

$$A = \sigma_1\mathbf{u}_1\mathbf{v}_1^\dagger + \sum_{i=2}^{r}\sigma_1\mathbf{u}_i\mathbf{v}_i^\dagger \equiv \sigma_1\mathbf{u}_1\mathbf{v}_1^\dagger + A_2$$

Then, the constraints imply that $\mathbf{u}^\dagger A\mathbf{v} = \mathbf{u}^\dagger(\sigma_1\mathbf{u}_1\mathbf{v}_1^\dagger + A_2)\mathbf{v} = \mathbf{u}^\dagger A_2\mathbf{v}$. But from the previous result, the maximum of this quantity is the maximum singular value of A_2, that is, σ_2, and this maximum is realized when $\mathbf{u} = \mathbf{u}_2$ and $\mathbf{v} = \mathbf{v}_2$. Then we repeat this argument by separating out the remaining singular terms $\sigma_i\mathbf{u}_i\mathbf{v}_i^\dagger$ one at a time, till we exhaust all the singular values.

This theorem is useful in canonical correlation analysis and in characterizing the angles between subspaces.

15.6 Moore-Penrose Pseudoinverse

For a full-rank $N\times N$ matrix with SVD $A = U\Sigma V^T$, the ordinary inverse is obtained by inverting the SVD factors and writing them in *reverse* order:

$$A^{-1} = V^{-T}\Sigma^{-1}U^{-1} = V\Sigma^{-1}U^T \tag{15.6.1}$$

where we used the orthogonality properties to write $V^{-T} = V$ and $U^{-1} = U^T$. For an $N\times M$ rectangular matrix with defective rank r, Σ^{-1} cannot be defined even if it were

[†] The quantity $\mathbf{u}^\dagger A\mathbf{v}$ could just as well be replaced by its absolute value $|\mathbf{u}^\dagger A\mathbf{v}|$ in (15.5.17) and (15.5.18).

square because some of its singular values are zero. For a scalar x, we may define its *pseudoinverse* by:

$$x^+ = \begin{cases} x^{-1}, & \text{if } \ x \neq 0 \\ 0, & \text{if } \ x = 0 \end{cases} \tag{15.6.2}$$

For a square $M\times M$ diagonal matrix, we define its pseudoinverse to be the diagonal matrix of the pseudoinverses:

$$\Sigma = \mathrm{diag}(\sigma_1, \sigma_2, \dots, \sigma_M) \quad \Rightarrow \quad \Sigma^+ = \mathrm{diag}(\sigma_1^+, \sigma_2^+, \dots, \sigma_M^+) \tag{15.6.3}$$

And, for an $N\times M$ rectangular diagonal matrix of r non-zero singular values $\Sigma \in \mathbb{R}^{N\times M}$, we define its pseudoinverse to be the $M\times N$ diagonal matrix $\Sigma^+ \in \mathbb{R}^{M\times N}$:

$$\Sigma = \left[\begin{array}{c|c} \Sigma_r & 0 \\ \hline 0 & 0 \end{array}\right] \in \mathbb{R}^{N\times M} \quad \Rightarrow \quad \Sigma^+ = \left[\begin{array}{c|c} \Sigma_r^{-1} & 0 \\ \hline 0 & 0 \end{array}\right] \in \mathbb{R}^{M\times N} \tag{15.6.4}$$

The pseudoinverse of an $N\times M$ matrix A is defined by replacing Σ^{-1} in Eq. (15.6.1) by Σ^+, that is, if $A = U\Sigma V^T \in \mathbb{R}^{N\times M}$, then $A^+ \in \mathbb{R}^{M\times N}$:

$$\boxed{A^+ = V\Sigma^+ U^T} \qquad \text{(Moore-Penrose pseudoinverse)} \tag{15.6.5}$$

Equivalently, using the block-matrix form (15.5.6), we have:

$$\begin{aligned} A &= [U_r \mid \tilde{U}_r]\left[\begin{array}{c|c} \Sigma_r & 0 \\ \hline 0 & 0 \end{array}\right]\left[\begin{array}{c} V_r^T \\ \hline \tilde{V}_r^T \end{array}\right] = U_r \Sigma_r V_r^T \\ A^+ &= [V_r \mid \tilde{V}_r]\left[\begin{array}{c|c} \Sigma_r^{-1} & 0 \\ \hline 0 & 0 \end{array}\right]\left[\begin{array}{c} U_r^T \\ \hline \tilde{U}_r^T \end{array}\right] = V_r \Sigma_r^{-1} U_r^T \end{aligned} \tag{15.6.6}$$

Eqs. (15.6.6) can be written as sums of r rank-1 matrices:

$$\begin{aligned} A &= \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^T = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^T + \sigma_2 \mathbf{u}_2 \mathbf{v}_2^T + \cdots + \sigma_r \mathbf{u}_r \mathbf{v}_r^T \\ A^+ &= \sum_{i=1}^{r} \frac{1}{\sigma_i} \mathbf{v}_i \mathbf{u}_i^T = \frac{1}{\sigma_1} \mathbf{v}_1 \mathbf{u}_1^T + \frac{1}{\sigma_2} \mathbf{v}_2 \mathbf{u}_2^T + \cdots + \frac{1}{\sigma_r} \mathbf{v}_r \mathbf{u}_r^T \end{aligned} \tag{15.6.7}$$

The matrix A^+ satisfies (and is uniquely determined by) the four standard Penrose conditions [1234]:

$$\begin{aligned} AA^+A &= A\,, \qquad (AA^+)^T = AA^+ \\ A^+AA^+ &= A^+\,, \quad\ (A^+A)^T = A^+A \end{aligned} \tag{15.6.8}$$

These conditions are equivalent to the fact that AA^+ and A^+A are the projectors onto the range spaces $R(A)$ and $R(A^T)$, respectively. Indeed, using the definition (15.6.6) and Eq. (15.5.11), we have:

$$\mathcal{P}_{R(A)} = U_r U_r^T = AA^+\,, \qquad \mathcal{P}_{R(A^T)} = V_r V_r^T = A^+A \tag{15.6.9}$$

It is straightforward also to verify the three expressions for A^+ given by Eq. (15.4.10) for the full-rank cases. For example, if $N > M = r$, the matrix V_r is square and orthogonal, so that $A^TA = V_r\Sigma_r^2 V_r^T$ is invertible, $(A^TA)^{-1} = V_r \Sigma_r^{-2} V_r^T$. Thus,

$$(A^TA)^{-1}A^T = \big(V_r\Sigma_r^{-2}V_r^T\big)\big(V_r\Sigma_r U_r^T\big) = V_r\Sigma_r^{-1}U_r^T = A^+$$

15.7 Least-Squares Problems and the SVD

Having defined the pseudoinverse and convenient bases for the four fundamental subspaces of A, we may revisit the least-squares solution of the system $A\mathbf{x} = \mathbf{b}$.

First, we show that the solution of $A\mathbf{x}_\parallel = \mathbf{b}_\parallel$ is, indeed, given by the pseudoinverse A^+ acting directly on $\mathbf{b}$. By definition, we have $\mathbf{b}_\parallel = \mathcal{P}_{R(A)}\mathbf{b}$. Using the SVD of A and the projectors (15.6.9), we have:

$$A\mathbf{x}_\parallel = \mathbf{b}_\parallel \quad \Rightarrow \quad U_r\Sigma_r V_r^T \mathbf{x}_\parallel = U_r U_r^T \mathbf{b} \quad \Rightarrow \quad V_r^T \mathbf{x}_\parallel = \Sigma_r^{-1} U_r^T \mathbf{b}$$

where we multiplied both sides of the second equation by U_r^T and divided by Σ_r to get the third. Multiplying from the left by V_r and using (15.6.6), we find:

$$V_r V_r^T \mathbf{x}_\parallel = V_r \Sigma_r^{-1} U_r^T \mathbf{b} = A^+\mathbf{b}$$

but we have $\mathbf{x}_\parallel = V_r V_r^T \mathbf{x}_\parallel$, which follows from $(V_r V_r^T)^2 = V_r V_r^T$, that is, $\mathbf{x}_\parallel = V_r V_r^T \mathbf{x} = V_r V_r^T (V_r V_r^T \mathbf{x}) = V_r V_r^T \mathbf{x}_\parallel$. Thus, we find $\mathbf{x}_\parallel = A^+\mathbf{b}$. Using (15.6.8) and (15.6.9), we also have $A^+\mathbf{b} = (A^+AA^+)\mathbf{b} = A^+(AA^+\mathbf{b}) = A^+\mathbf{b}_\parallel$. Thus, we have shown:

$$\boxed{\mathbf{x}_\parallel = A^+\mathbf{b}_\parallel = A^+\mathbf{b}} \qquad \text{(minimum-norm solution)} \tag{15.7.1}$$

or, explicitly in terms of the non-zero singular values:

$$\boxed{\mathbf{x}_\parallel = A^+\mathbf{b} = V_r\Sigma_r^{-1}U_r^T\mathbf{b} = \sum_{i=1}^{r} \frac{1}{\sigma_i} \mathbf{v}_i \, \mathbf{u}_i^T \mathbf{b}} \tag{15.7.2}$$

We recall that the most general least-squares solution of $A\mathbf{x} = \mathbf{b}$ is given by $\mathbf{x} = \mathbf{x}_\parallel + \mathbf{x}_\perp$, where $\mathbf{x}_\perp \in N(A)$. We can give an explicit construction of $\mathbf{x}_\perp$ by noting that $\tilde{V}_r$ is an orthonormal basis for $N(A)$. Therefore, we may write $\mathbf{x}_\perp = \tilde{V}_r \mathbf{z}$, where $\mathbf{z}$ is an $(M - r)$-dimensional column vector of expansion coefficients, that is,

$$\mathbf{x}_\perp = \sum_{i=r+1}^{M} z_i \mathbf{v}_i = [\mathbf{v}_{r+1}, \mathbf{v}_{r+2}, \dots, \mathbf{v}_M] \begin{bmatrix} z_{r+1} \\ z_{r+2} \\ \vdots \\ z_M \end{bmatrix} = \tilde{V}_r \mathbf{z}$$

Because $\mathbf{x}_\perp$ is arbitrary, so is $\mathbf{z}$. Thus, the most general solution of the least-squares problem can be written in the form [1234]:

$$\boxed{\mathbf{x} = \mathbf{x}_\parallel + \mathbf{x}_\perp = A^+\mathbf{b} + \tilde{V}_r \mathbf{z}}\,, \quad \text{for arbitrary } \mathbf{z} \in \mathbb{R}^{M-r} \tag{15.7.3}$$

The error vector is:

$$\mathbf{e} = \mathbf{e}_\perp = \mathbf{b}_\perp = \mathcal{P}_{N(A^T)}\mathbf{b} = \tilde{U}_r \tilde{U}_r^T \mathbf{b} = (I_N - U_r U_r^T)\mathbf{b} = (I_N - AA^+)\mathbf{b}$$

and the minimized value of the least-squares performance index:

$$\mathcal{J}_{\min} = \|\mathbf{e}\|^2 = \mathbf{b}^T(I_N - AA^+)\mathbf{b} \tag{15.7.4}$$

where we used the property $(I_N - AA^+)^T(I_N - AA^+) = (I_N - AA^+)$, which can be proved directly using (15.6.8). Indeed,

$$(I_N - AA^+)^T(I_N - AA^+) = I_N - 2AA^+ + AA^+AA^+ = I_N - 2AA^+ + AA^+ = I_N - AA^+$$

Example 15.7.1: Here, we revisit Example 15.4.2 from the point of view of Eq. (15.7.3). The full and economy SVD of $A = [a_1, a_2]$ are:

$$A = [a_1, a_2] = [1]\,[\sigma_1, 0]\begin{bmatrix} a_1/\sigma_1 & -a_2/\sigma_1 \\ a_2/\sigma_1 & a_1/\sigma_1 \end{bmatrix}^T = [1]\,[\sigma_1]\begin{bmatrix} a_1/\sigma_1 \\ a_2/\sigma_1 \end{bmatrix}^T$$

with the singular value $\sigma_1 = \sqrt{a_1^2 + a_2^2}$. Thus, the pseudoinverse of A and the basis $\tilde{V}_r$ of $N(A)$ will be:

$$A^+ = [a_1, a_2]^+ = \begin{bmatrix} a_1/\sigma_1 \\ a_2/\sigma_1 \end{bmatrix}[\sigma_1^{-1}]\,[1]^T = \frac{1}{\sigma_1^2}\begin{bmatrix} a_1 \\ a_2 \end{bmatrix}, \quad \tilde{V}_r = \frac{1}{\sigma_1}\begin{bmatrix} -a_2 \\ a_1 \end{bmatrix}$$

It follows from (15.7.3) that the most general solution of $a_1x_1 + a_2x_2 = b_1$ will be:

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = A^+[b_1] + \frac{1}{\sigma_1}\begin{bmatrix} -a_2 \\ a_1 \end{bmatrix} z = \frac{1}{\sigma_1^2}\begin{bmatrix} a_1 \\ a_2 \end{bmatrix} b_1 + \frac{1}{\sigma_1}\begin{bmatrix} -a_2 \\ a_1 \end{bmatrix} z$$

which is equivalent to that given in Example 15.4.2 up to a redefinition of z. □

Example 15.7.2: Find the most general solution of the following linear system, and in particular, find the minimum-norm and MATLAB's backslash solutions:

$$A\mathbf{x} = \begin{bmatrix} 1.8 & 2.4 & 4.0 \\ -1.8 & -2.4 & -4.0 \\ 1.8 & 2.4 & 4.0 \\ -1.8 & -2.4 & -4.0 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 10 \\ 20 \\ 30 \\ 40 \end{bmatrix} = \mathbf{b}$$

A possible SVD of A is as follows:

$$A = U\Sigma V^T = \begin{bmatrix} 0.5 & 0.5 & -0.5 & 0.5 \\ -0.5 & -0.5 & -0.5 & 0.5 \\ 0.5 & -0.5 & 0.5 & 0.5 \\ -0.5 & 0.5 & 0.5 & 0.5 \end{bmatrix}\begin{bmatrix} 10 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} 0.36 & -0.48 & 0.8 \\ 0.48 & -0.64 & -0.6 \\ 0.80 & 0.60 & 0.0 \end{bmatrix}^T$$

The matrix A has rank one, so that the last three columns of U and the last two columns of V are not uniquely defined. The pseudoinverse of A will be:

$$A = \begin{bmatrix} 0.5 \\ -0.5 \\ 0.5 \\ -0.5 \end{bmatrix}[10]\,[0.36, 0.48, 0.80]\,, \quad A^+ = \begin{bmatrix} 0.36 \\ 0.48 \\ 0.80 \end{bmatrix}[10^{-1}]\,[0.5, -0.5, 0.5, -0.5]$$

Therefore, the minimum-norm solution is:

$$\mathbf{x}_\| = A^+\mathbf{b} = \begin{bmatrix} 0.36 \\ 0.48 \\ 0.80 \end{bmatrix}[10^{-1}]\,[0.5, -0.5, 0.5, -0.5]\begin{bmatrix} 10 \\ 20 \\ 30 \\ 40 \end{bmatrix} = \begin{bmatrix} -0.36 \\ -0.48 \\ -0.80 \end{bmatrix}$$

The term $\tilde{V}_r\mathbf{z}$ of Eq. (15.7.3) depends on the two last columns of V, where $\mathbf{z}$ is an arbitrary two-dimensional vector. Thus, the most general least-squares solution is:

$$\mathbf{x} = \begin{bmatrix} -0.36 \\ -0.48 \\ -0.80 \end{bmatrix} + \begin{bmatrix} -0.48 & 0.80 \\ -0.64 & -0.60 \\ 0.60 & 0.00 \end{bmatrix}\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} -0.36 - 0.48z_1 + 0.80z_2 \\ -0.48 - 0.64z_1 - 0.60z_2 \\ -0.80 + 0.60z_1 \end{bmatrix}$$

MATLAB's backslash solution is obtained by fixing z_1, z_2 such that $\mathbf{x}$ will have at most one nonzero entry. For example, demanding that the top two entries be zero, we get:

$$\begin{aligned} -0.36 - 0.48z_1 + 0.80z_2 &= 0 \\ -0.48 - 0.64z_1 - 0.60z_2 &= 0 \end{aligned} \quad\Rightarrow\quad z_1 = -0.75\,, \quad z_2 = 0$$

which gives $-0.8 + 0.6z_1 = -1.25$, and therefore, $\mathbf{x} = [0, 0, -1.25]^T$. This is indeed MATLAB's output of the operation $A\backslash\mathbf{b}$. □

15.8 Condition Number

The condition number of a full-rank $N\times N$ matrix A is given by:

$$\kappa(A) = \|A\|_2\|A^{-1}\|_2 = \frac{\sigma_{\max}}{\sigma_{\min}} \tag{15.8.1}$$

where $\sigma_{\max}, \sigma_{\min}$ are the largest and smallest singular values of A, that is, σ_1, σ_N. The last equation of (15.8.1) follows from $\|A\|_2 = \sigma_1$ and $\|A^{-1}\|_2 = \sigma_N^{-1}$.

The condition number characterizes the sensitivity of the solution of a linear system $A\mathbf{x} = \mathbf{b}$ to small changes in A and $\mathbf{b}$. Taking differentials of both sides of the equation $A\mathbf{x} = \mathbf{b}$, we find:

$$Ad\mathbf{x} + (dA)\mathbf{x} = d\mathbf{b} \quad\Rightarrow\quad d\mathbf{x} = A^{-1}\big[d\mathbf{b} - (dA)\mathbf{x}\big]$$

Taking (the Euclidean) norms of both sides, we have:

$$\|d\mathbf{x}\| \le \|A^{-1}\|\|d\mathbf{b} - (dA)\mathbf{x}\| \le \|A^{-1}\|\big[\|d\mathbf{b}\| + \|dA\|\|\mathbf{x}\|\big]$$

Using the inequality $\|\mathbf{b}\| = \|A\mathbf{x}\| \le \|A\|\|\mathbf{x}\|$, we get:

$$\frac{\|d\mathbf{x}\|}{\|\mathbf{x}\|} \le \kappa(A)\left[\frac{\|dA\|}{\|A\|} + \frac{\|d\mathbf{b}\|}{\|\mathbf{b}\|}\right] \tag{15.8.2}$$

Large condition numbers result in a highly sensitive system, that is, small changes in A and $\mathbf{b}$ may result in very large changes in the solution $\mathbf{x}$. Large condition numbers, $\kappa(A) \gg 1$, imply that $\sigma_1 \gg \sigma_N$, or that A is nearly singular.

Example 15.8.1: Consider the matrix A, which is very close to the singular matrix A_0:

$$A = \begin{bmatrix} 10.0002 & 19.9999 \\ 4.9996 & 10.0002 \end{bmatrix}, \quad A_0 = \begin{bmatrix} 10 & 20 \\ 5 & 10 \end{bmatrix}$$

Its SVD is:

$$A = \begin{bmatrix} \sqrt{0.8} & -\sqrt{0.2} \\ \sqrt{0.2} & \sqrt{0.8} \end{bmatrix} \begin{bmatrix} 25.0000 & 0.0000 \\ 0.0000 & 0.0005 \end{bmatrix} \begin{bmatrix} \sqrt{0.2} & -\sqrt{0.8} \\ \sqrt{0.8} & \sqrt{0.2} \end{bmatrix}^T = U\Sigma V^T$$

Its condition number is $\kappa(A) = \sigma_1/\sigma_2 = 25/0.0005 = 50000$. Computing the solutions of $A\mathbf{x} = \mathbf{b}$ for three slightly different $\mathbf{b}$'s, we find:

$$\mathbf{b}_1 = \begin{bmatrix} 10.00 \\ 5.00 \end{bmatrix} \quad \Rightarrow \quad \mathbf{x}_1 = A\backslash\mathbf{b}_1 = \begin{bmatrix} 0.2 \\ 0.4 \end{bmatrix}$$

$$\mathbf{b}_2 = \begin{bmatrix} 10.00 \\ 5.01 \end{bmatrix} \quad \Rightarrow \quad \mathbf{x}_2 = A\backslash\mathbf{b}_2 = \begin{bmatrix} -15.79992 \\ 8.40016 \end{bmatrix}$$

$$\mathbf{b}_3 = \begin{bmatrix} 10.01 \\ 5.00 \end{bmatrix} \quad \Rightarrow \quad \mathbf{x}_3 = A\backslash\mathbf{b}_3 = \begin{bmatrix} 8.20016 \\ -3.59968 \end{bmatrix}$$

The solutions are exact in the decimal digits shown. Even though the $\mathbf{b}$'s differ only slightly, there are very large differences in the $\mathbf{x}$'s. □

15.9 Reduced-Rank Approximation

The Euclidean and Frobenius matrix norms of an $N\times M$ matrix A of rank r can be expressed conveniently in terms of the singular values of A:

$$\begin{aligned} \|A\|_2 &= \sigma_1 = \text{maximum singular value} \\ \|A\|_F &= (\sigma_1^2 + \sigma_2^2 + \cdots + \sigma_r^2)^{1/2} \end{aligned} \tag{15.9.1}$$

Associated with the SVD expansion (15.5.7), we define a family of reduced-rank matrices A_k obtained by keeping only the first k terms in the expansion:

$$A_k = \sum_{i=1}^{k} \sigma_i \mathbf{u}_i \mathbf{v}_i^T = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^T + \sigma_2 \mathbf{u}_2 \mathbf{v}_2^T + \cdots \sigma_k \mathbf{u}_k \mathbf{v}_k^T\,, \quad k = 1, 2, \dots, r \tag{15.9.2}$$

Clearly, A_k has rank k, and when $k = r$, we have $A_r = A$. In terms of the original full SVD of A, we can write:

$$A_k = U \left[\begin{array}{c|c} \Sigma_k & 0 \\ \hline 0 & 0 \end{array}\right] V^T\,, \quad \Sigma_k = \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_k, \underbrace{0, \dots, 0}_{r-k \text{ zeros}}) \in \mathbb{R}^{r\times r} \tag{15.9.3}$$

Thus, A and A_k agree in their highest k singular values, but the last $r - k$ singular values of A, that is, $\sigma_{k+1}, \dots, \sigma_r$, have been replaced by zeros in A_k. The matrices A_k play a special role in constructing reduced-rank matrices that approximate the original matrix A.

The *reduced-rank approximation theorem* [1234] states that within the set of $N\times M$ matrices of rank k (we assume $k < r$), the matrix B that most closely approximates A in the Euclidean (or the Frobenius) matrix norm is the matrix A_k, that is, the distance $\|A - B\|$ is minimized over the rank-k $N\times M$ matrices when $B = A_k$. The minimized matrix distance is:

$$\begin{aligned} \|A - A_k\|_2 &= \sigma_{k+1} \\ \|A - A_k\|_F &= (\sigma_{k+1}^2 + \cdots + \sigma_r^2)^{1/2} \end{aligned} \tag{15.9.4}$$

This theorem is an essential tool in signal processing, data compression, statistics, principal component analysis, and other applications, such as chaotic dynamics, meteorology, and oceanography.

In remarkably many applications the matrix A has full rank but its singular values tend to cluster into two groups, those that are *large* and those that are *small*, that is, assuming $N \geq M$, we group the M singular values into:

$$\underbrace{\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r}_{\text{large group}} \gg \underbrace{\sigma_{r+1} \geq \cdots \geq \sigma_M}_{\text{small group}} \tag{15.9.5}$$

Fig. 15.9.1 illustrates the typical pattern. A similar pattern arises in the practical determination of the rank of a matrix. To infinite arithmetic precision, a matrix A may have rank r, but to finite precision, the matrix might acquire full rank. However, its lowest $M - r$ singular values are expected to be small.

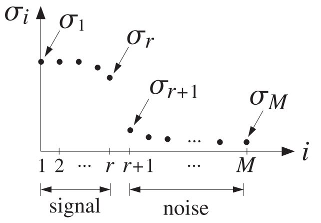


Fig. 15.9.1 Signal subspace vs. noise subspace singular values.

The presence of a significant gap between the large and small singular values allows us to define an *effective* or *numerical rank* for the matrix A.

In least-squares solutions, the presence of small non-zero singular values may cause inaccuracies in the computation of the pseudoinverse. If the last $(M - r)$ small singular values in (15.9.5) are kept, then A^+ would be given by (15.6.7):

$$A^+ = \sum_{i=1}^{r} \frac{1}{\sigma_i} \mathbf{v}_i \mathbf{u}_i^T + \sum_{i=r+1}^{M} \frac{1}{\sigma_i} \mathbf{v}_i \mathbf{u}_i^T$$

and the last $(M - r)$ terms would tend to dominate the expression. For this reason, the rank and the pseudoinverse can be determined with respect to a *threshold level* or tolerance, say, δ such that if $\sigma_i \leq \delta$, for $i = r + 1, \dots, M$, then these singular values may be set to zero and the effective rank will be r. MATLAB's functions **rank** and **pinv** allow the user to specify any desired level of tolerance.

Example 15.9.1: Consider the matrices:

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad \hat{A} = \begin{bmatrix} 0.9990 & -0.0019 & -0.0008 & -0.0004 \\ 0.0037 & 0.9999 & 0.0009 & -0.0005 \\ 0.0008 & -0.0016 & 0.0010 & -0.0002 \\ -0.0007 & 0.0004 & 0.0004 & -0.0006 \end{bmatrix}$$

where the second was obtained by adding small random numbers to the elements of the first using the MATLAB commands:

```
A = zeros(4); A(1,1)=1; A(2,2)=1;          % define the matrix A
Ahat = A + 0.001*randn(size(A));
```

The singular values of the two matrices are:

$$\begin{aligned} \sigma_i &= [1.0000,\ 1.0000,\ 0.0000,\ 0.0000] \\ \hat{\sigma}_i &= [1.0004,\ 0.9984,\ 0.0012,\ 0.0005] \end{aligned}$$

Although A and $\hat{A}$ are very close to each other, and so are the two sets of singular values, the corresponding pseudoinverses differ substantially:

$$A^+ = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad \hat{A}^+ = \begin{bmatrix} 0.9994 & 0.0043 & 1.1867 & -1.0750 \\ -0.0035 & 0.9992 & -0.6850 & -0.5451 \\ -1.1793 & 2.0602 & 1165.3515 & -406.8197 \\ -1.8426 & 1.8990 & 701.5460 & -1795.6280 \end{bmatrix}$$

This would result in completely inaccurate least-squares solutions. For example,

$$\mathbf{b} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} \quad \Rightarrow \quad \mathbf{x} = A^+\mathbf{b} = \begin{bmatrix} 1 \\ 2 \\ 0 \\ 0 \end{bmatrix}, \quad \hat{\mathbf{x}} = \hat{A}^+\mathbf{b} = \begin{bmatrix} 0.2683 \\ -2.2403 \\ 1871.7169 \\ -5075.9187 \end{bmatrix}$$

On the other hand, if we define $\hat{A}^+ = \text{pinv}(A, \delta)$ with a tolerance of $\delta = 10^{-2}$, which amounts to setting $\hat{\sigma}_3 = \hat{\sigma}_4 = 0$, we get acceptable results:

$$\hat{A}^+ = \begin{bmatrix} 1.0010 & 0.0020 & 0.0008 & -0.0007 \\ -0.0037 & 1.0001 & -0.0016 & 0.0004 \\ -0.0008 & 0.0009 & -0.0000 & 0.0000 \\ -0.0004 & -0.0005 & 0.0000 & 0.0000 \end{bmatrix} \quad \Rightarrow \quad \hat{\mathbf{x}} = \hat{A}^+\mathbf{b} = \begin{bmatrix} 1.0043 \\ 1.9934 \\ 0.0010 \\ -0.0014 \end{bmatrix}$$

To avoid such potential pitfalls in solving least squares problems, one may calculate first the singular values of A and then make a decision as to the rank of A. □

In the previous example, we saw that a small change in A caused a small change in the singular values. The following theorem [1236] establishes this property formally. If A and $\hat{A}$ are $N \times M$ matrices with $N \geq M$, then their singular values differ by:

$$\begin{aligned} \max_{1 \leq i \leq M} |\hat{\sigma}_i - \sigma_i| &\leq \|\hat{A} - A\|_2 \\ \sum_{i=1}^{M} |\hat{\sigma}_i - \sigma_i|^2 &\leq \|\hat{A} - A\|_F^2 \end{aligned} \tag{15.9.6}$$

In signal processing applications, we think of the large group of singular values as arising from a desired *signal* or dynamics, and the small group as arising from *noise*. Often, the choice of the r that separates the large from the small group is unambiguous. Sometimes, it is ambiguous and we may need to choose it by trial and error. Replacing the original matrix A by its rank-r approximation tends to reduce the effects of noise and enhance the desired signal.

The construction procedure for the rank-r approximation is as follows. Assuming $N \geq M$ and starting with the *economy* SVD of A, we may partition the singular values according to (15.9.5):

$$A = [U_r \mid \tilde{U}_r] \left[\begin{array}{c|c} \Sigma_r & 0 \\ \hline 0 & \tilde{\Sigma}_r \end{array}\right] \left[\begin{array}{c} V_r^T \\ \hline \tilde{V}_r^T \end{array}\right] = U_r \Sigma_r V_r^T + \tilde{U}_r \tilde{\Sigma}_r \tilde{V}_r^T = A_r + \tilde{A}_r \tag{15.9.7}$$

where $\Sigma_r = \text{diag}(\sigma_1, \dots, \sigma_r)$ and $\tilde{\Sigma}_r = \text{diag}(\sigma_{r+1}, \dots, \sigma_M)$, and we set

$$\begin{aligned} A_r &= [U_r \mid \tilde{U}_r] \left[\begin{array}{c|c} \Sigma_r & 0 \\ \hline 0 & 0 \end{array}\right] \left[\begin{array}{c} V_r^T \\ \hline \tilde{V}_r^T \end{array}\right] = U_r \Sigma_r V_r^T \\ \tilde{A}_r &= [U_r \mid \tilde{U}_r] \left[\begin{array}{c|c} 0 & 0 \\ \hline 0 & \tilde{\Sigma}_r \end{array}\right] \left[\begin{array}{c} V_r^T \\ \hline \tilde{V}_r^T \end{array}\right] = \tilde{U}_r \tilde{\Sigma}_r \tilde{V}_r^T \end{aligned} \tag{15.9.8}$$

where $U_r \in \mathbb{R}^{N \times r}$, $\tilde{U}_r \in \mathbb{R}^{N \times (M-r)}$, $V_r \in \mathbb{R}^{M \times r}$, $\tilde{V}_r \in \mathbb{R}^{M \times (M-r)}$.

We will refer to A_r as the "signal subspace" part of A and to $\tilde{A}_r$ as the "noise subspace" part. The two parts are mutually orthogonal, that is, $A_r^T \tilde{A}_r = 0$. Similarly, Σ_r and $\tilde{\Sigma}_r$ are called the signal subspace and noise subspace singular values.

Example 15.9.2: Consider the following 4×3 matrix:

$$A = \begin{bmatrix} -0.16 & -0.13 & 6.40 \\ 0.08 & 0.19 & -6.40 \\ 3.76 & 4.93 & 1.60 \\ -3.68 & -4.99 & -1.60 \end{bmatrix}$$

Its full SVD is:

$$U\Sigma V^T = \begin{bmatrix} 0.5 & 0.5 & -0.5 & 0.5 \\ -0.5 & -0.5 & -0.5 & 0.5 \\ 0.5 & -0.5 & 0.5 & 0.5 \\ -0.5 & 0.5 & 0.5 & 0.5 \end{bmatrix} \begin{bmatrix} 10 & 0 & 0 \\ 0 & 8 & 0 \\ 0 & 0 & 0.1 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0.36 & -0.48 & 0.80 \\ 0.48 & -0.64 & -0.60 \\ 0.80 & 0.60 & 0.00 \end{bmatrix}^T$$

The economy SVD is:

$$A = \begin{bmatrix} 0.5 & 0.5 & -0.5 \\ -0.5 & -0.5 & -0.5 \\ 0.5 & -0.5 & 0.5 \\ -0.5 & 0.5 & 0.5 \end{bmatrix} \begin{bmatrix} 10 & 0 & 0 \\ 0 & 8 & 0 \\ 0 & 0 & 0.1 \end{bmatrix} \begin{bmatrix} 0.36 & -0.48 & 0.80 \\ 0.48 & -0.64 & -0.60 \\ 0.80 & 0.60 & 0.00 \end{bmatrix}^T$$

The singular values are $\{\sigma_1, \sigma_2, \sigma_3\} = \{10,\ 8,\ 0.1\}$. The first two are "large" and we attribute them to the signal part, whereas the third is "small" and we assume that it is due to noise. The matrix A may be replaced by its rank-2 version by setting $\sigma_3 = 0$. The resulting signal subspace part of A is:

$$A_r = \begin{bmatrix} 0.5 & 0.5 & -0.5 \\ -0.5 & -0.5 & -0.5 \\ 0.5 & -0.5 & 0.5 \\ -0.5 & 0.5 & 0.5 \end{bmatrix} \begin{bmatrix} 10 & 0 & 0 \\ 0 & 8 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0.36 & -0.48 & 0.80 \\ 0.48 & -0.64 & -0.60 \\ 0.80 & 0.60 & 0.00 \end{bmatrix}^T$$

which gives:

$$A_r = \begin{bmatrix} -0.12 & -0.16 & 6.40 \\ 0.12 & 0.16 & -6.40 \\ 3.72 & 4.96 & 1.60 \\ -3.72 & -4.96 & -1.60 \end{bmatrix}$$

The full SVD of A_r, and the one generated by MATLAB are:

$$A_r = \begin{bmatrix} 0.5 & 0.5 & -0.5 & 0.5 \\ -0.5 & -0.5 & -0.5 & 0.5 \\ 0.5 & -0.5 & 0.5 & 0.5 \\ -0.5 & 0.5 & 0.5 & 0.5 \end{bmatrix} \begin{bmatrix} 10 & 0 & 0 \\ 0 & 8 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0.36 & -0.48 & 0.80 \\ 0.48 & -0.64 & -0.60 \\ 0.80 & 0.60 & 0.00 \end{bmatrix}^T$$

$$A_r = \begin{bmatrix} 0.5 & 0.5 & -0.6325 & -0.3162 \\ -0.5 & -0.5 & -0.6325 & -0.3162 \\ 0.5 & -0.5 & -0.3162 & 0.6325 \\ -0.5 & 0.5 & -0.3162 & 0.6325 \end{bmatrix} \begin{bmatrix} 10 & 0 & 0 \\ 0 & 8 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0.36 & -0.48 & 0.80 \\ 0.48 & -0.64 & -0.60 \\ 0.80 & 0.60 & 0.00 \end{bmatrix}^T$$

As usual, the last two columns of the U's are related by a 2×2 orthogonal matrix. □

The OSP MATLAB function **sigsub** constructs both the signal and noise subspace parts of a matrix. It has usage:

```
[As,An] = sigsub(A,r);    % signal + noise susbpaces, r = rank
```

Signal processing methods based on rank reduction are collectively referred to as "SVD signal enhancement methods," or "reduced-rank signal processing methods," or simply, "subspace methods." A number of applications are presented in Refs. [1259–1297]. We will discuss several of these later on.

One of the earliest applications of such methods was in image compression [1296,1297], essentially via the Karhunen-Loève transform. A typical black and white image is represented by a square $N{\times}N$ matrix, where N depends on the resolution, but typical values are $N = 256, 512, 1024$. A color image is represented by three such matrices, one for each primary color (red, green, blue.)

The N singular values of an image matrix drop rapidly to zero. Keeping only the r largest singular values leads to the approximation:

$$A_r = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^T + \sigma_2 \mathbf{u}_2 \mathbf{v}_2^T + \cdots + \sigma_r \mathbf{u}_r \mathbf{v}_r^T$$

Data compression arises because each term in the expansion requires the storage of $2N$ coefficients, that is, N coefficients for each of the vectors $\sigma_i \mathbf{u}_i$ and $\mathbf{v}_i$. Thus, the total number of coefficients to be stored is $2Nr$. Compression takes place as long as this is less than N^2, the total number of matrix elements of the original image. Thus, we require $2Nr < N^2$ or $r < N/2$. In practice, typical values of r that work well are of the order of $N/6$ to $N/5$.

Example 15.9.3: Fig. 15.9.2 shows the singular values of a 512×512 image. They were computed by first removing the column means of the image and then performing a full SVD. The singular values become small after the first 100.

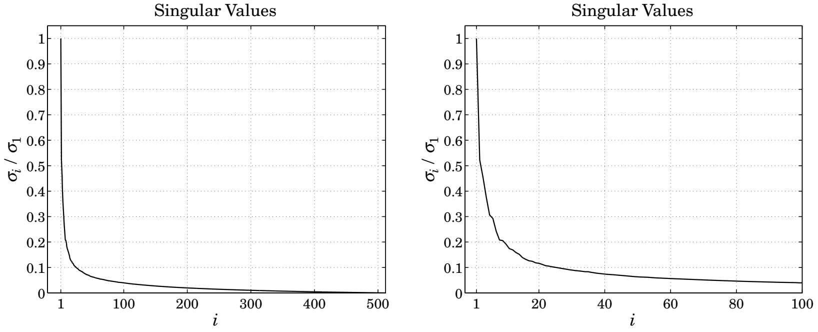


Fig. 15.9.2 Singular values of 512×512 image, with expanded view of first 100 values.

Fig. 15.9.3 shows the original image and the image reconstructed on the basis of the first 100 singular values. The typical MATLAB code was as follows:

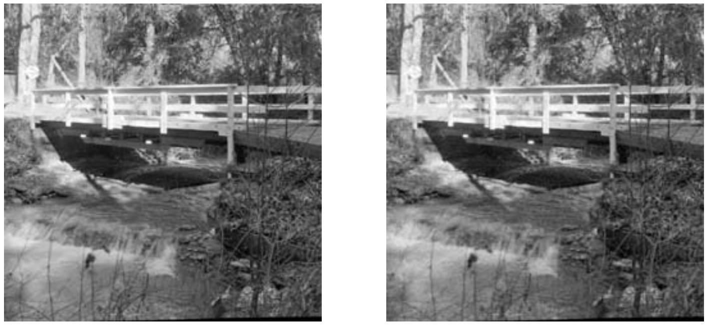


Fig. 15.9.3 Original (left) and compressed images, keeping $r = 100$ components.

```
A = imread('stream.tiff', 'tiff');                  % read image file, size 512×512

[B,M] = zmean(double(A));                           % remove and save mean
[U,S,V] = svd(B);                                   % perform svd

r = 100;
Ar = M + U(:,1:r) * S(1:r,1:r) * V(:,1:r)';         % image from first r components
Ar = uint8(round(Ar));                              % convert to unsigned 8-bit int

figure; image(A);  colormap('gray(256)');           % display image
figure; image(Ar); colormap('gray(256)');
```

The image was obtained from the USC image database [1298]. The function **zmean** removes the mean of each column and saves it. After rank-reduction, the matrix of the means is added back to the image. □

15.10 Regularization of Ill-Conditioned Problems

We saw in the previous section that the presence of small, but nonzero, singular values can cause the least squares solution $\mathbf{x} = A^+\mathbf{b}$ to be highly inaccurate.

Thresholding of the singular values is one of many possible ways to regularize the problem and produce an accurate solution. In all such methods, the true pseudo inverse of $A = U\Sigma V^T$ is replaced by a “filtered” or “regularized” version:

$$\begin{aligned} A^+ &= V\Sigma^+U^T = \sum_{i=1}^{r}\frac{1}{\sigma_i}\mathbf{v}_i\mathbf{u}_i^T && \text{(true)} \\ A_f^+ &= f(A)A^+ = Vf(\Sigma)\Sigma^+U^T = \sum_{i=1}^{r}\frac{f(\sigma_i)}{\sigma_i}\mathbf{v}_i\mathbf{u}_i^T && \text{(regularized)} \end{aligned} \tag{15.10.1}$$

The regularized least-squares solution becomes $\mathbf{x}_f = A_f^+\mathbf{b}$. The function $f(\sigma)$ is chosen so that it is nearly unity for large σ, and $f(\sigma)/\sigma$ is nearly zero for small σ (they may be thought of as highpass filters). Some examples are:

$$\begin{aligned} f(\sigma) &= u(\sigma-\delta) && \text{(thresholding)} \\ f(\sigma) &= \frac{\sigma^2}{\sigma^2+\lambda} && \text{(Tikhonov)} \end{aligned} \tag{15.10.2}$$

where $u(t)$ is the unit-step and $\delta > 0, \lambda > 0$ are positive selectable parameters. The unit-step keeps only those singular values that are above the threshold, $\sigma_i > \delta$. The Tikhonov regularization is explicitly:

$$\mathbf{x}_f = A_f^+\mathbf{b} = \sum_{i=1}^{r}\frac{\sigma_i}{\sigma_i^2+\lambda}\mathbf{v}_i\mathbf{u}_i^T\mathbf{b} \tag{15.10.3}$$

Tikhonov regularization can also be obtained from the following modified least-squares criterion, also known as *ridge regression*,

$$\boxed{\mathcal{J} = \|\mathbf{b}-A\mathbf{x}\|^2 + \lambda\|\mathbf{x}\|^2 = \min} \tag{15.10.4}$$

Indeed, setting the gradient of $\mathcal{J}$ to zero, we find:

$$\frac{\partial\mathcal{J}}{\partial\mathbf{x}} = 2A^T(A\mathbf{x}-\mathbf{b}) + 2\lambda\mathbf{x} = 0 \quad\Rightarrow\quad (A^TA+\lambda I)\mathbf{x} = A^T\mathbf{b}$$

where I is the identity matrix. The solution can be expressed in the form of Eq. (15.10.1). Assuming that A is ill-conditioned but has full rank, then, $A^+ = (A^TA)^{-1}A^T$ (for the case $N \geq M$), so that:

$$\mathbf{x} = (A^TA+\lambda I)^{-1}A^T\mathbf{b} = \big[(A^TA)(A^TA+\lambda I)^{-1}\big](A^TA)^{-1}A^T\mathbf{b} = f(A)A^+\mathbf{b}$$

Regularization is used in many practical inverse problems, such as the deblurring of images or tomography. The second term in the performance index (15.10.4) guards both against ill-conditioning and against noise in the data. If the parameter λ is chosen to be too large, it is possible that noise is removed too much at the expense of getting an accurate inverse. In large-scale inverse problems (e.g., a 512×512 image is represented by a vector $\mathbf{x}$ of dimension $512^2 = 2.6\times10^5$), performing the SVD required in (15.10.1) is not practical and the solution is obtained iteratively, for example, using conjugate-gradients. Regularization can be incorporated into such iterative methods, for example, see Ref. [1236].

Often, the second term in (15.10.4) is replaced by the more general term $\|D\mathbf{x}\|^2 = \mathbf{x}^TD^TD\mathbf{x}$, where D is an appropriate matrix. For example, in an image restoration application, D could be chosen to be a differentiation matrix so that the performance index would attempt to preserve the sharpness of the image. The more general ridge regression performance index and its solution are:

$$\boxed{\mathcal{J} = \|\mathbf{b}-A\mathbf{x}\|^2 + \lambda\|D\mathbf{x}\|^2 = \min} \quad\Rightarrow\quad \boxed{\mathbf{x} = \big(A^TA+\lambda D^TD\big)^{-1}A^T\mathbf{b}} \tag{15.10.5}$$

For example, the Whittaker-Henderson case of Sec. 8.1 corresponds to $A = I$ and D the s-differencing matrix. Another variation of regularization is to assume a decomposition into multiple components of the form, $\mathbf{b} = A_1\mathbf{x}_1 + A_2\mathbf{x}_2$, and impose different regularization constraints on each part, for example, with positive λ_1, λ_2,

$$\mathcal{J} = \|\mathbf{b}-A_1\mathbf{x}_1-A_2\mathbf{x}_2\|^2 + \lambda_1\|D_1\mathbf{x}_1\|^2 + \lambda_2\|D_1\mathbf{x}_2\|^2 = \min \tag{15.10.6}$$

whose minimization with respect to $\mathbf{x}_1, \mathbf{x}_2$, leads to the solution,

$$\begin{bmatrix}\mathbf{x}_1\\ \mathbf{x}_2\end{bmatrix} = \begin{bmatrix} A_1^TA_1+\lambda_1D_1^TD_1 & A_1^TA_2 \\ A_2^TA_1 & A_2^TA+\lambda_2D_2^TD_2\end{bmatrix}^{-1}\begin{bmatrix}A_1^T\mathbf{b}\\ A_2^T\mathbf{b}\end{bmatrix} \tag{15.10.7}$$

An example of such decomposition was the seasonal Whittaker-Henderson case discussed in Sec. 9.9 in which $\mathbf{x}_1$ represented the seasonal component, and $\mathbf{x}_2$, the trend. In addition to the Whittaker-Henderson smoothing methods and their L_2, L_1, and seasonal versions, we previously discussed regularization in the context of Kernel machines in Sec. 8.6, and also in Sec. 12.14 with regard to inverse filtering and deconvolution. Next, we consider sparse regularization.

15.11 Sparse Regularization

Replacing the ℓ_2-norm in the regularization term of Eq. (15.10.5) by the ℓ_p norm leads to the alternative minimization criterion, referred to as *ℓ_p-regularized least-squares*,

$$\mathcal{J} = \|\mathbf{b}-A\mathbf{x}\|_2^2 + \lambda\|D\mathbf{x}\|_p^p = \min \tag{15.11.1}$$

where the first term in (15.11.1) is still the ℓ_2 norm of the modeling error, $\mathbf{b} - A\mathbf{x}$, and $\|\mathbf{x}\|_p$ denotes the ℓ_p norm of the vector $\mathbf{x} = [x_1, x_2, \cdots, x_M]^T$,

$$\|\mathbf{x}\|_p = \left[\sum_{n=1}^{M}|x_n|^p\right]^{\frac{1}{p}} \quad\Rightarrow\quad \|\mathbf{x}\|_p^p = \sum_{n=1}^{M}|x_n|^p$$

Such criteria have been studied very extensively in inverse problems, with renewed interest in the past 15 years in sparse modeling, statistical learning, and compressive sensing applications. Even though $\|\mathbf{x}\|_p$ is a proper norm only for $p \geq 1$, the cases $0 \leq p \leq 1$ have also been considered widely because they promote the sparsity of the resulting solution vector $\mathbf{x}$, or rather, the sparsity of the vector $D\mathbf{x}$ in (15.11.1). In particular, the case $p = 1$ is unique for the following reasons: (a) it corresponds to the smallest possible proper norm, (b) it typically results in a sparse solution, which under many circumstances is close to, or coincides with, the sparsest solution, and (c) the minimization problem (15.11.1) is a convex optimization problem for which there are efficient numerical methods.

We concentrate below on the three cases $p = 0, 1, 2$, and also set $D = I$ for now, and consider the following three optimization criteria for solving the linear system $A\mathbf{x} = \mathbf{b}$, with $A \in \mathbb{R}^{N\times M}$, $\mathbf{b} \in \mathbb{R}^N$, and, $\mathbf{x} \in \mathbb{R}^M$,

$$\begin{aligned}
(L_0):&\quad \mathcal{J} = \|\mathbf{b} - A\mathbf{x}\|_2^2 + \lambda\,\|\mathbf{x}\|_0 = \min \\
(L_1):&\quad \mathcal{J} = \|\mathbf{b} - A\mathbf{x}\|_2^2 + \lambda\,\|\mathbf{x}\|_1 = \min \\
(L_2):&\quad \mathcal{J} = \|\mathbf{b} - A\mathbf{x}\|_2^2 + \lambda\,\|\mathbf{x}\|_2^2 = \min
\end{aligned} \qquad (15.11.2)$$

where the ℓ_0 norm, $\|\mathbf{x}\|_0$, is the cardinality of the vector $\mathbf{x}$, that is, the number of its non-zero entries. The criteria try to minimize the corresponding norm of $\mathbf{x}$ while being consistent with the given linear system. Criterion (L_0) results in the sparsest solution but is essentially intractable. Criterion (L_1) is used as an alternative to (L_0) and results also in a sparse solution. It is known as the *lasso*[†] [531], or as *basis pursuit denoising* [534], or as *ℓ_1-regularized least squares.*

There is a vast literature on the properties, applications, and numerical methods of the above criteria. A small and incomplete set of references is [479–589]. A comprehensive review is [557]. Several MATLAB-based packages are also available [590].

Below we discuss two examples that illustrate the sparsity of the resulting solutions: (i) an overdetermined sparse spike deconvolution problem, and (ii) an underdetermined sparse signal recovery example. In these examples, the (L_0) problem is solved with an iteratively re-weighted least-squares (IRLS) method, and the (L_1) problem, with the CVX package[‡] as well as with the IRLS method for comparison.

We introduced the IRLS method in the context of sparse Whittaker-Henderson smoothing, or, ℓ_1-trend filtering, in Sec. 8.7. There are several variants of this method, [520–528,532,553,557,560,565,566], but the basic idea is to replace the ℓ_p norm with a weighted ℓ_2 norm, which can be solved iteratively. We recall from Sec. 8.7 that given a real number $0 \leq p \leq 2$, set $q = 2 - p$, and write for any real number $x \neq 0$,

$$|x|^p = \frac{|x|^2}{|x|^q} \approx \frac{|x|^2}{|x|^q + \varepsilon}$$

[†] Least Absolute Shrinkage and Selection Operator

[‡] `http://cvxr.com/cvx/`

where ε is a sufficiently small positive number needed to also allow the case $x = 0$. Similarly, we can write for the ℓ_p-norm of a vector $\mathbf{x} \in \mathbb{R}^M$,

$$\begin{aligned}
\|\mathbf{x}\|_p^p &= \sum_{i=1}^{M} |x_i|^p \approx \sum_{i=1}^{M} \frac{|x_i|^2}{|x_i|^q + \varepsilon} = \mathbf{x}^T W(\mathbf{x})\mathbf{x} \\
W(\mathbf{x}) &= \text{diag}\left[\frac{1}{|\mathbf{x}|^q + \varepsilon}\right] = \text{diag}\left[\frac{1}{|x_1|^q + \varepsilon}\,,\; \frac{1}{|x_2|^q + \varepsilon}\,,\; \dots,\; \frac{1}{|x_M|^q + \varepsilon}\right]
\end{aligned} \qquad (15.11.3)$$

Alternatively, one can define $W(\mathbf{x})$ as the pseudo-inverse of the diagonal matrix of the powers $|x_i|^q$, $i = 1, 2, \dots, M$, that is, in MATLAB language,[†]

$$W(\mathbf{x}) = \text{pinv}\left(\text{diag}\big[|x_1|^q\,,\; |x_2|^q\,,\; \dots,\; |x_M|^q\big]\right) \qquad (15.11.4)$$

Then, the ℓ_p-regularized least-squares problem can be written in the form,

$$\mathcal{J} = \|\mathbf{b} - A\mathbf{x}\|_2^2 + \lambda\,\|\mathbf{x}\|_p^p = \|\mathbf{b} - A\mathbf{x}\|_2^2 + \lambda\,\mathbf{x}^T W(\mathbf{x})\,\mathbf{x} = \min \qquad (15.11.5)$$

This approximation leads to the following iterative solution in which the diagonal weighting matrix W to be used in the the next iteration is replaced by its value from the previous iteration,

$$\boxed{\begin{aligned}
&\text{for } k = 1, 2, \dots, K, \text{ do:} \\
&\qquad W_{k-1} = W(\mathbf{x}^{(k-1)}) \\
&\qquad \mathbf{x}^{(k)} = \arg\min_{\mathbf{x}} \|\mathbf{b} - A\mathbf{x}\|_2^2 + \lambda\,\mathbf{x}^T W_{k-1}\mathbf{x}
\end{aligned}} \quad \text{(IRLS)} \qquad (15.11.6)$$

with the algorithm initialized to the ordinary least-squares solution of criterion (L_2):

$$\mathbf{x}^{(0)} = (\lambda\,I + A^T A)^{-1} A^T \mathbf{b}$$

The solution of the optimization problem in (15.11.6) at the kth step is:

$$\mathbf{x}^{(k)} = (\lambda\,W_{k-1} + A^T A)^{-1} A^T \mathbf{b}$$

Thus, the choices $p = 0$ and $p = 1$ should resemble the solutions of the ℓ_0 and ℓ_1 regularized problems. The IRLS algorithm (15.11.6) works well for moderate-sized problems ($N, M < 1000$). For large-scale problems ($N, M > 10^6$), the successive least-squares problems could be solved with more efficient methods, such as conjugate gradients.

The general case of (15.11.1) that includes the smoothness-constraining matrix D can also be handled in the same way. Following the discussion of Sec. 8.7, we can write,

$$\mathcal{J} = ||\mathbf{b} - A\mathbf{x}||_2^2 + \lambda||D\mathbf{x}||_p^p = ||\mathbf{b} - A\mathbf{x}||_2^2 + \lambda\,\mathbf{x}^T D^T W(D\mathbf{x})\,D\mathbf{x} = \min \qquad (15.11.7)$$

which leads to to the following iterative algorithm,

$$\boxed{\begin{aligned}
&\text{for } k = 1, 2, \dots, K, \text{ do:} \\
&\qquad W_{k-1} = W(D\mathbf{x}^{(k-1)}) \\
&\qquad \mathbf{x}^{(k)} = \arg\min_{\mathbf{x}} ||\mathbf{b} - A\mathbf{x}||_2^2 + \lambda\,\mathbf{x}^T D^T W_{k-1} D\mathbf{x}
\end{aligned}} \quad \text{(IRLS)} \qquad (15.11.8)$$

[†] e.g., `pinv(diag([2, 0, 4]))` produces the diagonal matrix, `diag([0.50, 0, 0.25])`.

with the algorithm initialized to the ordinary least-squares solution:

$$\mathbf{x}^{(0)} = \big(A^TA + \lambda\, D^TD\big)^{-1}A^T\mathbf{b}$$

The solution of the optimization problem at the kth step is:

$$\mathbf{x}^{(k)} = \big(A^TA + \lambda\, D^TW_{k-1}D\big)^{-1}A^T\mathbf{b}$$

Sparse Spike Deconvolution Example

Consider a deconvolution problem in which the observed signal y_n is the noisy convolution, $y_n = h_n * s_n + v_n$, where v_n is zero-mean white noise of variance σ_v^2. The objective is to recover the signal s_n assuming knowledge of the filter h_n. For an FIR filter of order M and input of length L, the output will have length $N = L + M$, and we may cast the above convolutional filtering equation in the matrix form:

$$\mathbf{y} = H\mathbf{s} + \mathbf{v}$$

where $\mathbf{y}, \mathbf{v} \in \mathbb{R}^N$, $\mathbf{s} \in \mathbb{R}^L$, and H is the $N\times L$ convolution matrix corresponding to the filter. It can be constructed as a sparse matrix by the function:

```
H = convmat(h,L);    % H = convmtx(h,N) = non-sparse version
```

The filter is taken to be:

$$h_n = \cos\big(0.15\,(n - n_0)\big)\,\exp\big(-0.004\,(n - n_0)^2\big)\,, \qquad n = 0, 1, \dots, M$$

where $M = 53$ and $n_0 = 25$. The input is a sparse spike train consisting of S spikes:

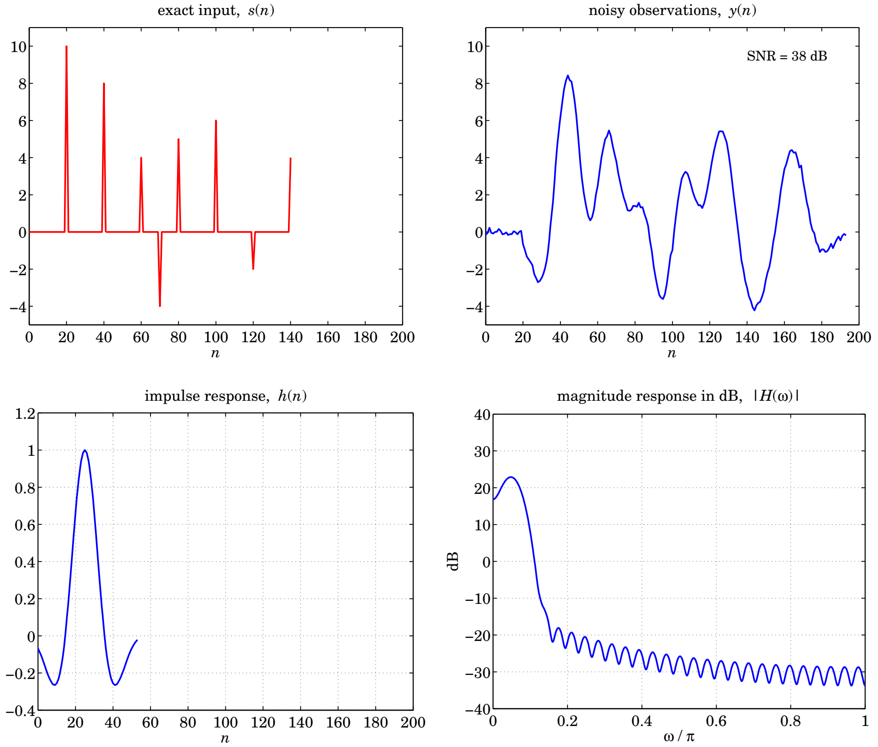
$$s_n = \sum_{i=1}^{S} a_i\delta(n - n_i)\,, \qquad n = 0, 1, \dots, L - 1 \tag{15.11.9}$$

where $S = 8$ and the spike locations and amplitudes are given as follows:

$$n_i = [20,\ 40,\ 60,\ 70,\ 80,\ 100,\ 120,\ 140]\,, \qquad a_i = [10,\ 8,\ 4,\ -4,\ 5,\ 6,\ -2,\ 4]$$

The input signal length is defined from the last spike location to be $L = n_8 + 1 = 141$. The noise standard deviation is chosen to be $\sigma_v = 0.1$, which corresponds to approximately 38 dB signal-to-noise ratio, that is, $SNR = 20\log_{10}\big(\max|H\mathbf{s}|\,/\,\sigma_v\big) = 38$.

The input signal s_n and the convolved noisy signal y_n are shown below. Also shown are the impulse response h_n and the corresponding magnitude response $|H(\omega)|$ plotted in dB versus $0 \le \omega \le \pi$ rads/sample.

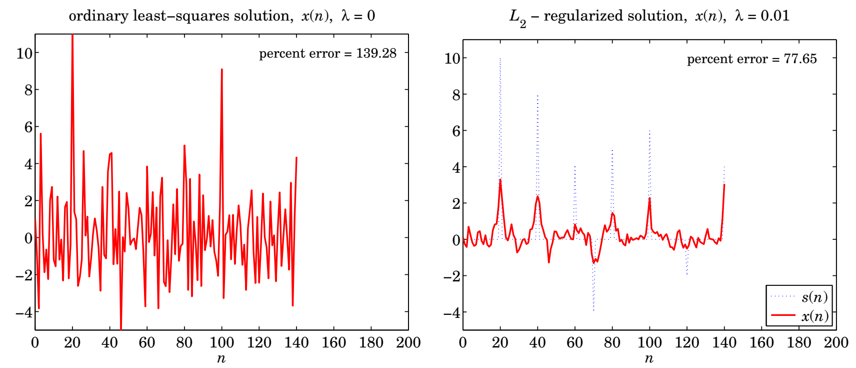


We note that $H(\omega)$ occupies a low frequency band, thus, we expect the effective deconvolution inverse filtering operation by $1/H(\omega)$ to be very sensitive to even small amounts of noise in y_n, even though the noise is barely visible in y_n itself. The three criteria of Eq. (15.11.2) to be implemented are,

$$\begin{aligned}
\mathcal{J} &= \|\mathbf{y} - H\mathbf{x}\|_2^2 + \lambda\,\|\mathbf{x}\|_0 = \min\\
\mathcal{J} &= \|\mathbf{y} - H\mathbf{x}\|_2^2 + \lambda\,\|\mathbf{x}\|_1 = \min\\
\mathcal{J} &= \|\mathbf{y} - H\mathbf{x}\|_2^2 + \lambda\,\|\mathbf{x}\|_2^2 = \min
\end{aligned} \tag{15.11.10}$$

The ℓ_2 case with $\lambda = 0$ corresponds to the ordinary (full-rank overdetermined) least-squares solution of the linear system, $H\mathbf{x} = \mathbf{y}$, that is, $\mathbf{x}_{\text{ord}} = (H^TH)^{-1}H^T\mathbf{y}$, or, $\mathbf{x}_{\text{ord}} = H\backslash\mathbf{y}$, in MATLAB.

Similarly, the ℓ_2-regularized solution with non-zero λ is, $\mathbf{x}_2 = (\lambda I + H^TH)^{-1}H^T\mathbf{y}$. These two solutions are depicted below, displaying also the percent error of recovering the desired signal $\mathbf{s}$, defined in terms of the ℓ_2 norms by, $P_{\text{error}} = 100\cdot\|\mathbf{x} - \mathbf{s}\|_2\,/\,\|\mathbf{s}\|_2$.



The MATLAB code for generating the above six graphs was as follows:

```
g = @(x) cos(0.15*x).*exp(-0.004*x.^2);     % filter function
delta = @(x) (x==0);

M = 53; n0 = 25; k = (0:M)'; h = g(k-n0);   % filter h(n)

ni = [20  40  60  70  80  100  120  140];   % spike locations & amplitudes
ai = [10  8   4   -4  5    6    -2   4 ];

L = ni(end)+1; N = M+L;                 % L = 141, N = 194
n = (0:L-1)'; t = (0:N-1)';             % time indices for s(n) and y(n)

s = 0;
for i=1:length(ni),                     % exact input s(n)
  s = s + ai(i) * delta(n-ni(i));
end

H = convmat(h,L);                       % NxL=194x141 convolution matrix

sigma = 0.1;
seed = 2017; randn('state',seed);       % initialize generator

y = H*s + sigma * randn(N,1);           % noisy observations y(n)

w = linspace(0,1,1001)*pi;              % frequencies in rads/sample
Hmag = 20*log10(abs(dtft(h,w)));        % can also use freqz(h,1,w)

xord = H\y;                             % ordinary least-squares
Perr = 100 * norm(s-xord)/norm(s);

la = 0.01;
x2 = (la * eye(L) + H'*H) \ (H'*y);     % L2-regularized
Perr = 100 * norm(s-x2)/norm(s);

figure; plot(n,s);    figure; plot(t,y);          % plot s(n) and y(n)
figure; plot(k,h);    figure; plot(w/pi,Hmag);    % plot h(n) and H(w)
figure; plot(n,xord); figure; plot(n,x2);         % plot xord(n) and x2(n)
```

As expected from the lowpass nature of $H(\omega)$, the ordinary least-squares solution is too noisy to be useful, while the regularized one is only slightly better. The effect

of increasing λ is to smooth the noise further, but at the expense of flattening and broadening the true spikes (for example, try the value, $\lambda = 0.1$).

To understand this behavior from the frequency point of view, let us pretend that the signals y_n, x_n are infinitely long. Following the approach of Sec. 8.2, we may replace the (L_2) criterion in Eq. (15.11.10) by the following,

$$\begin{aligned}\mathcal{J} &= \sum_{n=-\infty}^{\infty} |y_n - h_n * x_n|^2 + \lambda \sum_{n=-\infty}^{\infty} |x_n|^2 = \\ &= \int_{-\pi}^{\pi} |Y(\omega) - H(\omega)X(\omega)|^2 \frac{d\omega}{2\pi} + \lambda \int_{-\pi}^{\pi} |X(\omega)|^2 \frac{d\omega}{2\pi} = \min\end{aligned} \tag{15.11.11}$$

where we used Parseval's identity. The vanishing of the functional derivative of $\mathcal{J}$ with respect to $X^*(\omega)$, then leads to the following regularized inverse filtering solution,

$$\frac{\delta\mathcal{J}}{\delta X^*(\omega)} = |H(\omega)|^2 X(\omega) - H^*(\omega)Y(\omega) + \lambda X(\omega) = 0\,, \quad \text{or,} \tag{15.11.12}$$

$$\boxed{X(\omega) = \frac{H^*(\omega)}{\lambda + |H(\omega)|^2}\, Y(\omega)} \quad \text{(regularized inverse filter)} \tag{15.11.13}$$

If we express $Y(\omega)$ in terms of the spectrum $S(\omega)$ of the desired signal and the spectrum $V(\omega)$ of the added noise, then, Eq. (15.11.13) leads to,

$$Y(\omega) = H(\omega)S(\omega) + V(\omega) \quad \Rightarrow \quad X(\omega) = \frac{|H(\omega)|^2}{\lambda + |H(\omega)|^2} S(\omega) + \frac{H^*(\omega)}{\lambda + |H(\omega)|^2} V(\omega)$$

For $\lambda = 0$, this becomes the ordinary inverse filter,

$$X(\omega) = \frac{1}{H(\omega)} Y(\omega) = S(\omega) + \frac{1}{H(\omega)} V(\omega)$$

which, although it recovers the $S(\omega)$ term, it greatly amplifies the portions of the white-noise spectrum that lie in the stopband of the filter, that is where, $H(\omega) \approx 0$. For $\lambda \neq 0$ on the other hand, the regularization filter acts as a lowpass filter, becoming vanishingly small over the stopband, and hence removing some of the noise, but also smoothing and broadening the spikes for the same reason, that is, removing some of the high-frequencies in $S(\omega)$.

By contrast, the ℓ_0 and ℓ_1 regularized criteria of Eq. (15.11.10) behave dramatically differently and are capable of accurately extracting the input spikes, as seen in the graphs of Fig. 15.11.1.

The ℓ_1 case was computed with the CVX package, as well as with the IRLS algorithm of Eq. (15.11.6), with the parameter values, $\lambda = 0.1$, $p = 1$, $q = 1$, $\epsilon = 10^{-5}$, and $K = 100$ iterations.

The ℓ_0 case was computed with the IRLS algorithm using parameters, $\lambda = 0.1$, $p = 0$, $q = 2$, $\epsilon = 10^{-5}$, and $K = 100$ iterations—however, it actually converges within about 10 iterations as seen in the bottom-right graph that plots the iteration percentage error defined at the kth iteration by, $P(k) = 100 \cdot \|\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\|_2 / \|\mathbf{x}^{(k-1)}\|_2$.

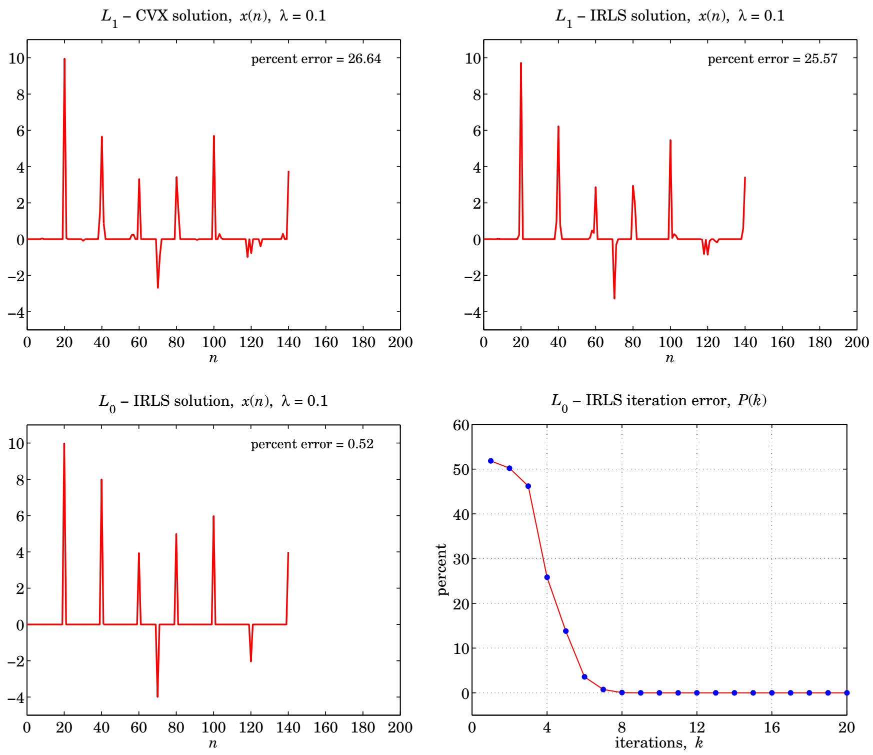


Fig. 15.11.1 Deconvolved signals based on the ℓ_1 and ℓ_0 criteria.

The recovered signal in the ℓ_0 case is slightly sparser than that of the ℓ_1 case, as is seen in the figures, or by evaluating the reconstruction error, $P_{\text{error}} = 100\cdot\|\mathbf{x}-\mathbf{s}\|_2\,/\,\|\mathbf{s}\|_2$, but both versions fairly accurately extract the spike amplitudes and locations. The MATLAB code used to produce these four graphs was as follows.

```
la = 0.1;

cvx_begin                                 % L1 case - CVX solution
   variable x(L)
   minimize( sum_square(H*x-y) + la * norm(x,1) )
cvx_end

Perr = 100*norm(x-s)/norm(s);             % reconstruction error

figure; plot(n,x,’r-’);                   % plot L1 - CVX version

% -------------------------------------------------------------------

p=1; q=2-p; epsilon=1e-5; K=100;          % L1 case - IRLS solution
W = speye(L);
```

```
x0 = (la * W + H’*H) \ (H’*y);

for k=1:K,
   W = diag(1./(abs(x0).^q + epsilon));
   x = (la * W + H’*H) \ (H’*y);
   P(k) = norm(x-x0)*100/norm(x0);
   x0 = x;
end

Perr = 100*norm(x-s)/norm(s);             % reconstruction error

figure; plot(n,x,’r-’);                   % plot L1 - IRLS version

% -------------------------------------------------------------------

p=0; q=2-p; epsilon=1e-5; K=100;          % L0 case - IRLS solution
W = speye(L);

x0 = (la * W + H’*H) \ (H’*y);            % initialize iteration

for k=1:K,                                % IRLS iteration
   W = diag(1./(abs(x0).^q + epsilon));
   x = (la * W + H’*H) \ (H’*y);
   P(k) = 100*norm(x-x0)/norm(x0);        % iteration error
   x0 = x;
end

Perr = 100*norm(x-s)/norm(s);             % reconstruction error

figure; plot(n,x,’r-’);                   % plot L0 - IRLS version
k = 1:K;
figure; plot(k,P,’r-’, k,P,’b.’);         % plot iteration error P(k)
```

Sparse Signal Recovery Example

In this example, based on [544], we consider the underdetermined noisy linear system:

$$\mathbf{y} = A\mathbf{s} + \mathbf{v}$$

where $A \in \mathbb{R}^{1000\times 2000}$, $\mathbf{s} \in \mathbb{R}^{2000}$, and $\mathbf{y},\mathbf{v} \in \mathbb{R}^{1000}$. The matrix A has full rank and consists of zero-mean, unit-variance, gaussian, independent random entries, and the 2000-long input signal $\mathbf{s}$ is sparse with only $L = 100$ non-zero entries taken to be randomly positioned within its length, and constructed to have amplitudes ± 1 with random signs and then weighted by a triangular window in order to get a variety of values.

The noise $\mathbf{v}$ is zero-mean gaussian white noise with standard deviation $\sigma_v = 0.1$. The recovery criteria are as in Eq. (15.11.2),

$$\begin{aligned}
\mathcal{J} &= \|\mathbf{y}-A\mathbf{x}\|_2^2 + \lambda\,\|\mathbf{x}\|_0 = \min\\
\mathcal{J} &= \|\mathbf{y}-A\mathbf{x}\|_2^2 + \lambda\,\|\mathbf{x}\|_1 = \min\\
\mathcal{J} &= \|\mathbf{y}-A\mathbf{x}\|_2^2 + \lambda\,\|\mathbf{x}\|_2^2 = \min
\end{aligned}\tag{15.11.14}$$

Fig. 15.11.2 shows the signal $s(n)$ and the observations $y(n)$, as well as the recovered signals $x(n)$ based on the above criteria. The ℓ_1 solution was computed with the CVX package and the IRLS algorithm, and the ℓ_0 solution, with the IRLS algorithm. The parameter λ was chosen to be $\lambda = 0.1$ in the ℓ_1 and ℓ_0 cases, and $\lambda = 0$ for the ℓ_2 case, which corresponds to the usual minimum-norm solution of the underdetermined linear system $A\mathbf{x} = \mathbf{y}$, that is, $\mathbf{x} = A^+\mathbf{y} = A^T(AA^T)^{-1}\mathbf{y}$, in terms of the pseudo-inverse of A. Note that using $\lambda = 0.1$ in the ℓ_2 case is virtually indistinguishable from the $\lambda = 0$ case.

The ℓ_2 criterion does not produce an acceptable solution. But both the ℓ_1 and the ℓ_0 criteria accurately recover the sparse signal $s(n)$, with the ℓ_0 solution being somewhat sparser and resulting in smaller recovery error, $P_{\text{error}} = 100\cdot\|\mathbf{x}-\mathbf{s}\|_2\,/\,\|\mathbf{s}\|_2$.

The IRLS algorithms were run with parameters $\lambda = 0.1$, $\epsilon = 10^{-6}$, and $K = 20$ iterations. The successive iteration percentage errors, $P(k) = 100\cdot\|\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\|_2\,/\,\|\mathbf{x}^{(k-1)}\|_2$, are plotted versus k in Fig. 15.11.3 for the ℓ_1 and ℓ_0 cases. The MATLAB code used to produce the solutions and graphs is given below.

```
N = 1000; M = 2000; L = 100;               % L-sparse

seed = 1000;                               % initialize generators
randn('state',seed);
rand('state',seed);

A = randn(N,M);                            % random NxM matrix
s = zeros(M,1);

I = randperm(M); I = I(1:L);               % L random indices in 1:M
s(I) = sign(randn(L,1));                   % L random signs at locations I

t = (0:N-1)'; n = (0:M-1)';
w = 1 - abs(2*n-M+1)/(M-1);                % triangular window
s = s .* w;                                % L-sparse windowed input

sigma = 0.1;
v = sigma * randn(N,1);
y = A*s + v;                               % noisy observations

SNR = 20*log10(norm(A*s,Inf)/sigma);       % SNR = 45 dB

figure; stem(n,s); figure; stem(t,y);      % plot s(n) and y(n)

% -----------------------------------

rank(A);                                   % verify full rank = 1000

x = pinv(A)*y;                             % L2 - minimum-norm solution

Perr = 100 * norm(x-s)/norm(s);            % reconstruction error = 71.21 %

figure; stem(n,x,'r-');

% -----------------------------------

la = 0.1;

cvx_begin                                  % L1 - CVX solution
   variable x(M)
   minimize( sum_square(A*x-y) + la * norm(x,1) )
cvx_end
```

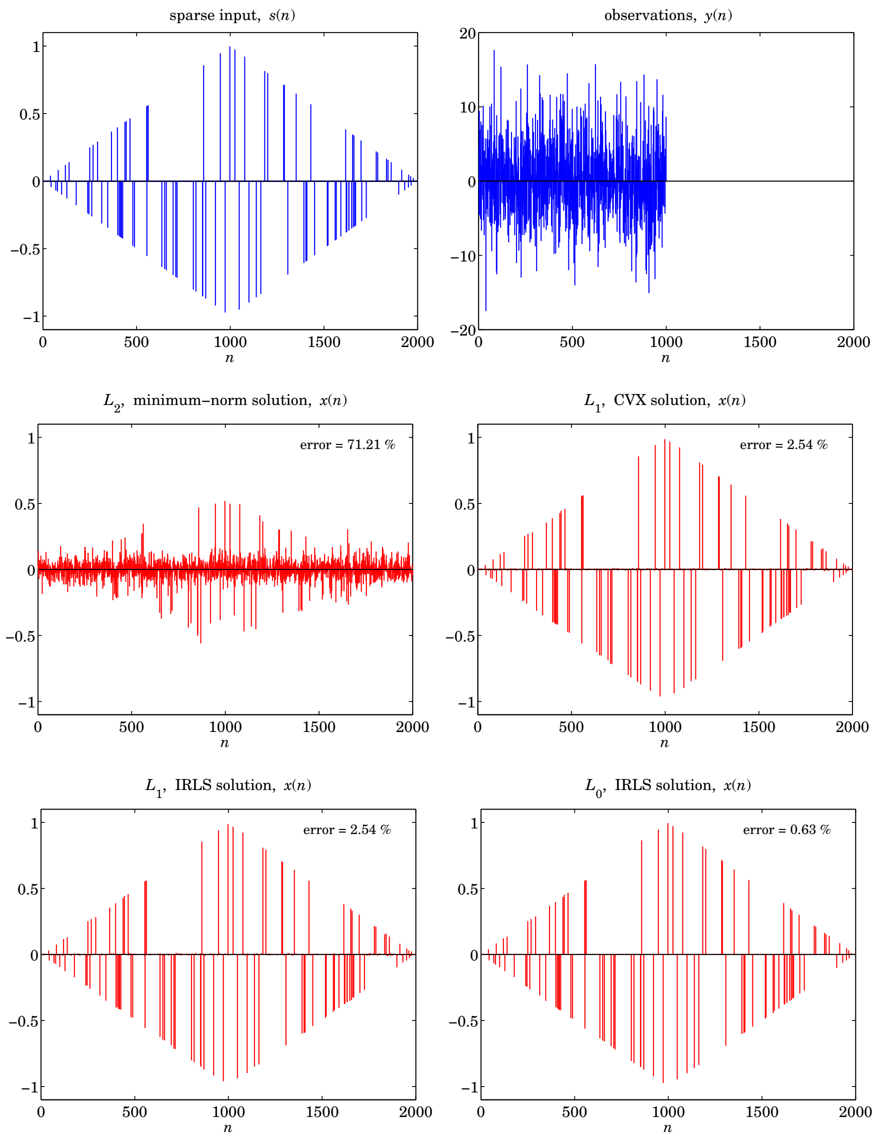


Fig. 15.11.2 Recovered signals based on the ℓ_2, ℓ_1, and ℓ_0 criteria.

```
Perr = 100 * norm(x-s)/norm(s);            % reconstruction error = 2.54 %

figure; stem(n,x,'r-');
```

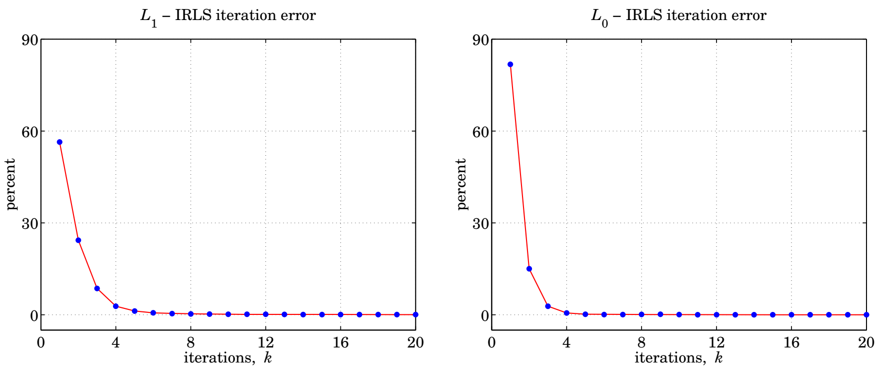


Fig. 15.11.3 IRLS iteration error based on the ℓ_1, and ℓ_0 criteria.

```
% ------------------------------------

p = 1; q = 2-p; epsilon = 1e-6; K = 20;
W = speye(M);

ATA = A'*A;
ATy = A'*y;

x0 = (la*W + ATA) \ ATy;

for k=1:K,                                   % L1 - IRLS solution
   W = diag(1./(abs(x0).^q + epsilon));
   x = (la*W + ATA) \ ATy;
   P(k) = norm(x-x0)*100/norm(x0);
   x0 = x;
end

Perr = 100 * norm(x-s)/norm(s);              % reconstruction error = 2.54 %

figure; stem(n,x,'r-');

k = 1:K;
figure; plot(k,P,'r-', k,P,'b.');            % iteration error

% ------------------------------------

p = 0; q = 2-p; epsilon = 1e-6; K = 20;
W = speye(M);

x0 = (la*W + ATA) \ ATy;

for k=1:K,                                   % L0 - IRLS solution
   W = diag(1./(abs(x0).^q + epsilon));
   x = (la*W + ATA) \ ATy;
   P(k) = norm(x-x0)*100/norm(x0);
   x0 = x;
```

```
end

Perr = 100 * norm(x-s)/norm(s);              % reconstruction error = 0.63 %

figure; stem(n,x,'r-');

k = 1:K;
figure; plot(k,P,'r-', k,P,'b.');            % iteration error
```

15.12 SVD and Signal Processing

In many signal processing applications, such as Wiener filtering and linear prediction, the SVD appears naturally in the context of solving the normal equations.

The optimum order-M Wiener filter for estimating a signal $x(n)$ on the basis of the signals $\{y_0(n), y_1(n), \dots, y_M(n)\}$ satisfies the normal equations:

$$R\mathbf{h} = \mathbf{r}\,, \quad \text{where} \quad R = E[\mathbf{y}^*(n)\mathbf{y}^T(n)]\,, \quad \mathbf{r} = E[x(n)\mathbf{y}^*(n)] \tag{15.12.1}$$

where we assumed stationarity and complex-valued signals. The optimum estimate of $x(n)$ is given by the linear combination:

$$\hat{x}(n) = \mathbf{h}^T\mathbf{y}(n) = [h_0, h_1, \dots, h_M]\begin{bmatrix} y_0(n) \\ y_1(n) \\ \vdots \\ y_M(n) \end{bmatrix} = \sum_{m=0}^{M} h_m y_m(n) \tag{15.12.2}$$

The observation signals $y_m(n)$ are typically (but not necessarily) either the outputs of a tapped delay line whose input is a *single* time signal y_n, so that $y_m(n) = y_{n-m}$, or, alternatively, they are the outputs of an antenna (or other spatial sensor) array. The two cases are shown in Fig. 15.12.1.

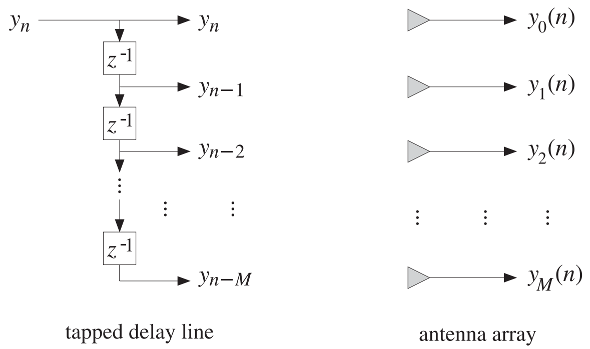


Fig. 15.12.1 Time-series processing versus spatial processing

The vector $\mathbf{y}(n)$ is defined as:

$$\mathbf{y}(n)=\begin{bmatrix} y_n \\ y_{n-1} \\ y_{n-2} \\ \vdots \\ y_{n-M} \end{bmatrix}, \quad \text{or,} \quad \mathbf{y}(n)=\begin{bmatrix} y_0(n) \\ y_1(n) \\ y_2(n) \\ \vdots \\ y_M(n) \end{bmatrix} \tag{15.12.3}$$

In the array case, $\mathbf{y}(n)$ is called a *snapshot* vector because it represents the measurement of the wave field across the array at the nth time instant. The autocorrelation matrix R measures *spatial correlations* among the antenna elements, that is, $R_{ij} = E[y_i^*(n)y_j(n)]$, $i,j, = 0,1,\dots,M$.

In the time-series case, R measures *temporal correlations* between successive samples of y_n, that is, $R_{ij} = E[y_{n-i}^* y_{n-j}] = E[y_{n+i-j}y_n^*] = R(i-j)$, where we used the stationarity assumption to shift the time indices and defined the autocorrelation function of y_n by:

$$R(k) = E[y_{n+k}y_n^*] \tag{15.12.4}$$

The normal equations are derived from the requirement that the optimum weights $\mathbf{h} = [h_0, h_1, \dots, h_M]^T$ minimize the mean-square estimation error:

$$\mathcal{E} = E[|e(n)|^2] = E[|x(n) - \hat{x}(n)|^2] = E[|x(n) - \mathbf{h}^T\mathbf{y}(n)|^2] = \min \tag{15.12.5}$$

The minimization condition is equivalent to the orthogonality equations, which are equivalent to the normal equations:

$$E[e(n)\mathbf{y}^*(n)] = 0 \quad \Leftrightarrow \quad E[\mathbf{y}^*(n)\mathbf{y}^T(n)]\mathbf{h} = E[x(n)\mathbf{y}^*(n)] \tag{15.12.6}$$

Setting $R = E[\mathbf{y}^*(n)\mathbf{y}^T(n)]$ and $\mathbf{r} = E[x(n)\mathbf{y}^*(n)]$, we find for the optimum weights and the optimum estimate of $x(n)$:

$$\begin{aligned} \mathbf{h} &= E[\mathbf{y}^*(n)\mathbf{y}^T(n)]^{-1}E[x(n)\mathbf{y}^*(n)] = R^{-1}\mathbf{r} \\ \hat{x}(n) &= \mathbf{h}^T\mathbf{y}(n) = E[x(n)\mathbf{y}^\dagger(n)]E[\mathbf{y}(n)\mathbf{y}^\dagger(n)]^{-1}\mathbf{y}(n) \end{aligned} \tag{15.12.7}$$

In practice, we may replace the above statistical expectation values by time-averages based on a finite, but stationary, set of time samples of the signals $x(n)$ and $\mathbf{y}(n)$, $n = 0,1,\dots,N-1$, where typically $N > M$. Thus, we make the replacements:

$$\begin{aligned} R = E[\mathbf{y}^*(n)\mathbf{y}^T(n)] \quad &\Rightarrow \quad \hat{R} = \frac{1}{N}\sum_{n=0}^{N-1}\mathbf{y}^*(n)\mathbf{y}^T(n) \\ \mathbf{r} = E[\mathbf{y}^*(n)x(n)] \quad &\Rightarrow \quad \hat{\mathbf{r}} = \frac{1}{N}\sum_{n=0}^{N-1}\mathbf{y}^*(n)x(n) \\ E[\mathbf{y}^*(n)e(n)] = 0 \quad &\Rightarrow \quad \frac{1}{N}\sum_{n=0}^{N-1}\mathbf{y}^*(n)e(n) = 0 \end{aligned} \tag{15.12.8}$$

To simplify the expressions, we will drop the common factor $1/N$ in the above time-averages. Next, we define the $N\times(M+1)$ *data matrix* Y whose *rows* are the N snapshots $\mathbf{y}^T(n)$,

$$Y = \begin{bmatrix} \mathbf{y}^T(0) \\ \mathbf{y}^T(1) \\ \vdots \\ \mathbf{y}^T(n) \\ \vdots \\ \mathbf{y}^T(N-1) \end{bmatrix} = \begin{bmatrix} y_0(0) & y_1(0) & \cdots & y_M(0) \\ y_0(1) & y_1(1) & \cdots & y_M(1) \\ \vdots & \vdots & \vdots & \\ y_0(n) & y_1(n) & \cdots & y_M(n) \\ \vdots & \vdots & \vdots & \\ y_0(N-1) & y_1(N-1) & \cdots & y_M(N-1) \end{bmatrix} \tag{15.12.9}$$

The ni-th matrix element of the data matrix is $Y_{ni} = y_i(n)$, $0 \le n \le N-1$, $0 \le i \le M$. In particular, in the time series case, we have $Y_{ni} = y_{n-i}$. We defined Y in terms of its rows. It can also be defined column-wise, where the ith column is an N-dimensional time signal $\mathbf{y}_i = [y_i(0),\dots,y_i(n),\dots,y_i(N-1)]^T$. Therefore,

$$Y = [\mathbf{y}_0, \mathbf{y}_1, \dots, \mathbf{y}_M] = \begin{bmatrix} \mathbf{y}^T(0) \\ \vdots \\ \mathbf{y}^T(n) \\ \vdots \\ \mathbf{y}^T(N-1) \end{bmatrix} \tag{15.12.10}$$

The $N\times 1$ column vectors of the $x(n)$, $e(n)$, and the estimates $\hat{x}(n)$ are:

$$\mathbf{x} = \begin{bmatrix} x(0) \\ x(1) \\ \vdots \\ x(n) \\ \vdots \\ x(N-1) \end{bmatrix}, \quad \mathbf{e} = \begin{bmatrix} e(0) \\ e(1) \\ \vdots \\ e(n) \\ \vdots \\ e(N-1) \end{bmatrix}, \quad \hat{\mathbf{x}} = \begin{bmatrix} \hat{x}(0) \\ \hat{x}(1) \\ \vdots \\ \hat{x}(n) \\ \vdots \\ \hat{x}(N-1) \end{bmatrix} \tag{15.12.11}$$

Noting that $Y^\dagger = Y^{*T} = [\mathbf{y}^*(0), \mathbf{y}^*(1), \dots, \mathbf{y}^*(N-1)]$, we can write Eqs. (15.12.8) in the following compact forms (without the $1/N$ factor):

$$\hat{R} = Y^\dagger Y, \qquad \hat{\mathbf{r}} = Y^\dagger\mathbf{x}, \qquad Y^\dagger\mathbf{e} = 0 \tag{15.12.12}$$

Indeed, we have:

$$\hat{R} = \sum_{n=0}^{N-1}\mathbf{y}^*(n)\mathbf{y}^T(n) = [\mathbf{y}^*(0), \mathbf{y}^*(1), \dots, \mathbf{y}^*(N-1)]\begin{bmatrix} \mathbf{y}^T(0) \\ \mathbf{y}^T(1) \\ \vdots \\ \mathbf{y}^T(N-1) \end{bmatrix} = Y^\dagger Y$$

$$\hat{\mathbf{r}} = \sum_{n=0}^{N-1}\mathbf{y}^*(n)x(n) = [\mathbf{y}^*(0), \mathbf{y}^*(1), \dots, \mathbf{y}^*(N-1)]\begin{bmatrix} x(0) \\ x(1) \\ \vdots \\ x(N-1) \end{bmatrix} = Y^\dagger\mathbf{x}$$

Similarly, replacing $\hat{x}(n) = \mathbf{y}^T(n)\mathbf{h}$ in (15.12.11), we obtain:

$$\hat{\mathbf{x}} = Y\mathbf{h}\,, \qquad \mathbf{e} = \mathbf{x} - \hat{\mathbf{x}} = \mathbf{x} - Y\mathbf{h} \tag{15.12.13}$$

The performance index is replaced by the least-squares index:

$$\mathcal{E} = E\big[|e(n)|^2\big] = \min \quad\Rightarrow\quad \hat{\mathcal{E}} = \sum_{n=0}^{N-1}|e(n)|^2 = \mathbf{e}^\dagger\mathbf{e} = \|\mathbf{x} - Y\mathbf{h}\|^2 = \min \tag{15.12.14}$$

The minimization of the least-squares index with respect to $\mathbf{h}$ gives rise to the orthogonality and normal equations, as in Eq. (15.4.2):

$$Y^\dagger\mathbf{e} = 0\,, \qquad Y^\dagger Y\mathbf{h} = Y^\dagger\mathbf{x} \quad\Rightarrow\quad \hat{R}\mathbf{h} = \hat{\mathbf{r}} \tag{15.12.15}$$

Thus, we recognize that replacing the theoretical normal equations $R\mathbf{h} = \mathbf{r}$ by their time-averaged versions $\hat{R}\mathbf{h} = \hat{\mathbf{r}}$ is equivalent to solving—in the *least-squares sense*—the overdetermined $N\times(M+1)$ linear system:

$$\boxed{Y\mathbf{h} = \mathbf{x}} \tag{15.12.16}$$

The SVD of the data matrix, $Y = U\Sigma V^\dagger$, can used to characterize the nature of the solutions of these equations. The min-norm and backslash solutions are in MATLAB's notation:

$$\mathbf{h} = \text{pinv}(Y)*\mathbf{x}\,, \quad \mathbf{h} = Y\backslash\mathbf{x} \tag{15.12.17}$$

Since $N > M+1$, these will be the same if Y has full rank, that is, $r = M+1$. In this case, the solution is unique and is given by:

$$\mathbf{h} = (Y^\dagger Y)^{-1}Y^\dagger\mathbf{x} = \hat{R}^{-1}\hat{\mathbf{r}} \qquad \text{(full rank } Y\text{)} \tag{15.12.18}$$

In the time-series case, some further clarification of the definition of the data matrix Y is necessary. Since $y_m(n) = y_{n-m}$, the estimate $\hat{x}(n)$ is obtained by convolving the order-M filter $\mathbf{h}$ with the sequence y_n:

$$\hat{x}(n) = \sum_{m=0}^{M} h_m y_m(n) = \sum_{m=0}^{M} h_m y_{n-m}$$

For a length-N input signal y_n, $n = 0, 1, \dots, N-1$, the output sequence $\hat{x}(n)$ will have length $N+M$, with the first M output samples corresponding to the *input-on* transients, the last M outputs being the *input-off* transients, and the middle $N-M$ samples, $\hat{x}(n)$, $n = M, \dots, N-1$, being the *steady-state* outputs.

There are several possible choices in defining the range of summation over n in the least-squares index:

$$\hat{\mathcal{E}} = \sum_n |e(n)|^2,$$

One can consider: (a) the full range, $0 \le n \le N-1+M$, leading to the so-called *autocorrelation method*, (b) the steady-state range, $M \le n \le N-1$, leading to the *covariance method*, (c) the pre-windowed range, $0 \le n \le N-1$, or (d) the post-windowed range, $M \le n \le N-1+M$. The autocorrelation and covariance choices are the most widely used:

$$\hat{\mathcal{E}}_{\text{aut}} = \sum_{n=0}^{N-1+M}|e(n)|^2\,, \qquad \hat{\mathcal{E}}_{\text{cov}} = \sum_{n=M}^{N-1}|e(n)|^2 \tag{15.12.19}$$

The minimization of these indices leads to the least-squares equations $Y\mathbf{h} = \mathbf{x}$, where Y is defined as follows. First, we define the input-on and input-off parts of Y in terms of the first M and last M data vectors:

$$Y_{\text{on}} = \begin{bmatrix} \mathbf{y}^T(0) \\ \vdots \\ \mathbf{y}^T(M-1) \end{bmatrix}, \qquad Y_{\text{off}} = \begin{bmatrix} \mathbf{y}^T(N) \\ \vdots \\ \mathbf{y}^T(N-1+M) \end{bmatrix}$$

Then, we define Y for the autocorrelation and covariance cases:

$$Y_{\text{aut}} = \left[\begin{array}{c} \mathbf{y}^T(0) \\ \vdots \\ \mathbf{y}^T(M-1) \\ \hline \mathbf{y}^T(M) \\ \vdots \\ \mathbf{y}^T(N-1) \\ \hline \mathbf{y}^T(N) \\ \vdots \\ \mathbf{y}^T(N-1+M) \end{array}\right] = \left[\begin{array}{c} Y_{\text{on}} \\ \hline Y_{\text{cov}} \\ \hline Y_{\text{off}} \end{array}\right], \qquad Y_{\text{cov}} = \begin{bmatrix} \mathbf{y}^T(M) \\ \vdots \\ \mathbf{y}^T(N-1) \end{bmatrix} \tag{15.12.20}$$

To clarify these expressions, consider an example where $N = 6$ and $M = 2$. The observation sequence is y_n, $n = 0, 1, \dots, 5$. Noting that y_n is causal and that it is zero for $n \ge 6$, we have:

$$Y_{\text{aut}} = \left[\begin{array}{ccc} y_0 & 0 & 0 \\ y_1 & y_0 & 0 \\ \hline y_2 & y_1 & y_0 \\ y_3 & y_2 & y_1 \\ y_4 & y_3 & y_2 \\ y_5 & y_4 & y_3 \\ \hline 0 & y_5 & y_4 \\ 0 & 0 & y_5 \end{array}\right], \qquad Y_{\text{cov}} = \begin{bmatrix} y_2 & y_1 & y_0 \\ y_3 & y_2 & y_1 \\ y_4 & y_3 & y_2 \\ y_5 & y_4 & y_3 \end{bmatrix}$$

These follow from the definition $\mathbf{y}^T(n) = [y_n, y_{n-1}, y_{n-2}]$, which gives, $\mathbf{y}^T(0) = [y_0, y_{-1}, y_{-2}] = [y_0, 0, 0]$, and so on until the last time sample at $n = N-1+M = 6-1+2 = 7$, that is, $\mathbf{y}^T(7) = [y_7, y_6, y_5] = [0, 0, y_5]$. The middle portion of Y_{aut} is the covariance version Y_{cov}.

The autocorrelation version Y_{aut} is recognized as the ordinary Toeplitz *convolution matrix* for a length-6 input signal and an order-2 filter. It can be constructed easily by invoking MATLAB's built-in function **convmtx**:

```
Y = convmtx(y,M+1);              % y is a column vector of time samples
```

The least-squares linear system $Y\mathbf{h} = \mathbf{x}$ for determining the optimum weights $\mathbf{h} = [h_0, h_1, h_2]^T$ reads as follows in the two cases:

$$\begin{bmatrix} y_0 & 0 & 0 \\ y_1 & y_0 & 0 \\ y_2 & y_1 & y_0 \\ y_3 & y_2 & y_1 \\ y_4 & y_3 & y_2 \\ y_5 & y_4 & y_3 \\ 0 & y_5 & y_4 \\ 0 & 0 & y_5 \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \\ h_2 \end{bmatrix} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix}, \qquad \begin{bmatrix} y_2 & y_1 & y_0 \\ y_3 & y_2 & y_1 \\ y_4 & y_3 & y_2 \\ y_5 & y_4 & y_3 \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \\ h_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}$$

where we assumed that the signal $x(n)$ was available for $0 \le n \le N - 1 + M = 7$.

There is yet a third type of a data matrix that is used in linear prediction applications. It corresponds to the *modified covariance* method, also known as the *forward-backward* method. The data matrix is obtained by appending its *row-reversed and complex-conjugated* version. For our example, this gives:

$$Y_{\text{fb}} = \left[\begin{array}{ccc} y_2 & y_1 & y_0 \\ y_3 & y_2 & y_1 \\ y_4 & y_3 & y_2 \\ y_5 & y_4 & y_3 \\ \hline y_0^* & y_1^* & y_2^* \\ y_1^* & y_2^* & y_3^* \\ y_2^* & y_3^* & y_4^* \\ y_3^* & y_4^* & y_5^* \end{array}\right] = \left[\begin{array}{c} Y_{\text{cov}} \\ \hline Y_{\text{cov}}^* J \end{array}\right] \tag{15.12.21}$$

where J is the usual reversing matrix consisting of ones along its antidiagonal. While Y_{aut} and Y_{cov} are Toeplitz matrices, only the upper half of Y_{fb} is Toeplitz whereas its lower half is a *Hankel* matrix, that is, it has the same entries along each antidiagonal.

Given one of the three types of a data matrix Y, one can extract the signal y_n that generated that Y. The MATLAB function **datamat** (in the OSP toolbox) constructs a data matrix from the signal y_n, whereas the function **datasig** extracts the signal y_n from Y. The functions have usage:

```
Y = datamat(y,M,type);      % type =0,1,2, for autocorrelation,
y = datasig(Y,type);        % covariance, or F/B methods
```

15.13 Least-Squares Linear Prediction

Next, we discuss briefly how linear prediction problems can be solved in a least-squares sense. For an order-M predictor, we define the forward and backward prediction errors in terms of the forward and an reversed-conjugated filters:

$$e_+(n) = [y_n, y_{n-1}, \dots, y_{n-M}] \begin{bmatrix} 1 \\ a_1 \\ \vdots \\ a_M \end{bmatrix} = \mathbf{y}^T(n)\,\mathbf{a} \tag{15.13.1}$$

$$e_-(n) = [y_n, y_{n-1}, \dots, y_{n-M}] \begin{bmatrix} a_M^* \\ \vdots \\ a_1^* \\ 1 \end{bmatrix} = \mathbf{y}^T(n)\,\mathbf{a}^{R*} \tag{15.13.2}$$

The prediction coefficients $\mathbf{a}$ are found by minimizing one of the three least-square performance indices, corresponding to the autocorrelation, covariance, and forward/backward methods:

$$\begin{aligned} \hat{\mathcal{E}}_{\text{aut}} &= \sum_{n=0}^{N-1+M} |e_+(n)|^2 = \min \\ \hat{\mathcal{E}}_{\text{cov}} &= \sum_{n=M}^{N-1} |e_+(n)|^2 = \min \\ \hat{\mathcal{E}}_{\text{fb}} &= \sum_{n=M}^{N-1} \big[|e_+(n)|^2 + |e_-(n)|^2\big] = \min \end{aligned} \tag{15.13.3}$$

Stacking the samples $e_\pm(n)$ into a column vector, we may express the error vectors in terms of the corresponding autocorrelation or covariance data matrices:

$$\begin{aligned} \mathbf{e}_+ &= Y\mathbf{a} \\ \mathbf{e}_- &= Y\mathbf{a}^{R*} \end{aligned} \qquad \text{where} \quad \mathbf{e}_+ = \begin{bmatrix} \vdots \\ e_+(n) \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ \mathbf{y}^T(n) \\ \vdots \end{bmatrix} \mathbf{a} = Y\mathbf{a} \tag{15.13.4}$$

and similarly for $\mathbf{e}_-$. Noting that $\mathbf{a}^R = J\mathbf{a}$, we have for the covariance case:

$$\mathbf{e}_- = Y_{\text{cov}} J\mathbf{a}^* \quad \Rightarrow \quad \mathbf{e}_-^* = (Y_{\text{cov}}^* J)\,\mathbf{a}$$

Then, we may define the extended error vector consisting of both the forward and backward errors:

$$\mathbf{e} = \begin{bmatrix} \mathbf{e}_+ \\ \mathbf{e}_-^* \end{bmatrix} = \begin{bmatrix} Y_{\text{cov}} \\ Y_{\text{cov}}^* J \end{bmatrix} \mathbf{a} = Y_{\text{fb}}\,\mathbf{a} \tag{15.13.5}$$

Noting that $\mathbf{e}^\dagger\mathbf{e} = \mathbf{e}_+^\dagger\mathbf{e}_+ + \mathbf{e}_-^\dagger\mathbf{e}_-$, we may express the indices (15.13.3) in the compact forms:

$$\begin{aligned} \hat{\mathcal{E}}_{\text{aut}} &= \mathbf{e}_+^\dagger\mathbf{e}_+ = \|\mathbf{e}_+\|^2 = \|Y_{\text{aut}}\,\mathbf{a}\|^2 \\ \hat{\mathcal{E}}_{\text{cov}} &= \mathbf{e}_+^\dagger\mathbf{e}_+ = \|\mathbf{e}_+\|^2 = \|Y_{\text{cov}}\,\mathbf{a}\|^2 \\ \hat{\mathcal{E}}_{\text{fb}} &= \mathbf{e}_+^\dagger\mathbf{e}_+ + \mathbf{e}_-^\dagger\mathbf{e}_- = \|\mathbf{e}_+\|^2 + \|\mathbf{e}_-\|^2 = \|\mathbf{e}\|^2 = \|Y_{\text{fb}}\,\mathbf{a}\|^2 \end{aligned} \tag{15.13.6}$$

Thus, in all three cases, the problem reduces to the least-squares solution of the linear equation $Y\mathbf{a} = 0$, that is,

$$Y\mathbf{a} = 0 \quad \Leftrightarrow \quad \hat{\mathcal{E}} = \|\mathbf{e}\|^2 = \|Y\mathbf{a}\|^2 = \min \tag{15.13.7}$$

subject to the constraint $a_0 = 1$. The solution is obtained by separating the first column of the matrix Y in order to take the constraint into account. Setting $Y = [\mathbf{y}_0, Y_1]$ and $\mathbf{a}^T = [1, \boldsymbol{\alpha}^T]$, we find the equivalent linear system:

$$Y\mathbf{a} = [\mathbf{y}_0, Y_1]\begin{bmatrix} 1 \\ \boldsymbol{\alpha} \end{bmatrix} = \mathbf{y}_0 + Y_1\boldsymbol{\alpha} = 0 \quad \Rightarrow \quad Y_1\boldsymbol{\alpha} = -\mathbf{y}_0 \tag{15.13.8}$$

The minimum-norm least-squares solution is obtained by the pseudoinverse:

$$\boldsymbol{\alpha} = -\text{pinv}(Y_1)*\mathbf{y}_0 = -Y_1^+\mathbf{y}_0 \quad \Rightarrow \quad \mathbf{a} = \begin{bmatrix} 1 \\ \boldsymbol{\alpha} \end{bmatrix} = \begin{bmatrix} 1 \\ -Y_1^+\mathbf{y}_0 \end{bmatrix} \tag{15.13.9}$$

The OSP function **lpls** implements this procedure. It has usage:

```
[a,E] = lpls(Y);    % least-squares linear prediction filter
```

where E is the minimized prediction error $E = \|\mathbf{e}\|^2/L$, where L is the column dimension of Y. Combined with the function **datamat**, one can obtain the prediction filter according to the three criteria:

```
[a,E] = lpls(datamat(y,M,0))    % autocorrelation or Yule-Walker method
[a,E] = lpls(datamat(y,M,1))    % covariance method
[a,E] = lpls(datamat(y,M,2))    % modified covariance or f/b method
```

The autocorrelation method can be computed by the alternative call to the Yule-Walker function **yw**:

```
a = lpf(yw(y,M));   % autocorrelation or Yule-Walker method
```

Further improvements of these methods result, especially in the case of extracting sinusoids in noise, when the least-squares solution (15.13.9) is used in conjunction with the SVD enhancement iteration procedure discussed in Sec. 15.17.

15.14 MA and ARMA modeling

There are many methods for fitting MA and ARMA models to a given data sequence y_n, $n = 0, 1, \dots, N-1$. Some methods are nonlinear and involve an iterative minimization of a maximum likelihood criterion. Other methods are adaptive, continuously updating the model parameters on a sample by sample basis.

Here, we briefly discuss a class of methods, originally suggested by Durbin [1299,1300], which begin by fitting a long AR model to the data, and then deriving the MA or ARMA model parameters from that AR model by using only least-squares solutions of linear equations.

MA Models

A moving-average model of order q, denoted by MA(q), is described by the I/O equation driven by a zero-mean white-noise signal ϵ_n of variance σ_ϵ^2:

$$y_n = b_0\epsilon_n + b_1\epsilon_{n-1} + b_2\epsilon_{n-2} + \cdots + b_q\epsilon_{n-q} \tag{15.14.1}$$

Thus, the synthesis model filter and the power spectrum of y_n are:

$$B(z) = b_0 + b_1z^{-1} + b_2z^{-2} + \cdots + b_qz^{-q}\,, \quad S_{yy}(\omega) = \sigma_\epsilon^2\,\big|B(\omega)\big|^2 \tag{15.14.2}$$

Without loss of generality, we may assume that $b_0 = 1$. We will also assume that $B(z)$ is a minimum-phase polynomial so that the analysis filter $A(z) = 1/B(z)$ is stable and causal.

Durbin's method consists of approximating the analysis filter $A(z)$ by a polynomial $A_M(z)$ of some large order M, such that $M \gg q$. The polynomial coefficients $\mathbf{a} = [1, a_1, \dots, a_M]^T$ are found by applying any least-squares LP method to the given sequence $\mathbf{y} = [y_0, y_1, \dots, y_{N-1}]^T$, including Burg's method.

Finally, the desired MA filter $\mathbf{b} = [1, b_1, \dots, b_q]^T$ is obtained by designing an order-q least-squares inverse to $\mathbf{a} = [1, a_1, \dots, a_M]^T$ using, for example, the techniques of section 12.14. Specifically, we wish to solve the approximate equation $A_M(z)B(z) \simeq 1$. This condition may be expressed in matrix form using the $(M+q+1)\times(q+1)$ convolution matrix of the filter $\mathbf{a}$ acting on the input $\mathbf{b}$:

$$A\mathbf{b} = \mathbf{u}\,, \quad \text{where } A = \text{datamat}(\mathbf{a}, q) \tag{15.14.3}$$

and $\mathbf{u} = [1, 0, \dots, 0]^T$ is the $(M+q+1)$-dimensional representation of δ_n. For example, if $q = 2$ and $M = 4$, we have:

$$\begin{bmatrix} 1 & 0 & 0 \\ a_1 & 1 & 0 \\ a_2 & a_1 & 1 \\ a_3 & a_2 & a_1 \\ a_4 & a_3 & a_2 \\ 0 & a_4 & a_3 \\ 0 & 0 & a_4 \end{bmatrix} \begin{bmatrix} 1 \\ b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \Rightarrow \quad \begin{bmatrix} a_1 & 1 & 0 \\ a_2 & a_1 & 1 \\ a_3 & a_2 & a_1 \\ a_4 & a_3 & a_2 \\ 0 & a_4 & a_3 \\ 0 & 0 & a_4 \end{bmatrix} \begin{bmatrix} 1 \\ b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{15.14.4}$$

where in the second equation, we deleted the first row of A, which corresponds to the identity $1 = 1$. Thus, denoting the bottom part of A by A_{bot}, we obtain the following $(M+q)\times(q+1)$ linear system to be solved by least-squares:

$$A_{\text{bot}}\mathbf{b} = 0 \quad \Rightarrow \quad \mathbf{b} = \text{lpls}(A_{\text{bot}}) \tag{15.14.5}$$

This problem is identical to that of Eq. (15.13.7) and therefore, its solution was obtained with the help of the function **lpls**. These design steps have been incorporated into the MATLAB function **madurbin** with usage:

```
[b,sigma2] = madurbin(y,q,M);     % MA modeling by Durbin's method
```

To clarify further the above solution, we write (15.14.5) in partitioned form, separating out the first column of A_{bot} and the bottom part of **b**:

$$A_{\text{bot}} = [\mathbf{a}_1, A_1]\,,\quad \mathbf{b} = \begin{bmatrix} 1 \\ \boldsymbol{\beta} \end{bmatrix} \quad\Rightarrow\quad A_{\text{bot}}\mathbf{b} = [\mathbf{a}_1, A_1]\begin{bmatrix} 1 \\ \boldsymbol{\beta} \end{bmatrix} = \mathbf{a}_1 + A_1\boldsymbol{\beta} = 0$$

The least-squares solution is (assuming A_1 has full rank):

$$\boldsymbol{\beta} = -A_1\backslash\,\mathbf{a}_1 = -(A_1^\dagger A_1)^{-1}A_1^\dagger\mathbf{a}_1 \tag{15.14.6}$$

This has the form of a linear prediction solution $\boldsymbol{\beta} = -R^{-1}\mathbf{r}$, where $R = A_1^\dagger A_1$ and $\mathbf{r} = A_1^\dagger\mathbf{a}_1$. It easily verified that $R, \mathbf{r}$ are given by:

$$R_{ij} = (A_1^\dagger A_1)_{ij} = R_{aa}(i-j)\,,\quad r_i = (A_1^\dagger\mathbf{a}_1)_i = R_{aa}(i+1) \tag{15.14.7}$$

for $i, j = 0, 1, \dots, q-1$, and R_{aa} is the sample autocorrelation of the filter **a**:

$$R_{aa}(k) = \sum_{m=0}^{M-|k|} a^*_{m+|k|}a_m\,,\quad -M \le k \le M \tag{15.14.8}$$

In other words, as observed by Durbin, the MA filter **b** may obtained by fitting an AR(q) model to the AR filter **a** using the autocorrelation or Yule-Walker method. Thus, an alternative design procedure is by the following two steps:

```
a = lpf(yw(y,M));     % fit an AR(M) model to y
b = lpf(yw(a,q));     % fit an AR(q) model to a
```

where the function **lpf** extracts the prediction filter from the output of the function **yw**. Once the MA filter is designed, the input noise variance σ_ϵ^2 may be calculated using Parseval's identity:

$$\sigma_y^2 = \sigma_\epsilon^2\int_{-\pi}^{\pi}|B(\omega)|^2\frac{d\omega}{2\pi} = \sigma_\epsilon^2\sum_{m=0}^{q}|b_m|^2 = \sigma_\epsilon^2\mathbf{b}^\dagger\mathbf{b} \quad\Rightarrow\quad \sigma_\epsilon^2 = \frac{\sigma_y^2}{\mathbf{b}^\dagger\mathbf{b}}$$

where σ_y^2 can be estimated directly from the data sequence by:

$$\hat{\sigma}_y^2 = \frac{1}{N}\sum_{n=0}^{N-1}|y_n|^2$$

ARMA models

An ARMA(p, q) model is characterized by a synthesis filter of the form:

$$H(z) = \frac{B(z)}{A(z)} = \frac{1 + b_1z^{-1} + \cdots + b_qz^{-q}}{1 + a_1z^{-1} + \ldots + a_pz^{-p}} \tag{15.14.9}$$

The sequence y_n is generated by driving $H(z)$ by zero-mean white-noise ϵ_n:

$$y_n + a_1y_{n-1} + \cdots + a_py_{n-p} = \epsilon_n + b_1\epsilon_{n-1} + \cdots + b_q\epsilon_{n-q} \tag{15.14.10}$$

The corresponding power spectrum of y_n is:

$$S_{yy}(\omega) = \sigma_\epsilon^2|H(\omega)|^2 = \sigma_\epsilon^2\left|\frac{B(\omega)}{A(\omega)}\right|^2 = \sigma_\epsilon^2\left|\frac{1 + b_1e^{-j\omega} + \cdots + b_qe^{-jq\omega}}{1 + a_1e^{-j\omega} + \cdots + a_pe^{-jp\omega}}\right|^2$$

If the innovations sequence ϵ_n were known, then by considering Eq. (15.14.10) at successive time instants, say, $n = 0, 1, \dots, N-1$, one could solve for the model parameters $\mathbf{a} = [1, a_1, \dots, a_p]^T$ and $\mathbf{b} = [1, b_1, \dots, b_q]^T$. To see how this might be done, we rewrite (15.14.10) vectorially in the form:

$$[y_n, y_{n-1}, \dots, y_{n-p}]\begin{bmatrix} 1 \\ a_1 \\ \vdots \\ a_p \end{bmatrix} = [\epsilon_n, \epsilon_{n-1}, \dots, \epsilon_{n-q}]\begin{bmatrix} 1 \\ b_1 \\ \vdots \\ b_q \end{bmatrix} \tag{15.14.11}$$

or, compactly,

$$\mathbf{y}^T(n)\mathbf{a} = \mathbf{e}^T(n)\mathbf{b}\,,\quad\text{where}\quad \mathbf{y}(n) = \begin{bmatrix} y_n \\ y_{n-1} \\ \vdots \\ y_{n-p} \end{bmatrix},\quad \mathbf{e}(n) = \begin{bmatrix} \epsilon_n \\ \epsilon_{n-1} \\ \vdots \\ \epsilon_{n-q} \end{bmatrix} \tag{15.14.12}$$

Arranging these into a column vector for $n = 0, 1\dots, N-1$, we may express them as a single vector equation involving the data matrices of y_n and ϵ_n:

$$Y\mathbf{a} = E\,\mathbf{b}\,,\quad\text{where}\quad Y = \begin{bmatrix} \mathbf{y}^T(0) \\ \vdots \\ \mathbf{y}^T(n) \\ \vdots \\ \mathbf{y}^T(N-1) \end{bmatrix},\quad E = \begin{bmatrix} \mathbf{e}^T(0) \\ \vdots \\ \mathbf{e}^T(n) \\ \vdots \\ \mathbf{e}^T(N-1) \end{bmatrix} \tag{15.14.13}$$

The data matrices Y and E have dimensions $N\times(p+1)$ and $N\times(q+1)$, respectively, and correspond to the "prewindowed" type. They can be constructed from the sequences $\mathbf{y} = [y_0, y_1\dots, y_{N-1}]^T$ and $\mathbf{e} = [\epsilon_0, \epsilon_1\dots, \epsilon_{N-1}]^T$ by the OSP function **datamat**:

$$\boxed{\begin{aligned} Y &= \text{datamat}(\mathbf{y}, p, \texttt{'pre'}) \\ E &= \text{datamat}(\mathbf{e}, q, \texttt{'pre'}) \end{aligned}}$$

For example, if $N = 7$, $p = 3$, and $q = 2$, and assuming zero initial conditions, then Eq. (15.14.13) reads as follows:

$$\begin{bmatrix} y_0 & 0 & 0 & 0 \\ y_1 & y_0 & 0 & 0 \\ y_2 & y_1 & y_0 & 0 \\ y_3 & y_2 & y_1 & y_0 \\ y_4 & y_3 & y_2 & y_1 \\ y_5 & y_4 & y_3 & y_2 \\ y_6 & y_5 & y_4 & y_3 \end{bmatrix}\begin{bmatrix} 1 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} \epsilon_0 & 0 & 0 \\ \epsilon_1 & \epsilon_0 & 0 \\ \epsilon_2 & \epsilon_1 & \epsilon_0 \\ \epsilon_3 & \epsilon_2 & \epsilon_1 \\ \epsilon_4 & \epsilon_3 & \epsilon_2 \\ \epsilon_5 & \epsilon_4 & \epsilon_3 \\ \epsilon_6 & \epsilon_5 & \epsilon_4 \end{bmatrix}\begin{bmatrix} 1 \\ b_1 \\ b_2 \end{bmatrix} \tag{15.14.14}$$

Even though overdetermined, these equations are consistent and may be solved for the model parameters. Unfortunately, in practice, only the observed output sequence $\mathbf{y} = [y_0, y_1 \dots, y_{N-1}]^T$ is available.

A possible strategy to overcome this problem, originally proposed by Durbin, is to replace the unknown exact innovation vector $\mathbf{e} = [\epsilon_0, \epsilon_1 \dots, \epsilon_{N-1}]^T$ by an estimated one $\hat{\mathbf{e}} = [\hat{\epsilon}_0, \hat{\epsilon}_1 \dots, \hat{\epsilon}_{N-1}]^T$ and then solve (15.14.13) approximately using least-squares, that is, if $\hat{E}$ is the data matrix of the approximate innovations, then solve the least-squares problem:

$$Y\hat{\mathbf{a}} = \hat{E}\hat{\mathbf{b}} \quad \Leftrightarrow \quad \mathcal{J} = \|Y\hat{\mathbf{a}} - \hat{E}\hat{\mathbf{b}}\|^2 = \min \tag{15.14.15}$$

One way to obtain an estimated innovations sequence $\hat{\mathbf{e}}$ is to fit to $\mathbf{y}$ an autoregressive model $A_M(z)$ of large order M, such that $M \gg p + q$. This amounts to approximating the synthesis filter by the all-pole model $\hat{H}(z) = 1/A_M(z)$. Passing the given sequence y_n through the approximate analysis filter $A_M(z)$ would generate the estimated innovations, that is, $\hat{E}(z) = A_M(z)Y(z)$. Thus, the *first step* in the design is, in MATLAB notation:

$$\boxed{\begin{aligned} \mathbf{a}_M &= \text{lpf}\big(\text{yw}(\mathbf{y}, M)\big) \\ \hat{\mathbf{e}} &= \text{filter}(\mathbf{a}_M, 1, \mathbf{y}) \\ \hat{E} &= \text{datamat}(\hat{\mathbf{e}}, q, \texttt{'pre'}) \end{aligned}} \tag{15.14.16}$$

The *second step* is to solve (15.14.15) using least squares. To this end, we separate the first columns of the matrices $Y, \hat{E}$, and the bottom parts of $\hat{\mathbf{a}}, \hat{\mathbf{b}}$ to get:

$$Y = [\mathbf{y}_0, Y_1], \quad \hat{E} = [\hat{\mathbf{e}}_0, \hat{E}_1], \quad \hat{\mathbf{a}} = \begin{bmatrix} 1 \\ \hat{\boldsymbol{\alpha}} \end{bmatrix}, \quad \hat{\mathbf{b}} = \begin{bmatrix} 1 \\ \hat{\boldsymbol{\beta}} \end{bmatrix}$$

and recast (15.14.15) in the form:

$$Y\hat{\mathbf{a}} = \hat{E}\hat{\mathbf{b}} \quad \Rightarrow \quad [\mathbf{y}_0, Y_1]\begin{bmatrix} 1 \\ \hat{\boldsymbol{\alpha}} \end{bmatrix} = [\hat{\mathbf{e}}_0, \hat{E}_1]\begin{bmatrix} 1 \\ \hat{\boldsymbol{\beta}} \end{bmatrix} \quad \Rightarrow \quad \mathbf{y}_0 + Y_1\hat{\boldsymbol{\alpha}} = \hat{\mathbf{e}}_0 + \hat{E}_1\hat{\boldsymbol{\beta}}$$

This may be rearranged into $Y_1\hat{\boldsymbol{\alpha}} - \hat{E}_1\hat{\boldsymbol{\beta}} = -(\mathbf{y}_0 - \hat{\mathbf{e}}_0)$, and solved:

$$[Y_1, -\hat{E}_1]\begin{bmatrix} \hat{\boldsymbol{\alpha}} \\ \hat{\boldsymbol{\beta}} \end{bmatrix} = -(\mathbf{y}_0 - \hat{\mathbf{e}}_0) \quad \Rightarrow \quad \begin{bmatrix} \hat{\boldsymbol{\alpha}} \\ \hat{\boldsymbol{\beta}} \end{bmatrix} = -[Y_1, -\hat{E}_1] \setminus (\mathbf{y}_0 - \hat{\mathbf{e}}_0) \tag{15.14.17}$$

This completes step two. We note that because of the prewindowed choice of the data matrices, the first columns $\mathbf{y}_0, \hat{\mathbf{e}}_0$ are the length-N signal sequences $\mathbf{y}$ and $\hat{\mathbf{e}}$ themselves, that is, $\mathbf{y}_0 = \mathbf{y} = [y_0, y_1 \dots, y_{N-1}]^T$ and $\hat{\mathbf{e}}_0 = \hat{\mathbf{e}} = [\hat{\epsilon}_0, \hat{\epsilon}_1 \dots, \hat{\epsilon}_{N-1}]^T$. Thus, we have, $\mathbf{y} + Y_1\hat{\boldsymbol{\alpha}} = \hat{\mathbf{e}} + \hat{E}_1\hat{\boldsymbol{\beta}}$, and rearranging, $\hat{\mathbf{e}} = \mathbf{y} + Y_1\hat{\boldsymbol{\alpha}} - \hat{E}_1\hat{\boldsymbol{\beta}}$. An alternative least-squares criterion that is used sometimes is the following:

$$\mathcal{J} = \|\hat{\mathbf{e}}\|^2 = \|\mathbf{y} + Y_1\hat{\boldsymbol{\alpha}} - \hat{E}_1\hat{\boldsymbol{\beta}}\|^2 = \min \tag{15.14.18}$$

This has the solution:

$$\begin{bmatrix} \hat{\boldsymbol{\alpha}} \\ \hat{\boldsymbol{\beta}} \end{bmatrix} = -[Y_1, -\hat{E}_1] \setminus \mathbf{y} \tag{15.14.19}$$

We will be using (15.14.17). The second-stage solutions can be shown not to be asymptotically efficient. In order to minimize their variance, Mayne and Firoozan [1301] proposed a *third step*. It consists of replacing the sequences $y_n, \hat{\epsilon}_n$ by the inverse-filtered versions $V(z) = Y(z)/\hat{B}(z)$ and $W(z) = \hat{E}(z)/\hat{B}(z)$ and repeating step two. This produces the final estimates of the ARMA parameters $\mathbf{a}$ and $\mathbf{b}$.

The filtered sequences v_n, w_n and their data matrices are constructed as follows, in MATLAB notation:

$$\boxed{\begin{aligned} \mathbf{v} &= \text{filter}(1, \hat{\mathbf{b}}, \mathbf{y}); \quad V = \text{datamat}(\mathbf{v}, p, \texttt{'pre'}); \\ \mathbf{w} &= \text{filter}(1, \hat{\mathbf{b}}, \hat{\mathbf{e}}); \quad W = \text{datamat}(\mathbf{w}, q, \texttt{'pre'}); \end{aligned}} \tag{15.14.20}$$

The resulting least-squares problem is then:

$$V\mathbf{a} = W\mathbf{b} \quad \Rightarrow \quad [\mathbf{v}_0, V_1]\begin{bmatrix} 1 \\ \boldsymbol{\alpha} \end{bmatrix} = [\mathbf{w}_0, W_1]\begin{bmatrix} 1 \\ \boldsymbol{\beta} \end{bmatrix} \tag{15.14.21}$$

with performance index $\mathcal{J} = \|V\mathbf{a} - W\mathbf{b}\|^2 = \min$. The solution of (15.14.21) is:

$$\begin{bmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{bmatrix} = -[V_1, -W_1] \setminus (\mathbf{v}_0 - \mathbf{w}_0) \tag{15.14.22}$$

In summary, the Mayne-Firoozan three-stage ARMA parameter estimation method consists of Eqs. (15.14.16), (15.14.17), and (15.14.22).

To justify the need for the inverse filtering, we consider an improved innovations vector obtained from $\hat{\mathbf{e}}$ by adding a small correction, that is, $\mathbf{e} = \hat{\mathbf{e}} + \delta\mathbf{e}$, or in terms of the data matrices, $E = \hat{E} + \delta E$. We imagine that the vector $\mathbf{e}$ is closer to the true innovations vector than $\hat{\mathbf{e}}$. The small change $\delta\mathbf{e}$ will induce similar small changes in the ARMA parameters, $\mathbf{a} = \hat{\mathbf{a}} + \delta\mathbf{a}$ and $\mathbf{b} = \hat{\mathbf{b}} + \delta\mathbf{b}$, which must satisfy the improved input/output equation:

$$Y\mathbf{a} = E\mathbf{b} \tag{15.14.23}$$

To first order in the corrections, that is, ignoring terms like $\delta E\delta\mathbf{b}$, we have:

$$Y\mathbf{a} = (\hat{E} + \delta E)(\hat{\mathbf{b}} + \delta\mathbf{b}) = \hat{E}(\hat{\mathbf{b}} + \delta\mathbf{b}) + \delta E\hat{\mathbf{b}} = \hat{E}\mathbf{b} + \delta E\hat{\mathbf{b}}, \quad \text{or}$$

$$Y\mathbf{a} - \hat{E}\mathbf{b} = \delta E\hat{\mathbf{b}} \tag{15.14.24}$$

The term $\delta E\hat{\mathbf{b}}$ represents the filtering of the vector $\delta\mathbf{e}$ by the filter $\hat{\mathbf{b}}$, and therefore, it can just as well be expressed in terms of the convolution matrix of $\hat{\mathbf{b}}$ acting on $\delta\mathbf{e}$, that is, $\delta E\hat{\mathbf{b}} = \hat{B}\delta\mathbf{e}$. Actually, $\hat{B}$ is the $N\times N$ square portion of the full convolution matrix, with the bottom q rows (the input-off transients) deleted. For example, with $N = 7$ and $q = 2$, we have:

$$\begin{bmatrix} \delta\epsilon_0 & 0 & 0 \\ \delta\epsilon_1 & \delta\epsilon_0 & 0 \\ \delta\epsilon_2 & \delta\epsilon_1 & \delta\epsilon_0 \\ \delta\epsilon_3 & \delta\epsilon_2 & \delta\epsilon_1 \\ \delta\epsilon_4 & \delta\epsilon_3 & \delta\epsilon_2 \\ \delta\epsilon_5 & \delta\epsilon_4 & \delta\epsilon_3 \\ \delta\epsilon_6 & \delta\epsilon_5 & \delta\epsilon_4 \end{bmatrix} \begin{bmatrix} 1 \\ \hat{b}_1 \\ \hat{b}_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hat{b}_1 & 1 & 0 & 0 & 0 & 0 & 0 \\ \hat{b}_2 & \hat{b}_1 & 1 & 0 & 0 & 0 & 0 \\ 0 & \hat{b}_2 & \hat{b}_1 & 1 & 0 & 0 & 0 \\ 0 & 0 & \hat{b}_2 & \hat{b}_1 & 1 & 0 & 0 \\ 0 & 0 & 0 & \hat{b}_2 & \hat{b}_1 & 1 & 0 \\ 0 & 0 & 0 & 0 & \hat{b}_2 & \hat{b}_1 & 1 \end{bmatrix} \begin{bmatrix} \delta\epsilon_0 \\ \delta\epsilon_1 \\ \delta\epsilon_2 \\ \delta\epsilon_3 \\ \delta\epsilon_4 \\ \delta\epsilon_5 \\ \delta\epsilon_6 \end{bmatrix}$$

The matrix $\hat{B}$ is invertible. Therefore, we may solve (15.14.24) for $\delta\mathbf{e}$. Defining $V = \hat{B}^{-1}Y$ and $W = \hat{B}^{-1}\hat{E}$, we have:

$$Y\mathbf{a} - \hat{E}\,\mathbf{b} = \delta E\,\hat{\mathbf{b}} = \hat{B}\delta\mathbf{e} \;\Rightarrow\; \hat{B}^{-1}(Y\mathbf{a} - \hat{E}\,\mathbf{b}) = \delta\mathbf{e} \;\Rightarrow\; V\mathbf{a} - W\,\mathbf{b} = \delta\mathbf{e} \tag{15.14.25}$$

Thus, the least-squares problem (15.14.21) is equivalent to minimizing the norm of the correction vector:

$$\mathcal{J} = \|V\mathbf{a} - W\,\mathbf{b}\|^2 = \|\delta\mathbf{e}\|^2 = \min \quad\Leftrightarrow\quad V\mathbf{a} = W\,\mathbf{b}$$

The operations $V = \hat{B}^{-1}Y$ and $W = \hat{B}^{-1}\hat{E}$ are equivalent to the inverse filtering operations (15.14.20). The MATLAB function **armamf** implements this three-step ARMA modeling algorithm. It has usage:

```
[a,b,sigma2] = armamf(y,p,q,M,iter);   % Mayne-Firoozan ARMA modeling
```

The third stage may be repeated a few additional times. At each iteration, the filtered signals $V(z) = Y(z)/B(z)$ and $W(z) = \hat{E}(z)/B(z)$ are obtained by using the filter $B(z)$ from the previous iteration. The parameter `iter` specifies the total number of iterations. The default value is `iter=2`.

The innovations variance σ_ϵ^2 is estimated by calculating the impulse response h_n of the designed ARMA filter, $H(z) = B(z)/A(z)$, and using:

$$\sigma_y^2 = \sigma_\epsilon^2 \sum_{n=0}^{\infty} |h_n|^2 \tag{15.14.26}$$

where the infinite summation may be approximated by a finite one of appropriate length—typically, a multiple of the 60-dB time-constant of the filter.

We have written a number of other MATLAB functions for MA and ARMA work. Examples of their usage are included in their respective help files.

```
h = arma2imp(a,b,N);            % ARMA impulse response
[a,b] = imp2arma(h,p,q);        % impulse response to ARMA coefficients
R = armaacf(a,b,s2,M);          % ARMA autocorrelation function
[A,B,D] = armachol(a,b,s2,N);   % ARMA covariance matrix Cholesky factorization
y = armasim(a,b,s2,N,seed);     % simulate an ARMA process using FILTER
y = armasim2(a,b,s2,N,seed);    % simulate an ARMA process using ARMACHOL
J = armainf(a,b);               % ARMA asymptotic Fisher information matrix
-------------------------------------------------------------------------------
[b,sig2] = mafit(R);            % fit MA(q) model to given covariance lags
[a,b,sig2] = armafit(R,p,q);    % fit ARMA(p,q) model to given covariance lags
-------------------------------------------------------------------------------
[b,sig2] = madurbin(y,q,M);     % MA modeling by Durbin's method
[b,sig2] = mainnov(y,q,M);      % MA modeling by the innovations method
-------------------------------------------------------------------------------
[a,b,sig2] = armainnov(y,p,q,M);     % ARMA modeling by innovations method
[a,b,sig2] = armamf(y,p,q,M);        % ARMA modeling by Mayne-Firoozan method
[a,b,sig2] = armamyw(y,p,q,Ma,Mb);   % ARMA modeling by modified Yule-Walker
-------------------------------------------------------------------------------
[B,D] = cholgs(R);       % Cholesky factorization by Gram-Schmidt method
[B,D] = cholinnov(R);    % Cholesky factorization by innovations method
```

15.15 Karhunen-Loève Transform

Traditionally, the Karhunen-Loève transform (KLT), also known as the Hotelling transform, of an $(M+1)$-dimensional stationary zero-mean random signal vector $\mathbf{y}(n) = [y_0(n), y_1(n), \dots, y_M(n)]^T$ with covariance matrix $R = E[\mathbf{y}^*(n)\mathbf{y}^T(n)]$ is defined as the linear transformation:

$$\boxed{\mathbf{z}(n) = V^T\mathbf{y}(n)} \qquad \text{(KLT)} \tag{15.15.1}$$

where V is the $(M+1)\times(M+1)$ unitary matrix of eigenvectors of R, that is,

$$V = [\mathbf{v}_0, \mathbf{v}_1, \dots, \mathbf{v}_M]\,, \quad R\mathbf{v}_i = \lambda_i\mathbf{v}_i\,, \quad i = 0, 1, \dots, M \tag{15.15.2}$$

with the eigenvalues λ_i assumed to be in decreasing order. The orthonormality of the eigenvectors $\mathbf{v}_i^\dagger\mathbf{v}_j = \delta_{ij}$ is equivalent to the unitarity of V,

$$V^\dagger V = VV^\dagger = I_{M+1} \tag{15.15.3}$$

The eigenvalue equations can be written compactly in the form:

$$RV = V\Lambda\,, \quad \Lambda = \text{diag}\{\lambda_0, \lambda_1, \dots, \lambda_M\} \quad\Rightarrow\quad V^\dagger RV = \Lambda \tag{15.15.4}$$

The components of the transformed vector, $\mathbf{z}(n) = [z_0(n), z_1(n), \dots, z_M(n)]^T$, are called *principal components*. They can be expressed as the dot products of the eigenvectors $\mathbf{v}_i$ with $\mathbf{y}(n)$:

$$\mathbf{z}(n) = V^T\mathbf{y}(n) \quad\Rightarrow\quad \begin{bmatrix} z_0(n) \\ z_1(n) \\ \vdots \\ z_M(n) \end{bmatrix} = \begin{bmatrix} \mathbf{v}_0^T\mathbf{y}(n) \\ \mathbf{v}_1^T\mathbf{y}(n) \\ \vdots \\ \mathbf{v}_M^T\mathbf{y}(n) \end{bmatrix}, \quad \text{or,}$$

$$\boxed{z_i(n) = \mathbf{v}_i^T\mathbf{y}(n)}\,, \quad i = 0, 1, \dots, M \tag{15.15.5}$$

These may be thought of as the filtering of $\mathbf{y}(n)$ by the FIR filters $\mathbf{v}_i$. Therefore, the vectors $\mathbf{v}_i$ are often referred to as *eigenfilters*. The principal components are mutually orthogonal, that is, uncorrelated. The matrix $V^\dagger RV = \Lambda$ is the covariance matrix of the transformed vector $\mathbf{z}(n)$:

$$E[\mathbf{z}^*(n)\mathbf{z}^T(n)] = V^\dagger E[\mathbf{y}^*(n)\mathbf{y}^T(n)]V = V^\dagger RV\,, \quad \text{or,}$$

$$\boxed{E[\mathbf{z}^*(n)\mathbf{z}^T(n)] = \Lambda} \tag{15.15.6}$$

or, component-wise:

$$\boxed{E[z_i^*(n)z_j(n)] = \lambda_i\delta_{ij}}\,, \quad i,j = 0, 1, \dots, M \tag{15.15.7}$$

Thus, the KLT decorrelates the components of the vector $\mathbf{y}(n)$. The eigenvalues of R are the variances of the principal components, $\sigma_i^2 = E[|z_i(n)|^2] = \lambda_i$. Because $\lambda_0 \geq \lambda_1 \geq \cdots \geq \lambda_M$, the principal component $z_0(n)$ will have the largest variance, the component $z_1(n)$, the next to largest, and so on.

Defining the total variance of $\mathbf{y}(n)$ to be the sum of the variances of its $M+1$ components, we can show that the total variance is equal to the sum of the variances of the principal components, or the sum of the eigenvalues of R. We have:

$$\sigma_y^2 = \sum_{i=0}^{M} E[|y_i(n)|^2] = E[\mathbf{y}^\dagger(n)\mathbf{y}(n)] \qquad \text{(total variance)} \tag{15.15.8}$$

Using the trace property $\mathbf{y}^\dagger\mathbf{y} = \mathrm{tr}(\mathbf{y}^*\mathbf{y}^T)$, we find:

$$\sigma_y^2 = \mathrm{tr}\big(E[\mathbf{y}^*(n)\mathbf{y}^T(n)]\big) = \mathrm{tr}(R) = \lambda_0 + \lambda_1 + \cdots + \lambda_M \tag{15.15.9}$$

The inverse Karhunen-Loève transform is obtained by noting that $V^{-T} = V^*$, which follows from $V^\dagger V = I$. Therefore,

$$\boxed{\mathbf{y}(n) = V^*\mathbf{z}(n)} \qquad \text{(inverse KLT)} \tag{15.15.10}$$

It can be written as a sum of the individual principal components:

$$\mathbf{y}(n) = V^*\mathbf{z}(n) = [\mathbf{v}_0^*, \mathbf{v}_1^*, \dots, \mathbf{v}_M^*]\begin{bmatrix} z_0(n) \\ z_1(n) \\ \vdots \\ z_M(n) \end{bmatrix} = \sum_{i=0}^{M} \mathbf{v}_i^* z_i(n) \tag{15.15.11}$$

In many applications, the first few principal components, $z_i(n)$, $0 \le i \le r-1$, where $r \ll M+1$, account for most of the total variance. In such cases, we may keep only the first r terms in the inverse transform:

$$\hat{\mathbf{y}}(n) = \sum_{i=0}^{r-1} \mathbf{v}_i^* z_i(n) \tag{15.15.12}$$

If the ignored eigenvalues are small, the reconstructed signal $\hat{\mathbf{y}}(n)$ will be a good approximation of the original $\mathbf{y}(n)$. This approximation amounts to a rank-r reduction of the original problem. The mean-square approximation error is:

$$E[\|\mathbf{y}(n) - \hat{\mathbf{y}}(n)\|^2] = E\Big[\sum_{i=r}^{M} |z_i(n)|^2\Big] = \sum_{i=r}^{M} \lambda_i \tag{15.15.13}$$

15.16 Principal Component Analysis

Principal component analysis (PCA) is essentially equivalent to the KLT. The only difference is that instead of applying the KLT to the theoretical covariance matrix R, it is applied to the sample covariance matrix $\hat{R}$ constructed from N available signal vectors $\mathbf{y}(n)$, $n = 0, 1, \dots, N-1$:

$$\hat{R} = \frac{1}{N}\sum_{n=0}^{N-1} \mathbf{y}^*(n)\mathbf{y}^T(n) \tag{15.16.1}$$

where we assume that the sample means have been removed, so that

$$\mathbf{m} = \frac{1}{N}\sum_{n=0}^{N-1} \mathbf{y}(n) = 0$$

We will ignore the overall factor $1/N$, as we did in section 15.12, and work with the simpler definition:

$$\hat{R} = \sum_{n=0}^{N-1} \mathbf{y}^*(n)\mathbf{y}^T(n) = Y^\dagger Y, \qquad Y = \begin{bmatrix} \mathbf{y}^T(0) \\ \mathbf{y}^T(1) \\ \vdots \\ \mathbf{y}^T(N-1) \end{bmatrix} \tag{15.16.2}$$

where Y is the $N\times(M+1)$ data matrix constructed from the $\mathbf{y}(n)$. The eigenproblem of $\hat{R}$, that is, $\hat{R}V = V\Lambda$, defines the KLT/PCA transformation matrix V. The corresponding principal component signals will be:

$$\mathbf{z}(n) = V^T\mathbf{y}(n), \quad n = 0, 1, \dots, N-1 \tag{15.16.3}$$

These can be combined into a single compact equation involving the data matrix constructed from the $\mathbf{z}(n)$. Indeed, noting that $\mathbf{z}^T(n) = \mathbf{y}^T(n)V$, we have:

$$\boxed{Z = YV} \qquad \text{(PCA)} \tag{15.16.4}$$

where Z is the $N\times(M+1)$ data matrix of the $\mathbf{z}(n)$:

$$Z = \begin{bmatrix} \mathbf{z}^T(0) \\ \mathbf{z}^T(1) \\ \vdots \\ \mathbf{z}^T(N-1) \end{bmatrix} \tag{15.16.5}$$

The inverse transform can be obtained by multiplying (15.16.4) by $V^\dagger$ from the right and using the unitarity property of V, that is, $ZV^\dagger = YVV^\dagger$, or,

$$Y = ZV^\dagger \quad \Rightarrow \quad \mathbf{y}(n) = V^*\mathbf{z}(n), \quad n = 0, 1, \dots, N-1 \tag{15.16.6}$$

or, explicitly in terms of the PCA signals $z_i(n)$:

$$\mathbf{y}(n) = \sum_{i=0}^{M} \mathbf{v}_i^* z_i(n), \quad n = 0, 1, \dots, N-1 \tag{15.16.7}$$

The uncorrelatedness property of the KLT translates now to the orthogonality of the signals $z_i(n) = \mathbf{v}_i^T\mathbf{y}(n)$ as functions of time. It follows from (15.16.4) that Z has orthogonal columns, or equivalently, a diagonal sample covariance matrix:

$$Z^\dagger Z = V^\dagger \hat{R} V = \Lambda \quad \Rightarrow \quad \sum_{n=0}^{N-1} \mathbf{z}^*(n)\mathbf{z}^T(n) = \Lambda \tag{15.16.8}$$

or, written component-wise:

$$\boxed{\sum_{n=0}^{N-1} z_i^*(n)\,z_j(n) = \lambda_i\delta_{ij}}\,, \quad i,j = 0,1,\dots,M \tag{15.16.9}$$

In fact, the principal component signals $z_i(n)$ are, up to a scale, equal to the left singular eigenvectors of the SVD of the data matrix Y.

Following the simplified proof of the SVD that we gave in Sec. 15.5, we assume a full-rank case so that all the λ_i are nonzero and define the singular values $\sigma_i = \sqrt{\lambda_i}$, for $i = 0,1,\dots,M$, and the matrices:

$$U = Z\Sigma^{-1}\,, \quad \Sigma = \text{diag}\{\sigma_0,\sigma_1,\dots,\sigma_M\} = \Lambda^{1/2} \tag{15.16.10}$$

where U,Σ have sizes $N\times(M+1)$ and $(M+1)\times(M+1)$, respectively. It follows from (15.16.8) that U has orthonormal columns:

$$U^\dagger U = \Sigma^{-1}Z^\dagger Z\Sigma^{-1} = \Lambda^{-1/2}\Lambda\Lambda^{1/2} = I_{M+1} \tag{15.16.11}$$

Solving for Y in terms of U, we obtain the economy SVD of Y. Indeed, we have $Z = U\Sigma$ and $Y = ZV^\dagger$, so that

$$\boxed{Y = U\Sigma V^\dagger} \quad \text{(economy SVD)} \tag{15.16.12}$$

Thus, principal component analysis based on $\hat{R}$ is equivalent to performing the economy SVD of the data matrix Y.

The matrix U has the same size as Y, but mutually orthogonal columns. The $(M+1)$-dimensional vectors $\mathbf{u}(n) = \Sigma^{-1}\mathbf{z}(n) = \Sigma^{-1}V^T\mathbf{y}(n)$, $n = 0,1,\dots,N-1$, have U as their data matrix and correspond to normalized versions of the principal components with unit sample covariance matrix:

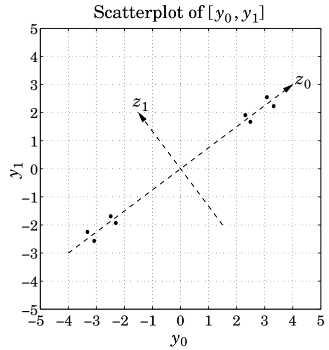
$$U^\dagger U = \sum_{n=0}^{N-1}\mathbf{u}^*(n)\mathbf{u}^T(n) = I_{M+1} \quad \Leftrightarrow \quad \sum_{n=0}^{N-1} u_i^*(n)\,u_j(n) = \delta_{ij}$$

where $u_i(n)$ is the ith component of $\mathbf{u}(n) = [u_0(n), u_1(n),\dots,u_M(n)]^T$. It is the same as $z_i(n)$, but normalized to unit norm.

Example 15.16.1: Consider the following 8×2 data matrix Y and its economy SVD:

$$Y = \begin{bmatrix} 2.31 & 1.92 \\ 2.49 & 1.68 \\ -2.31 & -1.92 \\ -2.49 & -1.68 \\ 3.32 & 2.24 \\ -3.08 & -2.56 \\ 3.08 & 2.56 \\ -3.32 & -2.24 \end{bmatrix} = \begin{bmatrix} 0.3 & 0.3 \\ 0.3 & -0.3 \\ -0.3 & -0.3 \\ -0.3 & 0.3 \\ 0.4 & -0.4 \\ -0.4 & -0.4 \\ 0.4 & 0.4 \\ -0.4 & 0.4 \end{bmatrix} \begin{bmatrix} 10 & 0 \\ 0 & 0.5 \end{bmatrix} \begin{bmatrix} 0.8 & -0.6 \\ 0.6 & 0.8 \end{bmatrix}^T = U\Sigma V^T$$

The singular values of Y are $\sigma_0 = 10$ and $\sigma_1 = 0.5$. Let the two columns of Y be $\mathbf{y}_0$ and $\mathbf{y}_1$, so that $Y = [\mathbf{y}_0, \mathbf{y}_1]$. The scatterplot of the eight pairs $[y_0, y_1]$ is shown below. We observe the clustering along a preferential direction. This is the direction of the first principal component.



The corresponding 8×2 matrix of principal components and its diagonal covariance matrix are:

$$Z = [\mathbf{z}_0, \mathbf{z}_1] = U\Sigma = \begin{bmatrix} 3 & 0.15 \\ 3 & -0.15 \\ -3 & -0.15 \\ -3 & 0.15 \\ 4 & -0.20 \\ -4 & -0.20 \\ 4 & 0.20 \\ -4 & 0.20 \end{bmatrix}, \quad \Lambda = Z^T Z = \begin{bmatrix} \sigma_0^2 & 0 \\ 0 & \sigma_1^2 \end{bmatrix} = \begin{bmatrix} 100 & 0 \\ 0 & 0.25 \end{bmatrix}$$

The covariance matrix of Y, $R = Y^T Y$, is diagonalized by the matrix V:

$$R = \begin{bmatrix} 64.09 & 47.88 \\ 47.88 & 36.16 \end{bmatrix} = \begin{bmatrix} 0.8 & -0.6 \\ 0.6 & 0.8 \end{bmatrix} \begin{bmatrix} 100 & 0 \\ 0 & 0.25 \end{bmatrix} \begin{bmatrix} 0.8 & -0.6 \\ 0.6 & 0.8 \end{bmatrix}^T = V\Lambda V^T$$

Each principal component pair $[z_0, z_1]$ is constructed by the following linear combinations of the $[y_0, y_1]$ pairs:

$$z_0 = \mathbf{v}_0^T\mathbf{y} = [0.8, 0.6]\begin{bmatrix} y_0 \\ y_1 \end{bmatrix} = 0.8y_0 + 0.6y_1$$

$$z_1 = \mathbf{v}_1^T\mathbf{y} = [-0.6, 0.8]\begin{bmatrix} y_0 \\ y_1 \end{bmatrix} = -0.6y_0 + 0.8y_1$$

Conversely, each $[y_0, y_1]$ pair may be reconstructed from the PCA pair $[z_0, z_1]$:

$$\begin{bmatrix} y_0 \\ y_1 \end{bmatrix} = V^*\mathbf{z} = [\mathbf{v}_0^*, \mathbf{v}_1^*]\begin{bmatrix} z_0 \\ z_1 \end{bmatrix} = \mathbf{v}_0^* z_0 + \mathbf{v}_1^* z_1 = \begin{bmatrix} 0.8 \\ 0.6 \end{bmatrix} z_0 + \begin{bmatrix} -0.6 \\ 0.8 \end{bmatrix} z_1$$

The two terms in this expression define parametrically two straight lines on the y_0, y_1 plane along the directions of the principal components, as shown in the above figure. The percentage variances carried by z_0, z_1 are:

$$\frac{\sigma_0^2}{\sigma_0^2+\sigma_1^2} = 0.9975 = 99.75\,\%\,,\qquad \frac{\sigma_1^2}{\sigma_0^2+\sigma_1^2} = 0.0025 = 0.25\,\%$$

This explains the clustering along the z_0 direction. □

Example 15.16.2: The table below gives $N = 24$ values of the signals $\mathbf{y}^T(n) = [y_0(n), y_1(n)]$. The data represent the measured lengths and widths of 24 female turtles and were obtained from the file **turtle.dat** on the book's web page. This data set represents one of the most well-known examples of PCA [1306]. To simplify the discussion, we consider only a subset of this data set.

The data matrix Y has dimension $N\times(M+1) = 24\times 2$. It must be replaced by its zero-mean version, that is, with the column means removed from each column. Fig. 15.16.1 shows the scatterplot of the pairs $[y_0, y_1]$.

n	$y_0(n)$	$y_1(n)$	n	$y_0(n)$	$y_1(n)$	n	$y_0(n)$	$y_1(n)$
0	98	81	8	133	102	16	149	107
1	103	84	9	133	102	17	153	107
2	103	86	10	134	100	18	155	115
3	105	86	11	136	102	19	155	117
4	109	88	12	137	98	20	158	115
5	123	92	13	138	99	21	159	118
6	123	95	14	141	103	22	162	124
7	133	99	15	147	108	23	177	132

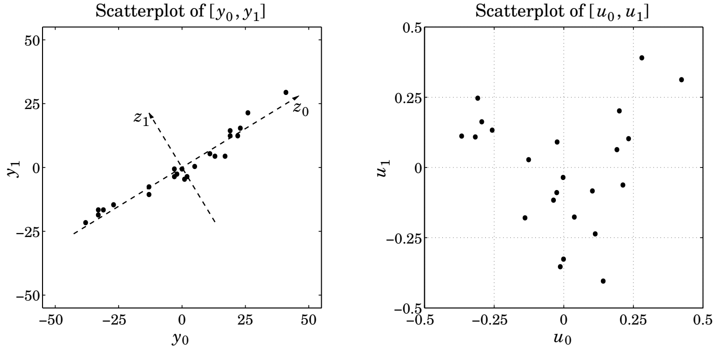


Fig. 15.16.1 Scatterplots of original data and their principal components.

We observe that the pairs are distributed essentially one-dimensionally along a particular direction, which is the direction of the first principal component.

Performing the economy SVD on (the zero-mean version of) Y gives the singular values $\sigma_0 = 119.05$ and $\sigma_1 = 12.38$, and the unitary PCA transformation matrix V:

$$V = [\mathbf{v}_0, \mathbf{v}_1] = \begin{bmatrix} 0.8542 & -0.5200 \\ 0.5200 & 0.8542 \end{bmatrix},\quad \mathbf{v}_0 = \begin{bmatrix} 0.8542 \\ 0.5200 \end{bmatrix},\quad \mathbf{v}_1 = \begin{bmatrix} -0.5200 \\ 0.8542 \end{bmatrix}$$

The total variance is $\sigma_y^2 = \sigma_0^2 + \sigma_1^2$. The percentages of this variance carried by the two principal components are:

$$\frac{\sigma_0^2}{\sigma_0^2+\sigma_1^2} = 0.989 = 98.9\,\%\,,\qquad \frac{\sigma_1^2}{\sigma_0^2+\sigma_1^2} = 0.011 = 1.1\,\%$$

Thus, the principal component z_0 carries the bulk of the variance. The two principal components are obtained by the linear combinations $\mathbf{z} = V^T\mathbf{y}$, or,

$$\begin{aligned} z_0 &= \mathbf{v}_0^T\mathbf{y} = 0.8542\,y_0 + 0.52\,y_1 \\ z_1 &= \mathbf{v}_1^T\mathbf{y} = -0.52\,y_0 + 0.8542\,y_1 \end{aligned}$$

The inverse relationships are $\mathbf{y} = V^*\mathbf{z} = \mathbf{v}_0^* z_0 + \mathbf{v}_1^* z_1$, or,

$$\begin{bmatrix} y_0 \\ y_1 \end{bmatrix} = \begin{bmatrix} 0.8542 \\ 0.5200 \end{bmatrix} z_0 + \begin{bmatrix} -0.5200 \\ 0.8542 \end{bmatrix} z_1$$

The two terms represent the projections of $\mathbf{y}$ onto the two PCA directions. The two straight lines shown in Fig. 15.16.1 are given by these two terms separately, where z_0 and z_1 can be used to parametrize points along these lines.

The MATLAB code used to generate this example was as follows:

```
Y = loadfile('turtle.dat');          % read full data set
Y = zmean(Y(:,4:5));                 % get columns 4,5 and remove column means

[U,S,V] = svd(Y,0);                      % economy SVD

figure; plot(Y(:,1),Y(:,2),'.');         % scatterplot of [y0,y1]
figure; plot(U(:,1),U(:,2),'.');         % scatterplot of [u0,u1]
```

The right graph in Fig. 15.16.1 is the scatterplot of the columns of U, that is, the unit-norm principal components $u_0(n), u_1(n)$, $n = 0, 1, \dots, N-1$. Being mutually uncorrelated, they do not exhibit clustering along any special directions. □

Several applications of PCA in diverse fields, such as statistics, physiology, psychology, meteorology, and computer vision, are discussed in [1237–1239,1241–1244,1303–1314].

15.17 SVD Signal Enhancement

The main idea of PCA is rank reduction for the purpose of reducing the dimensionality of the problem. In many signal processing applications, such as sinusoids in noise, or plane waves incident on an array, the noise-free signal has a data matrix of reduced rank. For example, the rank is equal to the number of (complex) sinusoids that are present. We will be discussing this in detail later.

The presence of noise causes the data matrix to become full rank. Forcing the rank back to what it is supposed to be in the absence of noise has a beneficial noise-reduction or signal-enhancement effect. However, rank-reduction destroys any special structure that the data matrix might have, for example, being Toeplitz or Toeplitz over Hankel. A further step is required after rank reduction that restores the special structure of the matrix. But when the structure is restored, the rank becomes full again. Therefore, one must iterate this process of rank-reduction followed by structure restoration.

Given an initial data matrix of a given type, such as the autocorrelation, covariance, or forward/backward type, the following steps implement the typical SVD enhancement iteration:

```
Y = datamat(y,M,type);    % construct data matrix from signal y
Ye = Y;                   % initialize enhancement iteration
for i=1:K,                % iterate K times, typically, K=2-3.
   Ye = sigsub(Ye,r);     % force rank reduction to r
   Ye = toepl(Ye,type);   % restore Toeplitz/Toeplitz-Hankel structure
end
ye = datasig(Ye,type);    % extract enhanced signal from Ye
```

After the iteration, one may extract the “enhanced” signal from the enhanced data matrix. The MATLAB function **sigsub**, introduced in Sec. 15.9, carries out an economy SVD of Y and then keeps only the r largest singular values, that is, it extracts the signal subspace part of Y. The function **toepl**, discussed in Sec. 15.18, restores the Toeplitz or Toeplitz-over-Hankel structure by finding the matrix with such structure that lies closest to the rank-reduced data matrix.

The SVD enhancement iteration method has been re-invented in different contexts. In the context of linear prediction and extracting sinusoids in noise it is known as the *Cadzow iteration* [1268–1270]. In the context of chaotic dynamics, climatology, and meteorology, it is known as *singular spectral analysis* (SSA)[†] [1322–1337]; actually, in SSA only one iteration ($K = 1$) is used. In nonlinear dynamics, the process of forming the data matrix Y is referred to as “delay-coordinate embedding” and the number of columns, $M + 1$, of Y is the “embedding dimension.”

In the literature, one often finds that the data matrix Y is defined as a Hankel instead of a Toeplitz matrix. This corresponds to reversing the rows of the Toeplitz definition. For example, using the reversing matrix J, we have:

$$Y = \begin{bmatrix} y_2 & y_1 & y_0 \\ y_3 & y_2 & y_1 \\ y_4 & y_3 & y_2 \\ y_5 & y_4 & y_3 \\ y_6 & y_5 & y_2 \end{bmatrix} = \text{Toeplitz} \quad \Rightarrow \quad YJ = \begin{bmatrix} y_0 & y_1 & y_2 \\ y_1 & y_2 & y_3 \\ y_2 & y_3 & y_4 \\ y_3 & y_4 & y_5 \\ y_4 & y_5 & y_6 \end{bmatrix} = \text{Hankel}$$

In such cases, in the SVD enhancement iterations one must invoke the function **toepl** with its Hankel option, that is, `type=1`.

Example 15.17.1: As an example that illustrates the degree of enhancement obtained from such methods, consider the length-25 signal y_n listed in the file **sine1.dat** on the book’s web page. The signal consists of two equal-amplitude sinusoids of frequencies $f_1 = 0.20$ and $f_2 = 0.25$ cycles/sample, in zero-mean, white gaussian noise with a 0-dB SNR. The signal samples were generated by:

$$y_n = \cos(2\pi f_1 n) + \cos(2\pi f_2 n) + 0.707\, v_n\,, \quad n = 0, 1, \dots, 24$$

[†] Sometimes also called “singular system analysis” or the “caterpillar” method.

where v_n is zero-mean, unit-variance, white noise, and the amplitude $1/\sqrt{2} = 0.707$ ensures that $SNR = 0$ dB.

The short duration and the low SNR make this a difficult signal to handle. Fig. 15.17.1 compares the performance of four spectrum estimation methods: the ordinary periodogram, the linear-prediction-based methods of Burg and Yule-Walker, and the SVD-enhanced Burg method in which the SVD-enhanced signal is subjected to Burg’s algorithm, instead of the original signal.

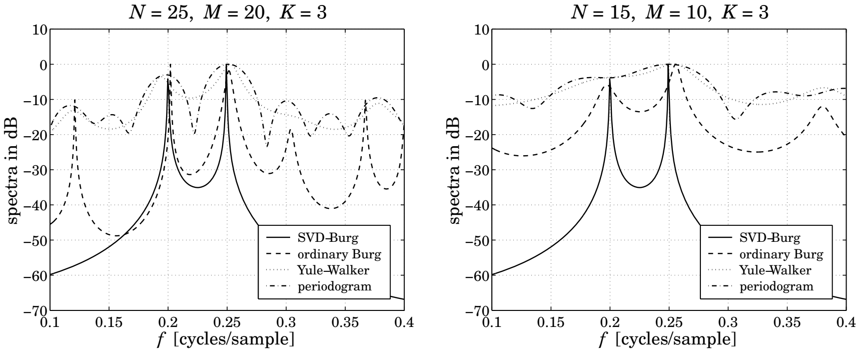


Fig. 15.17.1 SVD-enhanced linear prediction spectra.

The effective rank is $r = 4$ (each real sinusoid counts for two complex ones.) The SVD-enhanced version of Burg’s method gives narrow peaks at the two desired frequencies. The number of iterations was $K = 3$, and the prediction filter order $M = 20$.

The Yule-Walker method results in fairly wide peaks at the two frequencies—the SNR is just too small for the method to work. The ordinary Burg method gives narrower peaks, but because the filter order M is high, it also produces several false peaks that are just as narrow.

Reducing the order of the prediction filter from M down to r, as is done in the SVD method to avoid any false peaks, will not work at all for the Yule-Walker and ordinary Burg methods—both will fail to resolve the peaks.

The periodogram exhibits wide mainlobes and sidelobes—the signal duration is just too short to make the mainlobes narrow enough. If the signal is windowed prior to computing the periodogram, for example, using a Hamming window, the two mainlobes will broaden so much that they will overlap with each other, masking completely the frequency peaks.

The graph on the right of Fig. 15.17.1 makes the length even shorter, $N = 15$, by using only the first 15 samples of y_n. The SVD method, implemented with $M = 10$, still exhibits the two narrow peaks, whereas all of the other methods fail, with the ordinary Burg being a little better than the others, but still exhibiting a false peak. The SVD method works well also for $K = 2$ iterations, but not so well for $K = 1$. The following MATLAB code illustrates the computational steps for producing these graphs:

```
y = loadfile('sine1.dat');            % read signal samples y_n from file

r = 4; M = 20; K = 3;                 % rank, filter order, number of iterations
```

```
f = linspace(0.1,0.4,401); w = 2*pi*f;            % frequency band

a = lpf(burg(y,M));                         % Burg prediction filter of order M
H1 = 1./abs(dtft(a,w));                     % compute ordinary Burg LP spectrum
H1 = 20*log10(H1/max(H1));                  % spectrum in dB

a = lpf(yw(y,M));                           % Yule-Walker prediction filter
H2 = 1./abs(dtft(a,w));                     % compute Yule-Walker LP spectrum
H2 = 20*log10(H2/max(H2));

H3 = abs(dtft(y,w));
H3 = 20*log10(H3/max(H3));                  % periodogram spectrum in dB

Y = datamat(y,M);                           % Y is the autocorrelation type
Ye = Y;
for i=1:K,                                  % SVD enhancement iterations
    Ye = sigsub(Ye,r);                      % set rank to r
    Ye = toepl(Ye);                         % toeplitzize Ye
end
ye = datasig(Ye);                           % extract enhanced time signal

a = lpf(burg(ye,r));                        % Burg prediction filter of order r
H = 1./abs(dtft(a,w));                      % compute enhanced Burg LP spectrum
H = 20*log10(H/max(H));

plot(f,H,'-', f,H1,'--', f,H2,':', f,H3,'-.');
```

The functions **lpf**, **burg**, **yw** implement the standard Burg and Yule-Walker methods. □

Example 15.17.2: The SVD enhancement process can be used to smooth data and extract local or global trends from noisy times series. Typically, the first few principal components represent the trend.

As an example, we consider the global annual average temperature obtained from the web site: `www.cru.uea.ac.uk/cru/data/temperature/`. The data represent the temperature anomalies in degrees ^oC with respect to the 1961–1990 average.

Using $M = 30$ and one SVD enhancement iteration, $K = 1$, we find the first five variances, given as percentages of the total variance:

$$\{\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5\} = \{63.78,\ 12.44,\ 2.27,\ 1.79,\ 1.71\}$$

The first two PCs account for 76% of the total variance. The percent variances are plotted in Fig. 15.17.2.

The smoothed signals extracted from reducing the rank to $r = 1, 2, 3, 4, 5, 6$ are shown in Figs. 15.17.3, 15.17.4, and 15.17.5. We note that the $r = 2$ case represents the trend well. As the rank is increased, the smoothed signal tries to capture more and more of the finer variations of the original signal.

Assuming that the global trend $y_e(n)$ is represented by the first two principal components ($r = 2$), one can subtract it from the original sequence resulting into the residual $y_1(n) = y(n) - y_e(n)$, and the SVD enhancement method may be repeated on that signal. The first few components of $y_1(n)$ can be taken to represent the local variations in the original $y(n)$, such as short-period cyclical components. The rest of the principal components of $y_1(n)$ may be taken to represent the noise.

The MATLAB code used to generate these graphs was as follows:

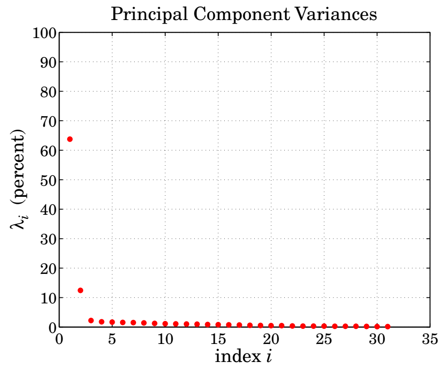


Fig. 15.17.2 Percentage variances of the first 31 principal components.

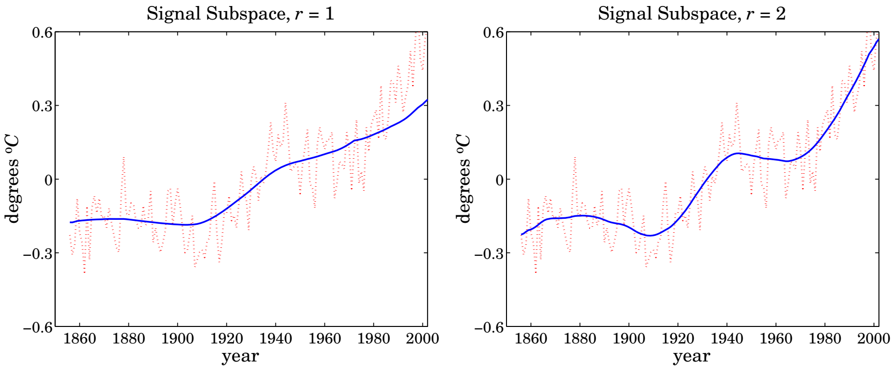


Fig. 15.17.3 Principal component signals of ranks $r = 1, 2$.

```
A = loadfile('TaveGL2.dat');       % read data file
y = A(:,14);                       % column-14 holds the annual averages
n = A(:,1);                        % column-1 holds the year

M = 30; K=1; r = 1;                % or, r = 2, 3, 4, 5, 6
y = zmean(y);                      % zero mean
Ye = datamat(y,M,2);               % forward-backward Toeplitz-Hankel type
for i=1:K,                         % SVD enhancement iteration
    Ye = sigsub(Ye,r);             % extract rank-r signal subspace
    Ye = toepl(Ye,2);              % convert to Toeplitz-Hankel
end
ye = datasig(Ye,2);                % extract smoothed signal

plot(n,y,':', n,ye,'-');           % plot original and smoothed signal
```

For comparison, we show in Fig. 15.17.6, the Whittaker-Henderson smoothing method, which appears to have comparable performance with the SVD method. The MATLAB code

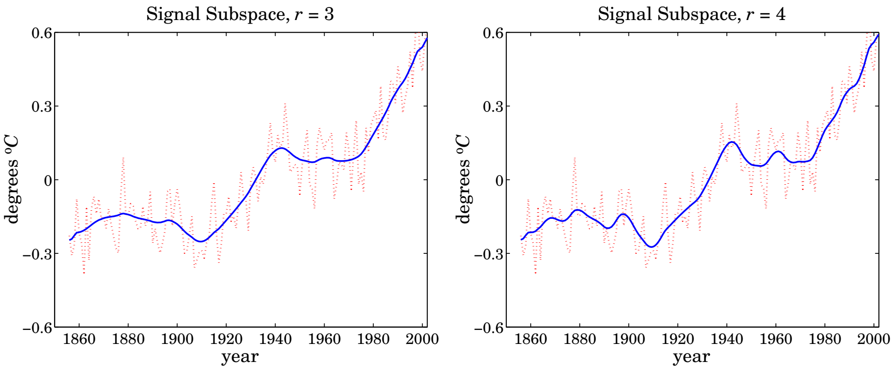


Fig. 15.17.4 Principal component signals of ranks $r = 3, 4$.

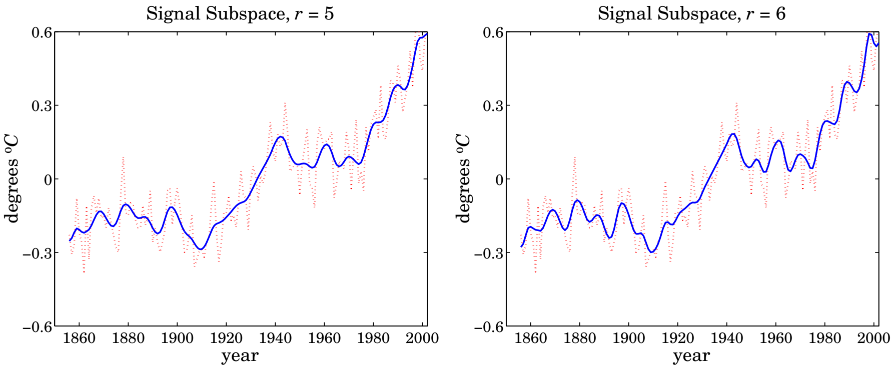


Fig. 15.17.5 Principal component signals of ranks $r = 5, 6$.

for that was,

```
    lambda = 10000;
    ywh = whsm(y,lambda,3);
    plot(n,y,'r:', n,ywh,'b-');
```

Here, the degree of smoothing is controlled by the regularization parameter λ. □

15.18 Structured Matrix Approximations

We saw in the previous section that the process of rank reduction destroys the Toeplitz or Toeplitz/Hankel nature of the data matrix. The purpose of the MATLAB function **toepl** was to restore the structure of the data matrix by finding the closest matrix of the desired structure.

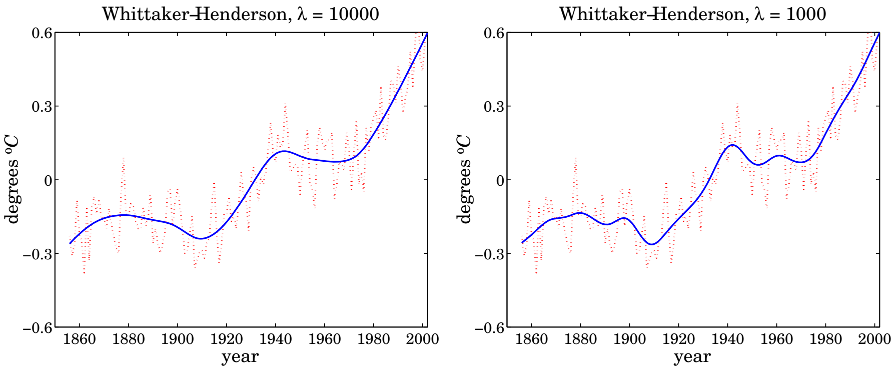


Fig. 15.17.6 Whittaker-Henderson smoothing method.

Given a data matrix Y that ideally should be Toeplitz, such as the autocorrelation or covariance types, one can find a Toeplitz matrix T that is closest to Y with respect to a matrix norm. The easiest norm to use is the Frobenius norm. Thus, we have the approximation problem:

$$\mathcal{J} = \|Y - T\|_F^2 = \min, \quad \text{where } T \text{ is required to be Toeplitz} \tag{15.18.1}$$

The solution is the Toeplitz matrix obtained by replacing each diagonal of Y by the average along that diagonal. We demonstrate this with a small example. Let Y and T be defined as:

$$Y = \begin{bmatrix} y_{11} & y_{12} & y_{13} \\ y_{21} & y_{22} & y_{23} \\ y_{31} & y_{32} & y_{33} \end{bmatrix}, \quad T = \begin{bmatrix} t_2 & t_1 & t_0 \\ t_3 & t_2 & t_1 \\ t_4 & t_3 & t_2 \end{bmatrix}$$

The difference matrix is:

$$Y - T = \begin{bmatrix} y_{11} - t_2 & y_{12} - t_1 & y_{13} - t_0 \\ y_{21} - t_3 & y_{22} - t_2 & y_{23} - t_1 \\ y_{31} - t_4 & y_{32} - t_3 & y_{33} - t_2 \end{bmatrix}$$

Because the Frobenius norm is the sum of the squares of all the matrix elements, we have:

$$\begin{aligned} \mathcal{J} = \|Y - T\|_F^2 = & |y_{11} - t_2|^2 + |y_{22} - t_2|^2 + |y_{33} - t_2|^2 \\ & + |y_{12} - t_1|^2 + |y_{23} - t_1|^2 + |y_{13} - t_0|^2 \\ & + |y_{21} - t_3|^2 + |y_{32} - t_3|^2 + |y_{13} - t_4|^2 \end{aligned}$$

The minimization conditions $\partial\mathcal{J}/\partial t_i = 0$, $i = 0, 1, 2, 3, 4$, easily lead to the desired solutions: $t_0 = y_{13}$, $t_4 = y_{31}$ and

$$t_1 = \frac{y_{12} + y_{23}}{2}, \quad t_2 = \frac{y_{11} + y_{22} + y_{33}}{3}, \quad t_3 = \frac{y_{21} + y_{32}}{2}$$

For a Hankel matrix approximation, we have the minimization problem:

$$\mathcal{J} = \|Y - H\|_F^2 = \min, \quad \text{where } H \text{ is required to be Hankel} \tag{15.18.2}$$

Its solution is obtained by replacing each antidiagonal of Y by the average along that antidiagonal. This problem can be reduced to an equivalent Toeplitz type by noting that the row-reversing operation $Y \to YJ$, where J is the usual reversing matrix, leaves the Frobenius norm unchanged and it maps a Hankel matrix into a Toeplitz one. Setting $T = HJ$, the problem (15.18.2) becomes:

$$\mathcal{J} = \|Y - H\|_F^2 = \|YJ - T\|_F^2 = \min, \quad \text{where } T \text{ is required to be Toeplitz} \tag{15.18.3}$$

Once T is found by averaging the diagonals of YJ, the Hankel matrix H is constructed by row-reversal, $H = TJ$. This amounts to averaging the antidiagonals of the original data matrix Y.

Finally, in the case of Toeplitz over Hankel structure, we have a data matrix whose upper half is to be Toeplitz and its lower half is the row-reversed and conjugated upper part. Partitioning Y into these two parts, we set:

$$Y = \left[\begin{array}{c} Y_T \\ \hline Y_H \end{array}\right], \quad M = \left[\begin{array}{c} T \\ \hline T^* J \end{array}\right] = \text{required approximation}$$

The matrix approximation problem is then:

$$\mathcal{J} = \|Y - M\|_F^2 = \left\| \left[\begin{array}{c} Y_T \\ \hline Y_H \end{array}\right] - \left[\begin{array}{c} T \\ \hline T^* J \end{array}\right] \right\|_F^2 = \|Y_T - T\|_F^2 + \|Y_H^* J - T\|_F^2 = \min$$

where we used the property $\|Y_H - T^*J\|_F^2 = \|Y_H^*J - T\|_F^2$. The solution of this minimization problem is obtained by choosing T to be the average of the Toeplitz approximations of Y_T and Y_H^*J, that is, in the notation of the function **toepl**:

$$T = \frac{\text{toepl}(Y_T) + \text{toepl}(Y_H^* J)}{2}$$

Example 15.18.1: As an example, we give below the optimum Toeplitz, Hankel, and Toeplitz over Hankel approximations of the same data matrix Y:

$$Y = \begin{bmatrix} 10 & 10 & 10 \\ 20 & 20 & 20 \\ 30 & 30 & 30 \\ 40 & 40 & 40 \\ 50 & 50 & 50 \\ 60 & 60 & 60 \end{bmatrix} \quad \Rightarrow \quad T = \begin{bmatrix} 20 & 15 & 10 \\ 30 & 20 & 15 \\ 40 & 30 & 20 \\ 50 & 40 & 30 \\ 55 & 50 & 40 \\ 60 & 55 & 50 \end{bmatrix}, \quad H = \begin{bmatrix} 10 & 15 & 20 \\ 15 & 20 & 30 \\ 20 & 30 & 40 \\ 30 & 40 & 50 \\ 40 & 50 & 55 \\ 50 & 55 & 60 \end{bmatrix}$$

$$Y = \left[\begin{array}{ccc} 10 & 10 & 10 \\ 20 & 20 & 20 \\ 30 & 30 & 30 \\ \hline 40 & 40 & 40 \\ 50 & 50 & 50 \\ 60 & 60 & 60 \end{array}\right] \quad \Rightarrow \quad M = \left[\begin{array}{ccc} 35 & 30 & 25 \\ 40 & 35 & 30 \\ 45 & 40 & 35 \\ \hline 25 & 30 & 35 \\ 30 & 35 & 40 \\ 35 & 40 & 45 \end{array}\right] = \left[\begin{array}{c} T \\ \hline T^* J \end{array}\right]$$

The function **toepl** has usage:

```
Z = toepl(Y,type);   % structured approximation of a data matrix
```

```
Y = data matrix
type=0: Toeplitz type, each diagonal of Y is replaced by its average
type=1: Hankel type, each anti-diagonal of Y is replaced by its average
type=2: Toeplitz over Hankel, Y must have even number of rows
```

15.19 Matrix Pencil Methods

The *matrix pencil* of two $N \times M$ matrices A, B, is defined to be the matrix:

$$A - \lambda B \tag{15.19.1}$$

where λ is a parameter. The *generalized eigenvalues* of the matrix pair $\{A, B\}$ are those values of λ that cause $A - \lambda B$ to reduce its rank. A *generalized eigenvector* corresponding to such a λ is a vector in the null space $N(A - \lambda B)$.

A matrix pencil is a generalization of the eigenvalue concept to non-square matrices. A similar rank reduction takes place in the ordinary eigenvalue problem of a square matrix. Indeed, the eigenvalue-eigenvector condition $A\mathbf{v}_i = \lambda_i \mathbf{v}_i$ can be written as $(A - \lambda_i I)\mathbf{v}_i = 0$, which states that $A - \lambda I$ loses its rank when $\lambda = \lambda_i$ and $\mathbf{v}_i$ lies in the null space $N(A - \lambda_i I)$.

Matrix pencil methods arise naturally in the problem of estimating damped or undamped sinusoids in noise [1280], and are equivalent to the so-called ESPRIT methods [1276]. Consider a signal that is the sum of r, possibly damped, complex sinusoids in additive, zero-mean, white noise:

$$\boxed{y_n = \sum_{i=1}^{r} A_i e^{-\alpha_i n} e^{j\omega_i n} + v_n = \sum_{i=1}^{r} A_i z_i^n + v_n} \tag{15.19.2}$$

where $z_i = e^{-\alpha_i + j\omega_i}$, $i = 1, 2, \dots, r$. The problem is to estimate the unknown damping factors, frequencies, and complex amplitudes of the sinusoids, $\{\alpha_i, \omega_i, A_i\}$, from available observations of a length-N data block $\mathbf{y} = [y_0, y_1, \dots, y_{N-1}]^T$ of the noisy signal y_n. We may assume that the z_i are distinct.

In the absence of noise, the $(N - M) \times (M + 1)$-dimensional, covariance-type, data matrix Y can be shown to have rank r, provided that the embedding order M is chosen such that $r \le M \le N - r$. The data matrix Y is defined as:

$$Y = \begin{bmatrix} \mathbf{y}^T(M) \\ \vdots \\ \mathbf{y}^T(n) \\ \vdots \\ \mathbf{y}^T(N-1) \end{bmatrix}, \quad \mathbf{y}(n) = \begin{bmatrix} y_n \\ y_{n-1} \\ y_{n-2} \\ \vdots \\ y_{n-M} \end{bmatrix} = \sum_{i=1}^{r} A_i z_i^n \begin{bmatrix} 1 \\ z_i^{-1} \\ z_i^{-2} \\ \vdots \\ z_i^{-M} \end{bmatrix} \tag{15.19.3}$$

and, Y becomes:

$$Y = \sum_{i=1}^{r} A_i \begin{bmatrix} z_i^M \\ \vdots \\ z_i^n \\ \vdots \\ z_i^{N-1} \end{bmatrix} \bigl[1,\, z_i^{-1},\, z_i^{-2},\, \dots,\, z_i^{-M}\bigr] \tag{15.19.4}$$

Thus, Y is the sum of r rank-1 matrices, and therefore, it will have rank r. Its null space $N(Y)$ can be characterized conveniently in terms of the order-r polynomial with the z_i as roots, that is,

$$A(z) = \prod_{i=1}^{r} (1 - z_i z^{-1}) \tag{15.19.5}$$

Multiplying $A(z)$ by an arbitrary polynomial $F(z)$ of order $M - r$, gives an order-M polynomial $B(z) = A(z)F(z)$, such that r of its roots are the z_i, that is, $B(z_i) = 0$, for $i = 1, 2\dots, r$, and the remaining $M - r$ roots are arbitrary. The polynomial $B(z)$ defines an $(M + 1)$-dimensional vector $\mathbf{b} = [b_0, b_1, \dots, b_M]^T$ through its inverse z-transform:

$$B(z) = b_0 + b_1 z^{-1} + \cdots + b_M z^{-M} = \bigl[1, z^{-1}, z^{-2}, \dots, z^{-M}\bigr] \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_M \end{bmatrix} \tag{15.19.6}$$

Then, the root conditions can be stated in the form:

$$B(z_i) = \bigl[1, z_i^{-1}, z_i^{-2}, \dots, z_i^{-M}\bigr] \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_M \end{bmatrix} = 0\,, \quad i = 1, 2, \dots, r \tag{15.19.7}$$

This implies that the vector $\mathbf{b}$ must lie in the null space of Y:

$$Y\mathbf{b} = \sum_{i=1}^{r} A_i \begin{bmatrix} z_i^M \\ \vdots \\ z_i^n \\ \vdots \\ z_i^{N-1} \end{bmatrix} \bigl[1,\, z_i^{-1},\, z_i^{-2},\, \dots,\, z_i^{-M}\bigr] \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_M \end{bmatrix} = 0 \tag{15.19.8}$$

Conversely, if $\mathbf{b}$ satisfies Eq. (15.19.8), then because $M \le N - r$, or, $r \le N - M$, the $(N{-}M)$-dimensional column vectors $[z_i^M, \dots, z_i^n, \dots, z_i^{N-1}]^T$ are linearly independent,[†] and therefore, we must have:

$$Y\mathbf{b} = \sum_{i=1}^{r} A_i \begin{bmatrix} z_i^M \\ \vdots \\ z_i^n \\ \vdots \\ z_i^{N-1} \end{bmatrix} B(z_i) = 0 \quad \Rightarrow \quad B(z_i) = 0 \tag{15.19.9}$$

[†] This follows from the fact that the $r{\times}r$ Vandermonde matrix $V_{ki} = z_i^{k-1}$, $k, i = 1, 2, \dots, r$, has nonvanishing determinant $\det(V) = \prod_{1\le i<j\le r}(z_i - z_j)$, because the z_i are distinct. See Ref. [1234].

Thus, $B(z)$ must have the form $B(z) = A(z)F(z)$. Because $F(z)$ has degree $M - r$, it will be characterized by $M + 1 - r$ arbitrary coefficients. It follows that the dimensionality of $\mathbf{b}$, and hence of the null space $N(Y)$, will be $M + 1 - r$. This implies that the rank of Y is r.

Next, we consider the matrix pencil of the two submatrices Y_1, Y_0 of Y, where Y_1 is defined to be the *first* M columns of Y, and Y_0, the *last* M, that is,

$$\begin{aligned} Y_1 &= \sum_{i=1}^{r} A_i \begin{bmatrix} z_i^M \\ \vdots \\ z_i^n \\ \vdots \\ z_i^{N-1} \end{bmatrix} \bigl[1,\, z_i^{-1},\, z_i^{-2},\, \dots,\, z_i^{-(M-1)}\bigr] \\ Y_0 &= \sum_{i=1}^{r} A_i \begin{bmatrix} z_i^M \\ \vdots \\ z_i^n \\ \vdots \\ z_i^{N-1} \end{bmatrix} \bigl[z_i^{-1},\, z_i^{-2},\, \dots,\, z_i^{-M}\bigr] \end{aligned} \tag{15.19.10}$$

They were obtained by keeping the first or last M entries of $[1,\, z_i^{-1},\, z_i^{-2},\, \dots,\, z_i^{-M}]$:

$$\bigl[\underbrace{1,\, z_i^{-1},\, z_i^{-2},\, \dots,\, z_i^{-(M-1)}}_{\text{first } M},\, z_i^{-M}\bigr] = \bigl[1,\, \underbrace{z_i^{-1},\, z_i^{-2},\, \dots,\, z_i^{-(M-1)},\, z_i^{-M}}_{\text{last } M}\bigr]$$

Both matrices Y_1, Y_0 have dimension $(N - M)\times M$. Noting that

$$\bigl[1,\, z_i^{-1},\, z_i^{-2},\, \dots,\, z_i^{-(M-1)}\bigr] = z_i\bigl[z_i^{-1},\, z_i^{-2},\, \dots,\, z_i^{-M}\bigr],$$

we may rewrite Y_1 in the form:

$$Y_1 = \sum_{i=1}^{r} z_i A_i \begin{bmatrix} z_i^M \\ \vdots \\ z_i^n \\ \vdots \\ z_i^{N-1} \end{bmatrix} \bigl[z_i^{-1},\, z_i^{-2},\, \dots,\, z_i^{-M}\bigr] \tag{15.19.11}$$

Therefore, the matrix pencil $Y_1 - \lambda Y_0$ can be written as:

$$Y_1 - \lambda Y_0 = \sum_{i=1}^{r} (z_i - \lambda) A_i \begin{bmatrix} z_i^M \\ \vdots \\ z_i^n \\ \vdots \\ z_i^{N-1} \end{bmatrix} [z_i^{-1},\, z_i^{-2},\, \dots,\, z_i^{-M}] \tag{15.19.12}$$

Because $r \le M$ and $r \le N - M$, and $Y_1 - \lambda Y_0$ is a sum of r rank-1 matrices, it follows that, as long as $\lambda \ne z_i$, the rank of $Y_1 - \lambda Y_0$ will be r. However, whenever λ becomes equal to one of the z_i, one of the rank-1 terms will vanish and the rank of $Y_1 - \lambda Y_0$ will collapse to $r - 1$. Thus, the r desired zeros z_i are the generalized eigenvalues of the rank-r matrix pencil $Y_1 - \lambda Y_0$.

When noise is added to the sinusoids, the matrix pencil $Y_1 - \lambda Y_0$ will have full rank, but we expect its r most dominant generalized eigenvalues to be good estimates of the z_i.

In fact, the problem of finding the r eigenvalues z_i can be reduced to an ordinary $r\times r$ eigenvalue problem. First, the data matrices Y_1, Y_0 are extracted from the matrix Y, for example, if $N = 10$ and $M = 3$, we have:

$$Y = \begin{bmatrix} y_3 & y_2 & y_1 & y_0 \\ y_4 & y_3 & y_2 & y_1 \\ y_5 & y_4 & y_3 & y_2 \\ y_6 & y_5 & y_4 & y_3 \\ y_7 & y_6 & y_5 & y_4 \\ y_8 & y_7 & y_6 & y_5 \\ y_9 & y_8 & y_7 & y_6 \end{bmatrix} \quad \Rightarrow \quad Y_1 = \begin{bmatrix} y_3 & y_2 & y_1 \\ y_4 & y_3 & y_2 \\ y_5 & y_4 & y_3 \\ y_6 & y_5 & y_4 \\ y_7 & y_6 & y_5 \\ y_8 & y_7 & y_6 \\ y_9 & y_8 & y_7 \end{bmatrix}, \quad Y_0 = \begin{bmatrix} y_2 & y_1 & y_0 \\ y_3 & y_2 & y_1 \\ y_4 & y_3 & y_2 \\ y_5 & y_4 & y_3 \\ y_6 & y_5 & y_4 \\ y_7 & y_6 & y_5 \\ y_8 & y_7 & y_6 \end{bmatrix}$$

Second, a rank-r reduction is performed on Y_0, approximating it as $Y_0 = U_r \Sigma_r V_r^\dagger$, where U_r has size $(N - M)\times r$, Σ_r is $r\times r$, and V_r, $M\times r$. The matrix pencil becomes then $Y_1 - \lambda U_r \Sigma_r V_r^\dagger$. Multiplying from the left by $U_r^\dagger$ and from the right by V_r and using the orthogonality properties $U_r^\dagger U_r = I_r$ and $V_r^\dagger V_r = I_r$, we obtain the equivalent $r\times r$ matrix pencil:

$$U_r^\dagger (Y_1 - \lambda Y_0) V_r = U_r^\dagger Y_1 V_r - \lambda \Sigma_r \quad \text{or,}$$

$$\Sigma_r^{-1} U_r^\dagger (Y_1 - \lambda Y_0) V_r = Z - \lambda I_r\,, \quad \text{where } Z = \Sigma_r^{-1} U_r^\dagger Y_1 V_r \tag{15.19.13}$$

Finally, the eigenvalues of the $r\times r$ matrix Z are computed, which are the desired estimates of the z_i. The matrix pencil $Z - \lambda I_r$ may also be obtained by inverting Y_0 using its pseudoinverse and then reducing the problem to size $r\times r$. Using $Y_0^+ = V_r \Sigma_r^{-1} U_r^\dagger$, it can be shown easily that:

$$V_r^\dagger (Y_0^+ Y_1 - \lambda I_M) V_r = Z - \lambda I_r$$

Once the z_i are determined, the amplitudes A_i may be calculated by least-squares. Writing Eq. (15.19.2) vectorially for the given length-N signal y_n, we have:

$$\begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{N-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ z_1 & z_2 & \cdots & z_r \\ \vdots & \vdots & \vdots & \vdots \\ z_1^{N-1} & z_2^{N-1} & \cdots & z_r^{N-1} \end{bmatrix} \begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_r \end{bmatrix} + \begin{bmatrix} v_0 \\ v_1 \\ \vdots \\ v_{N-1} \end{bmatrix} \tag{15.19.14}$$

or, written compactly as

$$\mathbf{y} = S\mathbf{A} + \mathbf{v} \tag{15.19.15}$$

with least-squares solution:

$$\mathbf{A} = S^+ \mathbf{y} = S \backslash \mathbf{y} \tag{15.19.16}$$

The above design steps have been implemented into the MATLAB **mpencil**:

```
[z,A] = mpencil(y,r,M);    % matrix pencil method
```

The $N\times r$ Vandermonde matrix S with matrix elements $S_{ni} = z_i^n$, for $0 \le n \le N - 1$ and $1 \le i \le r$, is known as a *steering matrix* and its individual columns as steering vectors. It can be computed by the MATLAB function **steering**:

```
S = steering(N-1,z);       % steering matrix
```

15.20 QR Factorization

The Gram-Schmidt orthogonalization of random variables has many uses: (a) it leads to signal models through the innovations representation, (b) it is equivalent to linear prediction, (c) it corresponds to the Cholesky factorization of the covariance matrix, and (d) it provides efficient computational bases for linear estimation problems, leading to fast solutions of normal equations via Levinson's or Schur's algorithms and to fast adaptive implementations, such as adaptive Gram-Schmidt preprocessors in antenna arrays and fast adaptive lattice filters in time-series applications.

The Gram-Schmidt orthogonalization of an $(M+1)$-dimensional complex-valued zero-mean random vector $\mathbf{y} = [y_0, y_1, \dots, y_M]^T$ is defined by:

$$\begin{aligned} &\epsilon_0 = y_0 \\ &\text{for } m = 1, 2, \dots, M \text{ do:} \\ &\quad \epsilon_m = y_m - \sum_{i=0}^{m-1} \frac{E[\epsilon_i^* y_m]}{E[\epsilon_i^* \epsilon_i]}\, \epsilon_i \end{aligned} \tag{15.20.1}$$

The constructed random vector $\boldsymbol{\epsilon} = [\epsilon_0, \epsilon_1 \dots, \epsilon_M]^T$ has uncorrelated components $E[\epsilon_i^* \epsilon_j] = 0$, if $i \ne j$. The unit lower-triangular innovations matrix B may be defined in terms of its lower-triangular matrix elements:

$$b_{mi} = \frac{E[y_m^* \epsilon_i]}{E[\epsilon_i^* \epsilon_i]}\,, \quad 1 \le m \le M\,, \quad 0 \le i \le m - 1 \tag{15.20.2}$$

Then, Eq. (15.20.1) can be written as $y_m = \epsilon_m + \sum_{i=0}^{m-1} b_{mi}^*\epsilon_i$, or expressed vectorially:

$$\boxed{\mathbf{y} = B^*\boldsymbol{\epsilon}} \tag{15.20.3}$$

for example,

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ b_{10}^* & 1 & 0 & 0 \\ b_{20}^* & b_{21}^* & 1 & 0 \\ b_{30}^* & b_{31}^* & b_{32}^* & 1 \end{bmatrix} \begin{bmatrix} \epsilon_0 \\ \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \end{bmatrix}$$

The matrix B is the unit lower-triangular Cholesky factor of the covariance matrix of the random vector $\mathbf{y}$:

$$\boxed{R = BDB^\dagger} \tag{15.20.4}$$

where $R = E[\mathbf{y}^*\mathbf{y}^T]$ and $D = E[\boldsymbol{\epsilon}^*\boldsymbol{\epsilon}^T] = \text{diag}\{E_0, E_1, \dots, E_M\}$, where E_i is the variance of ϵ_i, that is, $E_i = E[\epsilon_i^*\epsilon_i]$.

We may work also with the random variables $q_i = \epsilon_i/E_i^{1/2}$, $i = 0, 1, \dots, M$, normalized to have *unit variance*. Then, the random vector $\mathbf{q} = [q_0, q_1, \dots, q_M]^T$ will have unit covariance matrix:

$$\mathbf{q} = D^{-1/2}\boldsymbol{\epsilon} \quad \Rightarrow \quad E[\mathbf{q}^*\mathbf{q}^T] = I \tag{15.20.5}$$

where I is the $(M+1)$-dimensional identity matrix. Defining the *upper triangular* matrix $G = D^{1/2}B^\dagger$, we note that $G^\dagger G = BDB^\dagger$ and $G^T = B^* D^{1/2}$ and therefore, Eqs. (15.20.3) and (15.20.4) can be rewritten as:

$$\boxed{\mathbf{y} = G^T\mathbf{q}\,, \quad R = G^\dagger G} \tag{15.20.6}$$

In practice, we must work with sample covariance matrices estimated on the basis of N vectors $\mathbf{y}(n)$, $n = 0, 1, \dots, N-1$. The $N\times(M+1)$ data matrix Y constructed from these vectors is used to obtain the sample covariance matrix:

$$Y = \begin{bmatrix} \mathbf{y}^T(0) \\ \vdots \\ \mathbf{y}^T(n) \\ \vdots \\ \mathbf{y}^T(N-1) \end{bmatrix} \quad \Rightarrow \quad \hat{R} = Y^\dagger Y \tag{15.20.7}$$

The QR-factorization factors the data matrix Y into an $N\times(M+1)$ matrix Q with orthonormal columns and an $(M+1)\times(M+1)$ upper triangular matrix G:

$$Y = QG\,, \quad Q^\dagger Q = I\,, \quad G = \text{upper triangular} \tag{15.20.8}$$

The matrix Q is obtained by the Gram-Schmidt orthogonalization of the $(M+1)$ columns of Y. The QR-factorization implies the Cholesky factorization of the covariance matrix $\hat{R}$. Using $Q^\dagger Q = I$, we have:

$$\hat{R} = Y^\dagger Y = G^\dagger Q^\dagger Q G = G^\dagger G \tag{15.20.9}$$

Writing Q row-wise, we can extract the snapshot vector $\mathbf{q}(n)$ corresponding to $\mathbf{y}(n)$, that is,

$$Y = QG \quad \Rightarrow \quad \begin{bmatrix} \mathbf{y}^T(0) \\ \vdots \\ \mathbf{y}^T(n) \\ \vdots \\ \mathbf{y}^T(N-1) \end{bmatrix} = \begin{bmatrix} \mathbf{q}^T(0) \\ \vdots \\ \mathbf{q}^T(n) \\ \vdots \\ \mathbf{q}^T(N-1) \end{bmatrix} G \quad \Rightarrow \quad \mathbf{y}^T(n) = \mathbf{q}^T(n)G\,, \quad \text{or,}$$

$$\mathbf{y}(n) = G^T\mathbf{q}(n)\,, \quad n = 0, 1, \dots N-1 \tag{15.20.10}$$

which is the same as Eq. (15.20.6).

Writing $\mathbf{q}^T(n) = [q_0(n), q_1(n), \dots, q_M(n)]$, the ith column of Q is the time signal $q_i(n)$, $n = 0, 1, \dots, N-1$. Orthonormality in the statistical sense translates to orthonormality in the time-average sense:

$$E[\mathbf{q}^*\mathbf{q}^T] = I \quad \Rightarrow \quad Q^\dagger Q = \sum_{n=0}^{N-1} \mathbf{q}^*(n)\mathbf{q}^T(n) = I \tag{15.20.11}$$

or, component-wise, for $i, j, = 0, 1, \dots, M$:

$$E[q_i^* q_j] = \delta_{ij} \quad \Rightarrow \quad \sum_{n=0}^{N-1} q_i^*(n)q_j(n) = \delta_{ij} \tag{15.20.12}$$

In comparing the SVD versus the QR-factorization, we observe that both methods orthogonalize the random vector $\mathbf{y}$. The SVD corresponds to the KLT/PCA eigenvalue decomposition of the covariance matrix, whereas the QR corresponds to the Cholesky factorization. The following table compares the two approaches.

KLT/PCA	Cholesky Factorization
$R = E[\mathbf{y}^*\mathbf{y}^T] = V\Lambda V^\dagger$ $\mathbf{y} = V^*\mathbf{z} = V^*\Sigma\,\mathbf{u}$ $E[\mathbf{z}^*\mathbf{z}^T] = \Lambda = \Sigma^2$ $E[\mathbf{u}^*\mathbf{u}^T] = I$	$R = E[\mathbf{y}^*\mathbf{y}^T] = G^\dagger G = BDB^\dagger$ $\mathbf{y} = G^T\mathbf{q} = B^*\boldsymbol{\epsilon}$ $E[\boldsymbol{\epsilon}^*\boldsymbol{\epsilon}^T] = D$ $E[\mathbf{q}^*\mathbf{q}^T] = I$
SVD	QR
$Y = U\Sigma V^\dagger = ZV^\dagger$ $\hat{R} = Y^\dagger Y = V\Lambda V^\dagger = V\Sigma^2 V^\dagger$ $\mathbf{y}(n) = V^*\mathbf{z}(n) = V^*\Sigma\,\mathbf{u}(n)$ $\sum_{n=0}^{N-1} \mathbf{u}^*(n)\mathbf{u}^T(n) = I$	$Y = QG$ $\hat{R} = Y^\dagger Y = G^\dagger G$ $\mathbf{y}(n) = G^T\mathbf{q}(n) = B^*\boldsymbol{\epsilon}(n)$ $\sum_{n=0}^{N-1} \mathbf{q}^*(n)\mathbf{q}^T(n) = I$

15.21 Canonical Correlation Analysis

Canonical correlation analysis (CCA) attempts to determine if there are any significant correlations between two *groups* of random variables. It does so by finding linear combinations of the first group and linear combinations of the second group that are maximally correlated with each other [1237–1239,1315–1321].

Consider the two groups of random variables to be the components of two zero-mean random vectors $\mathbf{y}_a$ and $\mathbf{y}_b$ of dimensions p and q. Concatenating the vectors $\mathbf{y}_a, \mathbf{y}_b$ into a $(p+q)$-dimensional vector, we have:

$$\mathbf{y} = \left[\begin{array}{c} \mathbf{y}_a \\ \hline \mathbf{y}_b \end{array}\right], \quad \text{where} \quad \mathbf{y}_a = \begin{bmatrix} y_{a1} \\ y_{a2} \\ \vdots \\ y_{ap} \end{bmatrix}, \quad \mathbf{y}_b = \begin{bmatrix} y_{b1} \\ y_{b2} \\ \vdots \\ y_{bq} \end{bmatrix} \tag{15.21.1}$$

Its covariance matrix can be expressed in the partitioned form:

$$R = E[\mathbf{y}^*\mathbf{y}^T] = \left[\begin{array}{c|c} E[\mathbf{y}_a^*\mathbf{y}_a^T] & E[\mathbf{y}_a^*\mathbf{y}_b^T] \\ \hline E[\mathbf{y}_b^*\mathbf{y}_a^T] & E[\mathbf{y}_b^*\mathbf{y}_b^T] \end{array}\right] = \left[\begin{array}{c|c} R_{aa} & R_{ab} \\ \hline R_{ba} & R_{bb} \end{array}\right] \tag{15.21.2}$$

In general, the matrices R_{aa}, R_{ab}, R_{bb} are full and inspection of their entries does not provide—especially when the matrices are large—a clear insight as to the essential correlations between the two groups.

What should be the ideal form of R in order to bring out such essential correlations? As an example, consider the case $p = 3$ and $q = 2$ and suppose R has the following structure, referred to as the *canonical correlation structure*:

$$R = \left[\begin{array}{ccc|cc} R_{a1,a1} & R_{a1,a2} & R_{a1,a3} & R_{a1,b1} & R_{a1,b2} \\ R_{a2,a1} & R_{a2,a2} & R_{a2,a3} & R_{a2,b1} & R_{a2,b2} \\ R_{a3,a1} & R_{a3,a2} & R_{a3,a3} & R_{a3,b1} & R_{a3,b2} \\ \hline R_{b1,a1} & R_{b1,a2} & R_{b1,a3} & R_{b1,b1} & R_{b1,b2} \\ R_{b2,a1} & R_{b2,a2} & R_{b2,a3} & R_{b2,b1} & R_{b2,b2} \end{array}\right] = \left[\begin{array}{ccc|cc} 1 & 0 & 0 & c_1 & 0 \\ 0 & 1 & 0 & 0 & c_2 \\ 0 & 0 & 1 & 0 & 0 \\ \hline c_1 & 0 & 0 & 1 & 0 \\ 0 & c_2 & 0 & 0 & 1 \end{array}\right]$$

where the random vectors are $\mathbf{y}_a = [y_{a1}, y_{a2}, y_{a3}]^T$ and $\mathbf{y}_b = [y_{b1}, y_{b2}]^T$, and we denoted $R_{ai,aj} = E[y_{ai}^* y_{aj}]$, $R_{ai,bj} = E[y_{ai}^* y_{bj}]$, $R_{bi,bj} = E[y_{bi}^* y_{bj}]$.

This form tells us that the random variables $\{y_{a1}, y_{a2}, y_{a3}\}$ are mutually uncorrelated and have unit variance, and so are the $\{y_{b1}, y_{b2}\}$. Moreover, between group a and group b, the random variable y_{a1} is correlated only with y_{b1}, with correlation $c_1 = E[y_{a1}^* y_{b1}]$, and y_{a2} is correlated only with y_{b2}, with correlation $c_2 = E[y_{a2}^* y_{b2}]$. Assuming $c_1 \ge c_2$, the pair $\{y_{a1}, y_{b1}\}$ will be more correlated than the pair $\{y_{a2}, y_{b2}\}$. The case $p = 2$ and $q = 3$ would be:

$$R = \left[\begin{array}{cc|ccc} R_{a1,a1} & R_{a1,a2} & R_{a1,b1} & R_{a1,b2} & R_{a1,b3} \\ R_{a2,a1} & R_{a2,a2} & R_{a2,b1} & R_{a2,b2} & R_{a2,b3} \\ \hline R_{b1,a1} & R_{b1,a2} & R_{b1,b1} & R_{b1,b2} & R_{b1,b3} \\ R_{b2,a1} & R_{b2,a2} & R_{b2,b1} & R_{b2,b2} & R_{b2,b3} \\ R_{b3,a1} & R_{b3,a2} & R_{b3,b1} & R_{b3,b2} & R_{b3,b3} \end{array}\right] = \left[\begin{array}{cc|ccc} 1 & 0 & c_1 & 0 & 0 \\ 0 & 1 & 0 & c_2 & 0 \\ \hline c_1 & 0 & 1 & 0 & 0 \\ 0 & c_2 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

Thus, the canonical structure, having a diagonal submatrix R_{ab}, describes the correlations between a and b in their clearest form. The goal of CCA is to bring the general covariance matrix R of Eq. (15.21.2) into such a canonical form. This is accomplished by finding appropriate linear transformations that change the bases $\mathbf{y}_a$ and $\mathbf{y}_b$ into the above form.

One may start by finding p- and q-dimensional vectors $\mathbf{a}, \mathbf{b}$ such that the linear combinations $w_a = \mathbf{a}^T\mathbf{y}_a$ and $w_b = \mathbf{b}^T\mathbf{y}_b$ are maximally correlated, that is, finding $\mathbf{a}, \mathbf{b}$ that maximize the normalized correlation coefficient:

$$c = \frac{E[w_a^* w_b]}{\sqrt{E[w_a^* w_a]E[w_b^* w_b]}} = \max \tag{15.21.3}$$

Noting that $E[w_a^* w_b] = \mathbf{a}^\dagger R_{ab}\mathbf{b}$, $E[w_a^* w_a] = \mathbf{a}^\dagger R_{aa}\mathbf{a}$, and $E[w_b^* w_b] = \mathbf{b}^\dagger R_{bb}\mathbf{b}$, the above criterion becomes:

$$c = \frac{\mathbf{a}^\dagger R_{ab}\mathbf{b}}{\sqrt{(\mathbf{a}^\dagger R_{aa}\mathbf{a})\,(\mathbf{b}^\dagger R_{bb}\mathbf{b})}} = \max \tag{15.21.4}$$

We may impose the constraints that w_a, w_b have unit variance, that is, $E[w_a^* w_a] = \mathbf{a}^\dagger R_{aa}\mathbf{a} = 1$ and $E[w_b^* w_b] = \mathbf{b}^\dagger R_{bb}\mathbf{b} = 1$. Then, the equivalent criterion reads:

$$c = \mathbf{a}^\dagger R_{ab}\mathbf{b} = \max\,, \quad \text{subject to} \quad \mathbf{a}^\dagger R_{aa}\mathbf{a} = 1\,, \quad \mathbf{b}^\dagger R_{bb}\mathbf{b} = 1 \tag{15.21.5}$$

This is reminiscent of the maximization criterion for singular values that we discussed in Sec. 15.5. To recast (15.21.5) into that form, we first change into a basis in which the group random vectors have unit covariance matrix. Performing the full SVDs of R_{aa} and R_{bb}, we set:

$$\begin{aligned} R_{aa} &= U_a\Lambda_a V_a^\dagger = V_a\Sigma_a^2 V_a^\dagger\,, \quad U_a = V_a\,, \quad \Sigma_a = \Lambda_a^{1/2}\,, \quad V_a^\dagger V_a = I_p \\ R_{bb} &= U_b\Lambda_b V_b^\dagger = V_b\Sigma_b^2 V_b^\dagger\,, \quad U_b = V_b\,, \quad \Sigma_b = \Lambda_b^{1/2}\,, \quad V_b^\dagger V_b = I_q \end{aligned} \tag{15.21.6}$$

These are essentially the eigenvalue decompositions of the hermitian positive definite matrices R_{aa}, R_{bb}. We assume that Σ_a, Σ_b are non-singular, that is, R_{aa}, R_{bb} have full rank. Then, we define the transformed random vectors and corresponding cross-correlation matrix:

$$\begin{aligned} \mathbf{u}_a &= \Sigma_a^{-1}V_a^T\mathbf{y}_a \\ \mathbf{u}_b &= \Sigma_b^{-1}V_b^T\mathbf{y}_b \end{aligned} \quad \Rightarrow \quad C_{ab} = E[\mathbf{u}_a^*\mathbf{u}_b^T] = \Sigma_a^{-1}V_a^\dagger R_{ab}V_b\Sigma_b^{-1} \tag{15.21.7}$$

In this basis, $E[\mathbf{u}_a^*\mathbf{u}_a^T] = \Sigma_a^{-1}V_a^\dagger R_{aa}V_a\Sigma_a^{-1} = \Sigma_a^{-1}V_a^\dagger V_a\Sigma_a^2 V_a^\dagger V_a\Sigma_a^{-1} = I_p$, and similarly, $E[\mathbf{u}_b^*\mathbf{u}_b^T] = I_q$. The transformed covariance matrix will be:

$$\mathbf{u} = \begin{bmatrix} \mathbf{u}_a \\ \mathbf{u}_b \end{bmatrix} \quad \Rightarrow \quad R_{uu} = E[\mathbf{u}^*\mathbf{u}^T] = \left[\begin{array}{c|c} E[\mathbf{u}_a^*\mathbf{u}_a^T] & E[\mathbf{u}_a^*\mathbf{u}_b^T] \\ \hline E[\mathbf{u}_b^*\mathbf{u}_a^T] & E[\mathbf{u}_b^*\mathbf{u}_b^T] \end{array}\right] = \left[\begin{array}{c|c} I_p & C_{ab} \\ \hline C_{ab}^\dagger & I_q \end{array}\right]$$

An alternative method of obtaining unit-covariance bases is to use the Cholesky factorization. For example, we may set $R_{aa} = G_a^\dagger G_a$, where G_a is upper triangular, and define $\mathbf{u}_a = G_a^{-T}\mathbf{y}_a$.

Having transformed to a new basis, we also transform the coefficients $\mathbf{a},\mathbf{b}$ so that w_a, w_b are expressed as linear combinations in the new basis:

$$\begin{aligned} \mathbf{f}_a &= \Sigma_a V_a^\dagger \mathbf{a} \quad \Rightarrow \quad \mathbf{a} = V_a \Sigma_a^{-1} \mathbf{f}_a \quad \Rightarrow \quad w_a = \mathbf{a}^T \mathbf{y}_a = \mathbf{f}_a^T \mathbf{u}_a \\ \mathbf{f}_b &= \Sigma_b V_b^\dagger \mathbf{b} \quad \Rightarrow \quad \mathbf{b} = V_b \Sigma_b^{-1} \mathbf{f}_b \quad \Rightarrow \quad w_b = \mathbf{b}^T \mathbf{y}_b = \mathbf{f}_b^T \mathbf{u}_b \end{aligned} \tag{15.21.8}$$

Similarly, we have:

$$\begin{aligned} E[w_a^* w_a] &= \mathbf{a}^\dagger R_{aa} \mathbf{a} = \mathbf{f}_a^\dagger \Sigma_a^{-1} V_a^\dagger R_{aa} V_a \Sigma_a^{-1} \mathbf{f}_a = \mathbf{f}_a^\dagger \mathbf{f}_a \\ E[w_b^* w_b] &= \mathbf{b}^\dagger R_{bb} \mathbf{b} = \mathbf{f}_b^\dagger \Sigma_b^{-1} V_b^\dagger R_{bb} V_b \Sigma_b^{-1} \mathbf{f}_b = \mathbf{f}_b^\dagger \mathbf{f}_b \\ E[w_a^* w_b] &= \mathbf{a}^\dagger R_{ab} \mathbf{b} = \mathbf{f}_a^\dagger \Sigma_a^{-1} V_a^\dagger R_{ab} V_b \Sigma_b^{-1} \mathbf{f}_b = \mathbf{f}_a^\dagger C_{ab} \mathbf{f}_b \end{aligned} \tag{15.21.9}$$

Then, the criterion (15.21.5) may be expressed as an SVD maximization criterion in the new basis:

$$c = \mathbf{f}_a^\dagger C_{ab} \mathbf{f}_b = \max\,, \quad \text{subject to} \quad \mathbf{f}_a^\dagger \mathbf{f}_a = \mathbf{f}_b^\dagger \mathbf{f}_b = 1 \tag{15.21.10}$$

It follows from Eq. (15.5.17) that the solution is $c = c_1$, the maximum singular value of C_{ab}, and the vectors $\mathbf{f}_a, \mathbf{f}_b$ are the first singular vectors. The remaining singular values of C_{ab} are the lower maxima of (15.21.10) and are obtained subject to the orthogonality constraints of Eq. (15.5.18).

Thus, the desired canonical correlation structure is derived from the SVD of the matrix C_{ab}. The singular values of C_{ab} are called the *canonical correlations*. The following procedure will construct all of them. Start with the full SVD of C_{ab}:

$$C_{ab} = F_a C F_b^\dagger\,, \quad C = \text{diag}\{c_1, c_2 \dots, c_r\} \in \mathbb{C}^{p\times q} \tag{15.21.11}$$

where $c_1 \geq c_2 \geq \cdots \geq c_r > 0$ and $r = \min(p,q)$ (full rank case), and F_a, F_b are unitary matrices, that is, $F_a^\dagger F_a = F_a F_a^\dagger = I_p$ and $F_b^\dagger F_b = F_b F_b^\dagger = I_q$. Then, construct the CCA coefficient matrices:

$$\begin{aligned} A &= V_a \Sigma_a^{-1} F_a = [\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_p] = p\times p \text{ matrix} \\ B &= V_b \Sigma_b^{-1} F_b = [\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_q] = q\times q \text{ matrix} \end{aligned} \tag{15.21.12}$$

The coefficient matrices A, B transform the basis $\mathbf{y}_a, \mathbf{y}_b$ directly into the canonical correlation basis. We define:

$$\begin{aligned} \mathbf{w}_a &= A^T \mathbf{y}_a \\ \mathbf{w}_b &= B^T \mathbf{y}_b \end{aligned} \quad \Rightarrow \quad \mathbf{w} = \begin{bmatrix} \mathbf{w}_a \\ \mathbf{w}_b \end{bmatrix} = \begin{bmatrix} A^T & 0 \\ 0 & B^T \end{bmatrix} \begin{bmatrix} \mathbf{y}_a \\ \mathbf{y}_b \end{bmatrix} \tag{15.21.13}$$

Then, the corresponding covariance matrix will be:

$$R_{ww} = E[\mathbf{w}^* \mathbf{w}^T] = \left[\begin{array}{c|c} E[\mathbf{w}_a^* \mathbf{w}_a^T] & E[\mathbf{w}_a^* \mathbf{w}_b^T] \\ \hline E[\mathbf{w}_b^* \mathbf{w}_a^T] & E[\mathbf{w}_b^* \mathbf{w}_b^T] \end{array}\right] = \left[\begin{array}{c|c} A^\dagger R_{aa} A & A^\dagger R_{ab} B \\ \hline B^\dagger R_{ba} A & B^\dagger R_{bb} B \end{array}\right] \tag{15.21.14}$$

By construction, we have $A^\dagger R_{aa} A = I_p$, $A^\dagger R_{ab} B = C$, and $B^\dagger R_{bb} B = I_q$. Thus, we obtain the canonical correlation structure:

$$R_{ww} = E[\mathbf{w}^* \mathbf{w}^T] = \left[\begin{array}{c|c} E[\mathbf{w}_a^* \mathbf{w}_a^T] & E[\mathbf{w}_a^* \mathbf{w}_b^T] \\ \hline E[\mathbf{w}_b^* \mathbf{w}_a^T] & E[\mathbf{w}_b^* \mathbf{w}_b^T] \end{array}\right] = \left[\begin{array}{c|c} I_p & C \\ \hline C^\dagger & I_q \end{array}\right] \tag{15.21.15}$$

The canonical correlations and canonical random variables are obtained from the columns of A, B by the linear combinations:

$$c_i = E[w_{ai}^* w_{bi}]\,, \quad w_{ai} = \mathbf{a}_i^T \mathbf{y}_a\,, \quad w_{bi} = \mathbf{b}_i^T \mathbf{y}_b\,, \quad i = 1, 2, \dots, r \tag{15.21.16}$$

The MATLAB function **ccacov.m** takes as input a $(p+q)\times(p+q)$ covariance matrix R and computes the coefficient matrices A, B and canonical correlations C, using Eqs. (15.21.6), (15.21.7), and (15.21.12). It has usage:

```
[A,C,B] = ccacov(R,p);          % CCA of a covariance matrix
```

Next, we consider the practical implementation of CCA based on N observations $\mathbf{y}_a(n), \mathbf{y}_b(n)$, $n = 0, 1, \dots, N-1$. We form the $N\times p$ and $N\times q$ data matrices, as well as the concatenated $N\times(p+q)$ data matrix:

$$Y_a = \begin{bmatrix} \mathbf{y}_a^T(0) \\ \vdots \\ \mathbf{y}_a^T(n) \\ \vdots \\ \mathbf{y}_a^T(N-1) \end{bmatrix}, \quad Y_b = \begin{bmatrix} \mathbf{y}_b^T(0) \\ \vdots \\ \mathbf{y}_b^T(n) \\ \vdots \\ \mathbf{y}_b^T(N-1) \end{bmatrix}, \quad Y = [Y_a, Y_b] \tag{15.21.17}$$

We assume that the column means have been removed from Y. The corresponding sample covariance matrices are then:

$$\hat{R} = Y^\dagger Y = \left[\begin{array}{c|c} Y_a^\dagger Y_a & Y_a^\dagger Y_b \\ \hline Y_b^\dagger Y_a & Y_b^\dagger Y_b \end{array}\right] = \left[\begin{array}{c|c} \hat{R}_{aa} & \hat{R}_{ab} \\ \hline \hat{R}_{ba} & \hat{R}_{bb} \end{array}\right] \tag{15.21.18}$$

We may obtain the CCA transformation matrices A, B by applying the previous construction to $\hat{R}$. However, a more direct approach is as follows. Starting with the economy SVDs of Y_a and Y_b, we have:

$$\boxed{\begin{aligned} &Y_a = U_a \Sigma_a V_a^\dagger = \text{economy SVD of } Y_a\,, \quad U_a \in \mathbb{C}^{N\times p}\,, \quad U_a^\dagger U_a = I_p \\ &Y_b = U_b \Sigma_b V_b^\dagger = \text{economy SVD of } Y_b\,, \quad U_b \in \mathbb{C}^{N\times q}\,, \quad U_b^\dagger U_b = I_q \\ &C_{ab} = U_a^\dagger U_b = \text{cross-covariance in } u\text{-basis}\,, \quad C_{ab} \in \mathbb{C}^{p\times q} \\ &C_{ab} = F_a C F_b^\dagger = \text{full SVD}\,, \quad C = \text{diag}\{c_1, c_2 \dots, c_r\}\,, \quad r = \min(p,q) \\ &A = V_a \Sigma_a^{-1} F_a = \text{CCA coefficients}\,, \quad A \in \mathbb{C}^{p\times p} \\ &B = V_b \Sigma_b^{-1} F_b = \text{ CCA coefficients}\,, \quad B \in \mathbb{C}^{q\times q} \\ &W_a = Y_a A\,, \quad W_a \in \mathbb{C}^{N\times p} \text{ with orthonormal columns,} \quad W_a^\dagger W_a = I_p \\ &W_b = Y_b B\,, \quad W_b \in \mathbb{C}^{N\times q} \text{ with orthonormal columns,} \quad W_b^\dagger W_b = I_q \\ &W_a^\dagger W_b = C = p\times q \text{ diagonal matrix of canonical correlations} \end{aligned}} \tag{15.21.19}$$

The transformed data matrices W_a, W_b and $W = [W_a, W_b]$ have the canonical correlation structure:

$$W^\dagger W = \left[\begin{array}{c|c} W_a^\dagger W_a & W_a^\dagger W_b \\ \hline W_b^\dagger W_a & W_b^\dagger W_b \end{array}\right] = \left[\begin{array}{c|c} I_p & C \\ \hline C^\dagger & I_q \end{array}\right] \tag{15.21.20}$$

Denoting the ith columns of W_a, W_b by $w_{ai}(n), w_{bi}(n)$, $n = 0, 1, \dots, N-1$, we note that they have unit norm as N-dimensional vectors, and the ith canonical correlation is given by the time-average:

$$c_i = \sum_{n=0}^{N-1} w_{ai}^*(n)\, w_{bi}(n)\,, \quad i = 1, 2, \dots, r \tag{15.21.21}$$

The above steps have been implemented by the MATLAB function **cca**. It takes as inputs the data matrices Y_a, Y_b and outputs A, B, C. Its usage is as follows:

```
[A,C,B] = cca(Ya,Yb);       % CCA of two data matrices
```

Example 15.21.1: As an example, consider again the turtle data in the file **turtle.dat**. We take Y_a, Y_b to be the $(N = 24)$ measured lengths and widths of the male (group a) and female (group b) turtles. The data are shown below:

	Y_a		Y_b	
n	y_{a1}	y_{a2}	y_{b1}	y_{b2}
1	93	74	98	81
2	94	78	103	84
3	96	80	103	86
4	101	84	105	86
5	102	85	109	88
6	103	81	123	92
7	104	83	123	95
8	106	83	133	99
9	107	82	133	102
10	112	89	133	102
11	113	88	134	100
12	114	86	136	102

	Y_a		Y_b	
n	y_{a1}	y_{a2}	y_{b1}	y_{b2}
13	116	90	137	98
14	117	90	138	99
15	117	91	141	103
16	119	93	147	108
17	120	89	149	107
18	120	93	153	107
19	121	95	155	115
20	123	93	155	117
21	127	96	158	115
22	128	95	159	118
23	131	95	162	124
24	135	106	177	132

After removing the columns means, the computed sample covariance matrix is:

$$\hat{R} = Y^\dagger Y = \left[\begin{array}{c|c} Y_a^\dagger Y_a & Y_a^\dagger Y_b \\ \hline Y_b^\dagger Y_a & Y_b^\dagger Y_b \end{array}\right] = 10^3 \left[\begin{array}{cc|cc} 3.1490 & 1.8110 & 5.5780 & 3.3785 \\ 1.8110 & 1.1510 & 3.1760 & 1.9495 \\ \hline 5.5780 & 3.1760 & 10.3820 & 6.2270 \\ 3.3785 & 1.9495 & 6.2270 & 3.9440 \end{array}\right]$$

The computed CCA coefficients and canonical correlations are:

$$A = \begin{bmatrix} 0.0191 & 0.0545 \\ -0.0023 & -0.0955 \end{bmatrix}, \quad B = \begin{bmatrix} 0.0083 & 0.0418 \\ 0.0026 & -0.0691 \end{bmatrix}$$

$$C = \begin{bmatrix} c_1 & 0 \\ 0 & c_2 \end{bmatrix} = \begin{bmatrix} 0.9767 & 0 \\ 0 & 0.1707 \end{bmatrix}$$

The correlation structure in the transformed basis $W = [W_a, W_b]$ is:

$$W^\dagger W = \left[\begin{array}{c|c} W_a^\dagger W_a & W_a^\dagger W_b \\ \hline W_b^\dagger W_a & W_b^\dagger W_b \end{array}\right] = \left[\begin{array}{cc|cc} 1 & 0 & 0.9767 & 0 \\ 0 & 1 & 0 & 0.1707 \\ \hline 0.9767 & 0 & 1 & 0 \\ 0 & 0.1707 & 0 & 1 \end{array}\right]$$

The first columns of W_a, W_b are the most correlated. They are obtained as the following linear combinations of the columns of Y_a, Y_b:

$$\begin{aligned} w_{a1}(n) &= 0.0191\, y_{a1}(n) - 0.0023\, y_{a2}(n) \\ w_{b1}(n) &= 0.0083\, y_{b1}(n) + 0.0026\, y_{b2}(n) \end{aligned} \quad \Rightarrow \quad \sum_{n=0}^{N-1} w_{a1}(n)\, w_{b1}(n) = c_1 = 0.9767$$

where the linear combination coefficients are the first columns of A, B. The following MATLAB code implements this example:

```
D = loadfile('turtle.dat');        % read data file

Ya = zmean(D(:,1:2));              % extract columns 1,2 and remove their mean
Yb = zmean(D(:,4:5));              % extract columns 4,5 and remove their mean

[A,C,B] = cca(Ya,Yb);

Y = [Ya,Yb];  Ryy = Y'*Y;          % correlated basis

Wa = Ya*A;  Wb = Yb*B;
W = [Wa,Wb];  Rww = W'*W;          % canonical correlation basis
```

The quantities A, B, C could also be obtained by the function **ccacov** applied to $R = Y^\dagger Y$ with $p = 2$. Once the coefficients A, B are known, the data matrices Y_a, Y_b may be transformed to W_a, W_b. □

Finally, we mention that CCA is equivalent to the problem of finding the canonical angles between two linear subspaces. Consider the two subspaces of $\mathbb{C}^N$ spanned by the columns of Y_a and Y_b. The economy SVDs of Y_a, Y_b provide orthonormal bases U_a, U_b for these subspaces.

The *canonical angles* between the two subspaces are defined [1234,1320,1321] in terms of the singular values of the matrix $U_a^\dagger U_b$, but these are the canonical correlations. The cosines of the canonical angles are the canonical correlations:

$$c_i = \cos\theta_i\,, \quad i = 1, 2 \dots, r = \min(p, q) \tag{15.21.22}$$

The largest angle corresponds to the smallest singular value, that is, $\cos\theta_{\max} = c_{\min} = c_r$. This angle (in radians) is returned by the built-in MATLAB function **subspace**, that is,

```
th_max = subspace(Ya,Yb);
```

15.22 Problems

15.1 *SVD construction.* Consider the following 5×3 matrix, where ϵ is a small positive parameter:

$$Y = \begin{bmatrix} 1+3\epsilon & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1+3\epsilon & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1+3\epsilon \end{bmatrix}$$

a. Construct the economy SVD of Y, in the form, $Y = U\Sigma V^T$.

b. Show that the rank-1 and rank-2 approximations to Y are given by:

$$Y_1 = \begin{bmatrix} 1+\epsilon & 1+\epsilon & 1+\epsilon \\ 1 & 1 & 1 \\ 1+\epsilon & 1+\epsilon & 1+\epsilon \\ 1 & 1 & 1 \\ 1+\epsilon & 1+\epsilon & 1+\epsilon \end{bmatrix}, \quad Y_2 = \begin{bmatrix} 1+\epsilon & 1+\epsilon & 1+\epsilon \\ 1 & 1 & 1 \\ 1+\epsilon & 1+2.5\epsilon & 1-0.5\epsilon \\ 1 & 1 & 1 \\ 1+\epsilon & 1-0.5\epsilon & 1+2.5\epsilon \end{bmatrix}$$

c. Verify the results of parts (a-b) numerically for the value $\epsilon = 0.1$.

Hint: Note the following matrix has eigenvalues and normalized eigenvectors:

$$R = \begin{bmatrix} R_0 & R_1 & R_1 \\ R_1 & R_0 & R_1 \\ R_1 & R_1 & R_0 \end{bmatrix} \quad \Rightarrow \quad \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix} = \begin{bmatrix} R_0 + 2R_1 \\ R_0 - R_1 \\ R_0 - R_1 \end{bmatrix}, \quad V = \begin{bmatrix} 1/\sqrt{3} & 0 & 2/\sqrt{6} \\ 1/\sqrt{3} & 1/\sqrt{2} & -1/\sqrt{6} \\ 1/\sqrt{3} & -1/\sqrt{2} & -1/\sqrt{6} \end{bmatrix}$$

15.2 *Computer Experiment – Southern Oscillation Index.* It has been observed that in the southern Pacific there occurs regularly an upwelling of large masses of lower-level colder water which has important implications for marine life and coastal weather. This effect, which is variable on a monthly and yearly basis, has been termed *El Niño.* It has been held responsible for many strange global weather effects in the past decades.

One measure of the variability of this effect is the so-called *southern oscillation index* (SOI) which is the atmospheric pressure difference at sea level between two standard locations in the Pacific, namely, Tahiti and Darwin, Australia. This data exhibits a strong 40–50 month cycle and a weaker 10–12 month cycle.

The SOI data, spanning the years 1920–1992, are in the included file `soi2.dat`. The monthly data must be concatenated, resulting into a long one-dimensional time series $y(n)$ and the mean must be removed. (The concatenation can be done by the following MATLAB commands: assuming that Y is the data matrix whose rows are the monthly data for each year, then redefine `Y=Y';` and set `y = Y(:);`)

a. It is desired to fit an AR model to this data, plot the AR spectrum, and identify the spectral peaks. Starting with model order $M = 15$, calculate the ordinary Burg estimate of the prediction-error filter , say $\mathbf{a}_b$.

b. Form the order-M autocorrelation and forward/backward data matrices Y, perform an SVD, and plot the principal component variances as percentages of the total variance. You will observe that beyond the 5th principal component, the variances flatten out, indicating that the dimension of the signal subspace can be taken to be of the order of $r = 5$–9.

Start with the choice $r = 8$ and perform $K = 1$ and $K = 3$ rank-r enhancement operations on the data matrix, as expressed symbolically in MATLAB language:

```
Y = datamat(y,M,type)    % type = 0 or 2
Ye = Y;                  % initialize SVD iterations
for i=1:K,
   Ye = sigsub(Ye,r)     % rank-r signal subspace
   Ye = toepl(Ye,type)   % type = 0 or 2
end
ye = datasig(Ye,type)    % extract enhanced signal from Ye
```

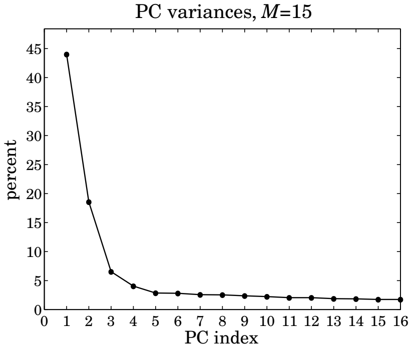
c. Using the *enhanced* data matrix Y_e, calculate the least-squares prediction error filter, $\mathbf{a}_{LS}$, by solving $Y_e\mathbf{a} = 0$.

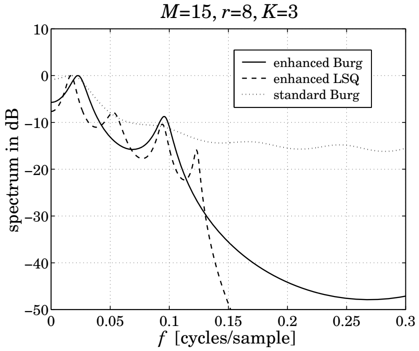
d. From the extracted *enhanced* signal $y_e(n)$, calculate the corresponding order-r Burg estimate of the prediction-error filter, say $\mathbf{a}_e$. (You could also do an order-M Burg estimate from $y_e(n)$, but the order-r choice is more appropriate since r is the assumed dimension of the signal subspace.)

e. Calculate and plot in dB the AR spectra of the three prediction filters, $\mathbf{a}_b$, $\mathbf{a}_{LS}$, $\mathbf{a}_e$. Normalize each spectrum to unity maximum.

Identify the frequency of the highest peak in each spectrum and determine the corresponding period of the cycle in months. Identify also the frequency and period of the secondary peak that would represent the 10–12 month cycle.

f. Repeat the steps (a)–(e) for the following values of the parameters: For $M = 4$, $r = 3$, $K = 1, 3$. And then, for $M = 15$, $r = 5, 6, 7, 9$, and $K = 1, 3$. Moreover, do both the autocorrelation and forward-backward versions of the data matrices. Some example graphs are shown below.



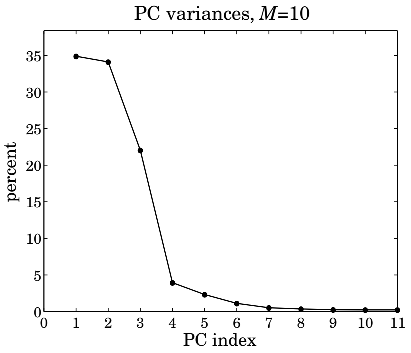


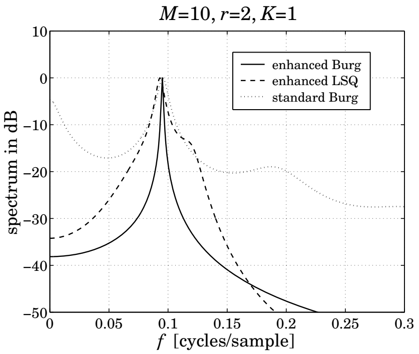
15.3 *Computer Experiment – Sunspot Numbers.* The Wolf sunspot numbers are of great historical importance in the development of spectral analysis methods (periodogram and parametric). Sunspot activity is cyclical and variation in the sunspot numbers has been correlated with weather and other terrestrial phenomena of economic significance. There is a strong 10-11 year cycle.

The observed yearly number of sunspots over the period 1700–2004 can be obtained from the course's web page. The mean of this data must be removed.

a. It is desired to fit an AR model to this data, plot the AR spectrum, and identify the dominant peak corresponding to the 10–11 year cycle.

b. Perform the steps (a)–(e) as described in Problem 15.2 for the following values of the parameters: $M = 10$, $r = 2, 3, 4$, $K = 1, 3$. Try also the simpler case $M = 3$, $r = 2$, $K = 1, 3$. Some example graphs are shown below.





15.4 *Computer Experiment – PCA analysis of Olympic Track Records.* Please read reference [1309] on applying PCA to the 1984 Olympic track records. The attached files, `olymp1.dat`, `olymp2.dat`, contain the women's and men's track data in a form that can be read by the function `loadfile.m`.

Read the data files into the 55×7 and 55×8 data matrices Y_1 and Y_2 and remove their column means using the function `zmean`.

a. For the women's data matrix Y_1, plot the scatterplot of the 100-meter and 200-meter columns (after removing their mean). Notice that they lie mostly along a one-dimensional subspace. Perform a PCA on these two columns and determine the percentage variances carried by the two principal components. On the scatterplot, place the two straight lines representing the two principal components, as was done in Fig.16.16.1 of the text.

b. Repeat part (a) for the following track pairs:

(100m, 800m), (100m,3000m), (100m, Marathon), (3000m, Marathon)

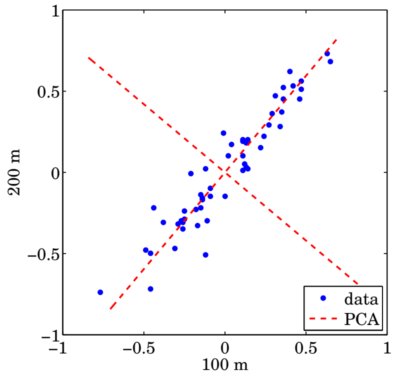
Comment on the observed clustering of the data points along one-dimensional directions. Do these make intuitive sense? For example, is a good 100m-sprinter also a good marathoner, or, is a good 100m-sprinter also a good 200m-sprinter?

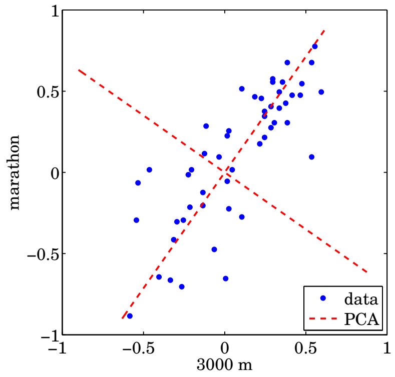
c. Next, consider the full data matrix Y_1. Working with the SVD of Y_1, perform a PCA on it and determine, print in a table, and plot the percentage variances of the principal components. Then, determine the PCA coefficients of the first two principal components and compare them with those given in the attached paper. Based on the first component determine the countries that correspond to the top 15 scores. (*Hint:* use the MATLAB function `sort`.)

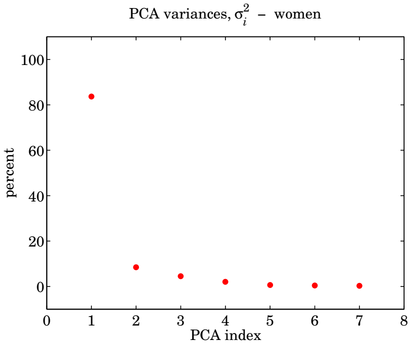
d. Repeat part (c) using the men's data matrix Y_2.

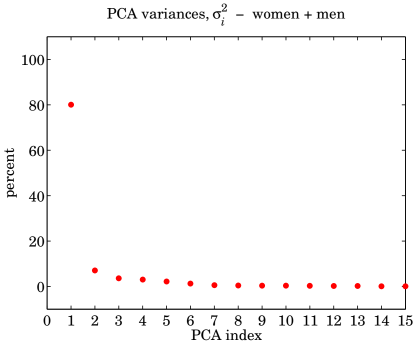
e. Next, combine the women's and men's data matrices into a single matrix by concatenating their columns, that is, $Y = [Y_1, Y_2]$. Carry out a PCA on Y and determine, print in a table, and plot the percentage variances. Determine the PCA coefficients of the first principal component. Then, determine the top 15 countries. Finally, make a table like the one below that presents the results of parts (c,d,e). Some representative graphs and results are included below.

```
rank   women              men                women + men
------------------------------------------------------------
 1     USA                USA                USA
 2     USSR               GB & NI            USSR
 3     GDR                Italy              GDR
 4     GB & NI            USSR               GB & NI
 5     FRG                GDR                FRG
 6     Czechoslovakia     FRG                Italy
 7     Canada             Australia          Canada
 8     Poland             Kenya              Poland
 9     Italy              France             Czechoslovakia
10     Finland            Belgium            Australia
11     Australia          Poland             Finland
12     Norway             Canada             France
13     New Zealand        Finland            New Zealand
14     Netherlands        Switzerland        Sweden
15     Romania            New Zealand        Netherlands
```







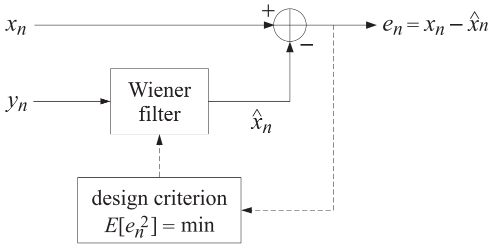


16

Adaptive Filters

16.1 Adaptive Implementation of Wiener Filters

We review briefly the solution of the Wiener filtering problem.



The general solution does not place any a priori restriction on the order of the Wiener filter. In general, an infinite number of weights is required to achieve the lowest estimation error. However, in adaptive implementations we must insist in advance that the number of filter weights be finite. This is so because the adaptation algorithm adapts each weight individually. Obviously, we cannot adapt an infinite number of weights. We will assume then, that the optimal Wiener filter is an FIR filter, say with $M+1$ weights

$$\mathbf{h} = [h_0, h_1, h_2, \dots, h_M]^T, \quad H(z) = h_0 + h_1 z^{-1} + h_2 z^{-2} + \cdots + h_M z^{-M}$$

This filter processes the available observations y_n to produce the estimate

$$\hat{x}_n = \sum_{m=0}^{M} h_m y_{n-m} = h_0 y_n + h_1 y_{n-1} + h_2 y_{n-2} + \cdots + h_M y_{n-M}$$

The weights h_m are chosen optimally so that the mean-square estimation error is minimized; that is,

$$\mathcal{E} = E[e_n^2] = \min, \quad e_n = x_n - \hat{x}_n$$

This minimization criterion leads to the orthogonality equations, which are the determining equations for the optimal weights. Writing the estimate in vector notation

$$\hat{x}_n = [h_0, h_1, \dots, h_M] \begin{bmatrix} y_n \\ y_{n-1} \\ \vdots \\ y_{n-M} \end{bmatrix} = \mathbf{h}^T \mathbf{y}(n)$$

we may write the orthogonality equations as

$$E[e_n y_{n-m}] = 0, \quad 0 \le m \le M$$

or, equivalently,

$$E[e_n \mathbf{y}(n)] = 0$$

These give the normal equations

$$E[(x_n - \hat{x}_n)\mathbf{y}(n)] = E\big[(x_n - \mathbf{h}^T\mathbf{y}(n))\mathbf{y}(n)\big] = 0, \quad \text{or,}$$

$$E[\mathbf{y}(n)\mathbf{y}(n)^T]\mathbf{h} = E[x_n \mathbf{y}(n)], \quad \text{or,}$$

$$R\,\mathbf{h} = \mathbf{r}, \quad R = E[\mathbf{y}(n)\mathbf{y}(n)^T], \quad \mathbf{r} = E[x_n \mathbf{y}(n)]$$

The optimal weights are obtained then by

$$\mathbf{h} = R^{-1}\mathbf{r} \tag{16.1.1}$$

The corresponding minimized value of the estimation error is computed by

$$\begin{aligned}\mathcal{E} = E[e_n^2] &= E[e_n(x_n - \mathbf{h}^T\mathbf{y}(n))] = E[e_n x_n] = E\big[(x_n - \mathbf{h}^T\mathbf{y}(n))x_n\big] \\ &= E[x_n^2] - \mathbf{h}^T E[\mathbf{y}(n)x_n] = E[x_n^2] - \mathbf{h}^T\mathbf{r} = E[x_n^2] - \mathbf{r}^T R^{-1}\mathbf{r}\end{aligned}$$

The normal equations, and especially the orthogonality equations, have their usual *correlation canceling* interpretations. The signal x_n being estimated can be written as

$$x_n = e_n + \hat{x}_n = e_n + \mathbf{h}^T\mathbf{y}(n)$$

It is composed of two parts, the term e_n which because of the orthogonality equations is entirely uncorrelated with $\mathbf{y}(n)$, and the second term, which is correlated with $\mathbf{y}(n)$. In effect, the filter removes from x_n any part of it that is correlated with the secondary input $\mathbf{y}(n)$; what is left, e_n, is uncorrelated with $\mathbf{y}(n)$. The Wiener filter acts as a correlation canceler. If the primary signal x_n and the secondary signal $\mathbf{y}(n)$ are in any way correlated, the filter will cancel from the output e_n any such correlations.

One difficulty with the above solution is that the statistical quantities R and $\mathbf{r}$ must be known, or at least estimated, in advance. This can be done either by block processing or adaptive processing methods. The principal advantages of block processing methods are that the design is based on a single, fixed, data record and that the length of the data record may be very short. Thus, such methods are most appropriate in applications where the *availability* of data is limited, as for example, in parametric spectrum estimation based on a single block of data, or in deconvolution applications where the data to be deconvolved are already available, for example, a still distorted picture or a recorded segment of a seismic response.

Availability of data, however, is not the only consideration. In a *changing environment*, even if more data could be collected, it may not be correct to use them in the design because stationarity may not be valid for the longer data block. Block processing methods can still be used in such cases, but the optimum filters must be *redesigned* every time the environment changes, so that the filter is always matched to the data being processed by it. This is, for example, what is done in speech processing. The input speech signal is divided into fairly short segments, with each segment assumed to arise from a stationary process, then the statistical correlations are estimated by sample correlations and the optimal prediction coefficients corresponding to each segment are computed. In a sense, this procedure is data-adaptive, but more precisely, it is block-by-block adaptive.

In other applications, however, we do not know how often to redesign and must use *adaptive* implementations that provide an automatic way of redesigning the optimum processors to continually track the environment. For example, communications and radar antennas are vulnerable to jamming through their sidelobes. Adaptive sidelobe cancelers continuously adjust themselves to steer nulls toward the jammers even when the jammers may be changing positions or new jammers may be coming into play. Another example is the equalization of unknown or changing channels, or both. In switched telephone lines the exact transmission channel is not known in advance but is established at the moment the connection is made. Similarly, in fading communications channels the channel is continuously changing. To undo the effects of the channel, such as amplitude and phase distortions, an equalizer filter must be used at the receiving end that effectively acts as an inverse to the channel. Adaptive equalizers determine automatically the characteristics of the channel and provide the required inverse response. Other applications, well-suited to adaptive implementations, are noise canceling, echo canceling, linear prediction and spectrum estimation, and system identification and control.

In this chapter we discuss several adaptation algorithms, such as the Widrow-Hoff least mean square (LMS) algorithm, the conventional recursive least squares (RLS) algorithm, the fast RLS algorithms, and the adaptive lattice algorithms and present some of their applications [1341–1349]. A typical adaptive implementation of a Wiener filter is depicted in Fig. 16.1.1.

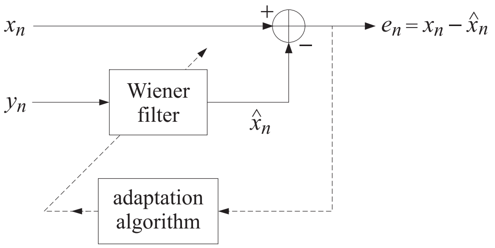


Fig. 16.1.1 Adaptive Wiener filter.

The adaptation algorithm continuously monitors the output error signal e_n and attempts to minimize the output power $E[e_n^2]$, or, equivalently tries to decorrelate e_n from the secondary input y_n. At each time instant n, the current values of the weights are used to perform the filtering operation. The computed output e_n is then used by the adaptation part of the algorithm to change the weights in the direction of their optimum values. As processing of the input signals x_n and y_n takes place and the filter gradually learns the statistics of these inputs, its weights gradually converge to their optimum values given by the Wiener solution (16.1.1). Clearly, the input statistics must remain unchanged for at least as long as it takes the filter to learn it and converge to its optimum configuration. If, after convergence, the input statistics should change, the filter will respond by readjusting its weights to their new optimum values, and so on. In other words, the adaptive filter will track the non-stationary changes of the input statistics as long as such changes occur slowly enough for the filter to converge between changes. The three basic issues in any adaptive implementation are:

1. The learning or convergence *speed* of the algorithm.
2. The computational *complexity* of the algorithm.
3. The numerical *accuracy and stability* of the algorithm.

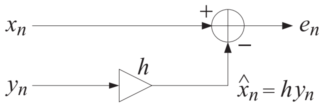
The convergence speed is an important factor because it determines the maximum rate of change of the input non-stationarities that can be usefully tracked by the filter. The computational complexity refers to the number of operations required to update the filter from one time instant to the next. The table below shows how various adaptive algorithms fare under these requirements.

algorithm	speed	complexity	stability
LMS	slow	simple	stable
RLS	fast	complex	stable
Fast RLS	fast	simple	unstable
Lattice	fast	simple	stable

Only adaptive lattice algorithms satisfy all three requirements. We will discuss these algorithms in detail later on. In the next section we begin with the LMS algorithm because it is the simplest and most widely used. We finish this section with the obvious remark that adaptive or block processing optimal filter designs, regardless of type, cannot do any better than the theoretical Wiener solution. The optimal filter, therefore, should be first analyzed theoretically to determine if it is worth using it in the application at hand.

16.2 Correlation Canceler Loop (CCL)

To illustrate the basic principles behind adaptive filters, consider the simplest possible filter, that is, a filter with only one weight



The weight h must be selected optimally so as to produce the best possible estimate of x_n:

$$\hat{x}_n = hy_n$$

The estimation error is expressed as

$$\begin{aligned} \mathcal{E} &= E[e_n^2] = E\big[(x_n - hy_n)^2\big] = E[x_n^2] - 2hE[x_n y_n] + E[y_n^2]h^2 \\ &= E[x_n^2] - 2hr + Rh^2 \end{aligned} \tag{16.2.1}$$

The minimization condition is

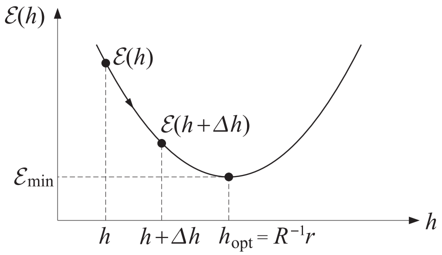
$$\frac{\partial \mathcal{E}}{\partial h} = 2E\left[e_n \frac{\partial e_n}{\partial h}\right] = -2E[e_n y_n] = -2r + 2Rh = 0 \tag{16.2.2}$$

which gives the optimum solution $h_{\text{opt}} = R^{-1}r$, and also shows the correlation cancellation condition $E[e_n y_n] = 0$. The adaptive implementation is based on solving the equation

$$\frac{\partial \mathcal{E}}{\partial h} = 0 \tag{16.2.3}$$

iteratively, using a gradient-descent method. The dependence of the error $\mathcal{E}$ on the filter parameter h is parabolic, with an absolute minimum occurring at the above optimal value $h_{\text{opt}} = R^{-1}r$.

This is shown below



In the adaptive version, the filter parameter h is made *time-dependent*, $h(n)$, and is updated from one time instant to the next as follows

$$h(n+1)=h(n)+\Delta h(n) \tag{16.2.4}$$

where $\Delta h(n)$ is a correction term that must be chosen properly in order to ensure that eventually the time-varying weight $h(n)$ will converge to the optimal value:

$$h(n)\to h_{\rm opt}=R^{-1}r \quad \text{as} \quad n\to\infty$$

The filtering operation is now given by the still *linear* but *time non-invariant* form

$$\hat{x}_n = h(n)y_n \tag{16.2.5}$$

The computation of the estimate at the next time instant should be made with the new weight, that is,

$$\hat{x}_{n+1} = h(n+1)y_{n+1}$$

and so on. The simplest way to choose the correction term $\Delta h(n)$ is the gradient-descent, or steepest-descent, method. The essence of the method is this: It is required that the change $h\to h+\Delta h$ must move the performance index closer to its minimum than before, that is, Δh must be such that

$$\mathcal{E}(h+\Delta h)\le\mathcal{E}(h)$$

Therefore, if we always demand this, the repetition of the procedure will lead to smaller and smaller values of $\mathcal{E}$ until the smallest value has been attained. Assuming that Δh is sufficiently small, we may expand to first order and obtain the condition

$$\mathcal{E}(h)+\Delta h\,\frac{\partial\mathcal{E}(h)}{\partial h}\le\mathcal{E}(h)$$

If Δh is selected as the *negative gradient* $-\mu(\partial\mathcal{E}/\partial h)$ then this inequality will be guaranteed, that is, if we choose

$$\Delta h = -\mu\,\frac{\partial\mathcal{E}(h)}{\partial h} \tag{16.2.6}$$

then the inequality is indeed satisfied:

$$\mathcal{E}(h)+\Delta h\,\frac{\partial\mathcal{E}(h)}{\partial h}=\mathcal{E}(h)-\mu\left|\frac{\partial\mathcal{E}(h)}{\partial h}\right|^2\le\mathcal{E}(h)$$

The adaptation parameter μ must be small enough to justify keeping only the first-order terms in the above Taylor expansion. Applying this idea to our little adaptive filter, we choose the correction $\Delta h(n)$ according to Eq. (16.2.6), so that

$$h(n+1)=h(n)+\Delta h(n)=h(n)-\mu\,\frac{\partial\mathcal{E}(h(n))}{\partial h} \tag{16.2.7}$$

Using the expression for the gradient $\dfrac{\partial\mathcal{E}(h)}{\partial h}=-2r+2Rh$, we find

$$\begin{aligned} h(n+1) &= h(n)-\mu\big[-2r+2Rh(n)\big] \\ &= (1-2\mu R)h(n)+2\mu r \end{aligned}$$

This difference equation may be solved in closed form. For example, using z-transforms with any initial conditions $h(0)$, we find

$$h(n)=h_{\rm opt}+(1-2\mu R)^n\big(h(0)-h_{\rm opt}\big) \tag{16.2.8}$$

where $h_{\rm opt}=R^{-1}r$. The coefficient $h(n)$ will converge to its optimal value $h_{\rm opt}$, regardless of the starting value $h(0)$, provided μ is selected such that

$$|1-2\mu R|<1$$

or, $-1<1-2\mu R<1$, or since μ must be positive (to be in the negative direction of the gradient), μ must satisfy

$$0<\mu<\frac{1}{R} \tag{16.2.9}$$

To select μ, one must have some a priori knowledge of the magnitude of the input variance $R=E[y_n^2]$. Such choice for μ will guarantee convergence, but the speed of convergence is controlled by how close the number $1-2\mu R$ is to one. The closer it is to unity, the slower the speed of convergence. As μ is selected closer to zero, the closer $1-2\mu R$ moves towards one, and thus the slower the convergence rate. Thus, the adaptation parameter μ must be selected to be small enough to guarantee convergence but not too small to cause a very slow convergence.

16.3 The Widrow-Hoff LMS Adaptation Algorithm

The purpose of the discussion in Sec. 16.2 was to show how the original Wiener filtering problem could be recast in an iterative form. From the practical point of view, this reformulation is still not computable since the adaptation of the weights requires a priori knowledge of the correlations R and r. In the Widrow-Hoff algorithm the above adaptation algorithm is replaced with one that is computable [1341,1342]. The gradient that appears in Eq. (16.2.7)

$$h(n+1)=h(n)-\mu\,\frac{\partial\mathcal{E}(h(n))}{\partial h}$$

is replaced by an *instantaneous* gradient by *ignoring* the expectation instructions, that is, the theoretical gradient

$$\frac{\partial\mathcal{E}(h(n))}{\partial h}=-2E[e_ny_n]=-2r+2Rh(n)=-2E[x_ny_n]+2E[y_n^2]h(n)$$

is replaced by

$$\frac{\partial\mathcal{E}}{\partial h}=-2e_ny_n=-2\big(x_n-h(n)y_n\big)y_n=-2x_ny_n+2y_n^2h(n) \tag{16.3.1}$$

so that the weight-adjustment algorithm becomes

$$h(n+1)=h(n)+2\mu e_ny_n \tag{16.3.2}$$

In summary, the required computations are done in the following order:

1. At time n, the filter weight $h(n)$ is available.

2. Compute the filter output $\hat{x}_n = h(n)y_n$.
3. Compute the estimation error $e_n = x_n - \hat{x}_n$.
4. Compute the next filter weight $h(n+1) = h(n) + 2\mu e_n y_n$.
5. Go to next time instant $n \to n+1$.

The following remarks are in order:

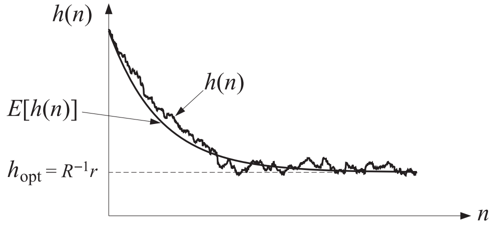
1. The output error e_n is fed back and used to control the adaptation of the filter weight $h(n)$.
2. The filter tries to decorrelate the secondary signal from the output e_n. This, is easily seen as follows: If the weight $h(n)$ has more or less reached its optimum value, then $h(n+1) \simeq h(n)$, and the adaptation equation implies also approximately that $e_n y_n \simeq 0$.
3. Actually, the weight $h(n)$ never really reaches the theoretical limiting value $h_{\text{opt}} = R^{-1}r$. Instead, it stabilizes about this value, and continuously fluctuates about it.
4. The approximation of ignoring the expectation instruction in the gradient is known as the *stochastic approximation*. It complicates the mathematical aspects of the problem considerably. Indeed, the difference equation

$$h(n+1) = h(n) + 2\mu e_n y_n = h(n) + 2\mu\big(x_n - h(n)y_n\big)y_n$$

   makes $h(n)$ depend on the random variable y_n in highly nonlinear fashion, and it is very difficult to discuss even the average behavior of $h(n)$.
5. In discussing the average behavior of the weight $h(n)$, the following approximation is typically (almost invariably) made in the literature

$$\begin{aligned} E\big[h(n+1)\big] &= E\big[h(n)\big] + 2\mu E[x_n y_n] - 2\mu E\big[h(n)y_n^2\big] \\ &= E\big[h(n)\big] + 2\mu E[x_n y_n] - 2\mu E\big[h(n)\big]E\big[y_n^2\big] \\ &= E\big[h(n)\big] + 2\mu r - 2\mu E\big[h(n)\big]R \end{aligned}$$

   where in the last term, the expectation $E\big[h(n)\big]$ was factored out, as though $h(n)$ were independent of y_n. With this approximation, the average $E\big[h(n)\big]$ satisfies the same difference equation as before with solution given by Eq. (16.2.8). Typically, the weight $h(n)$ will be fluctuating about the theoretical convergence curve as it converges to the optimal value, as shown below



   After convergence, the adaptive weight $h(n)$ continuously fluctuates about the Wiener solution h_{opt}. A measure of these fluctuations is the mean-square deviation of $h(n)$ from h_{opt}, that is, $E\big[\big(h(n) - h_{\text{opt}}\big)^2\big]$. Under some restrictive conditions, this quantity has been calculated [1350] to be

$$E\big[\big(h(n) - h_{\text{opt}}\big)^2\big] \to \mu\mathcal{E}_{\min} \quad \text{(for large } n\text{)}$$

   where $\mathcal{E}_{\min}$ is the minimized value of the performance index (16.2.1). Thus, the adaptation parameter μ controls the size of these fluctuations. This gives rise to the basic trade-off of the LMS algorithm: to obtain high accuracy in the converged weights (small fluctuations), a small value of μ is required, but this will slow down the convergence rate.

A realization of the CCL is shown in Fig. 16.3.1. The filtering part of the realization must be clearly distinguished from the feedback control loop that performs the adaptation of the filter weight.

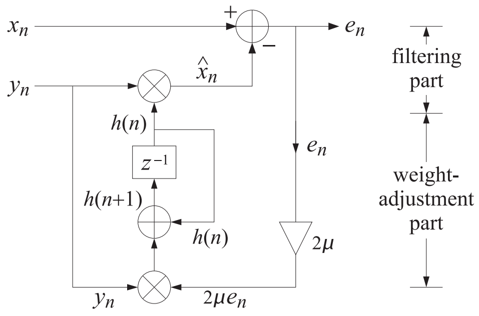


Fig. 16.3.1 Correlation canceler loop.

Historically, the correlation canceler loop was introduced in adaptive antennas as a *sidelobe canceler* [1351–1356] The CCL is the simplest possible adaptive filter, and forms the elementary *building block* of more complicated, higher-order adaptive filters.

We finish this section by presenting a simulation example of the CCL loop. The primary signal x_n was defined by

$$x_n = -0.8y_n + u_n$$

where the first term represents that part of x_n which is correlated with y_n. The part u_n is not correlated with y_n. The theoretical value of the CCL weight is found as follows:

$$r = E[x_n y_n] = -0.8E[y_n y_n] + E[u_n y_n] = -0.8R + 0 \quad \Rightarrow \quad h_{\text{opt}} = R^{-1}r = -0.8$$

The corresponding output of the CCL will be $\hat{x}_n = h_{\text{opt}}y_n = -0.8y_n$, and therefore it will completely cancel the first term of x_n leaving at the output $e_n = x_n - \hat{x}_n = u_n$.

In the simulation we generated 1000 samples of a zero-mean white-noise signal y_n of variance 0.1, and another independent set of 1000 samples of a zero-mean white-noise signal u_n also of variance 0.1, and computed x_n. The adaptation algorithm was initialized, as is usually done, to zero initial weight $h(0) = 0$. Fig. 16.3.2 shows the transient behavior of the adaptive weight $h(n)$, as well as the theoretical weight $E\big[h(n)\big]$, as a function of the number of iterations n, for the two values of μ, $\mu = 0.03$ and $\mu = 0.01$.

Note that in both cases, the adaptive weight converges to the theoretical value $h_{\text{opt}} = -0.8$, and that the smaller μ is slower but the fluctuations are also smaller. After the adaptive weight has reached its asymptotic value, the CCL begins to operate optimally, removing the correlated part of x_n from the output e_n.

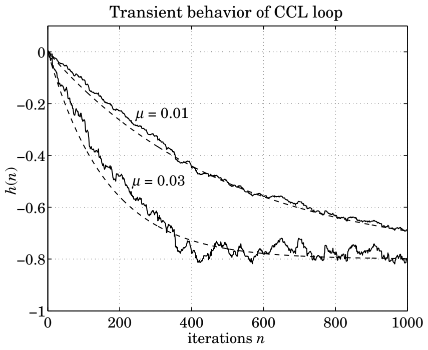
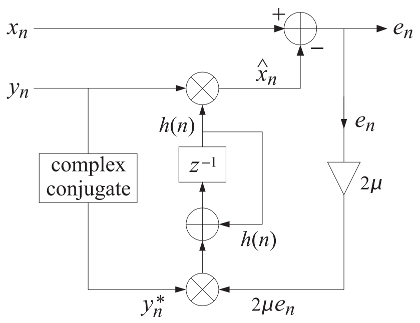


Fig. 16.3.2 Transient behavior of theoretical (dashed) and adaptive weights $h(n)$.

Later on we will consider the complex-valued version of adaptive Wiener filters. Their elementary building block is the complex CCL shown below



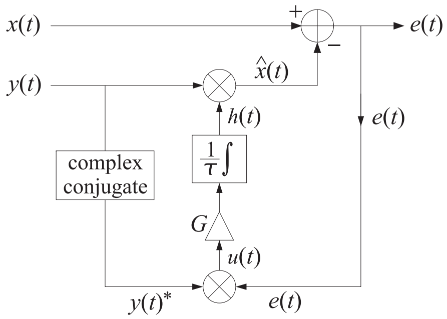
The performance index is now

$$\mathcal{E} = E\big[|e_n|^2\big] = E\big[|x_n - hy_n|^2\big] = \min$$

with optimum solution

$$h_{\text{opt}} = R^{-1}r\,, \quad R = E[y_n^* y_n]\,, \quad r = E[x_n y_n^*]$$

Analog implementations of the CCL are used in adaptive antennas. An analog CCL is shown below



where a high gain amplifier G and an ordinary RC-type integrator are used. If τ denotes the RC time constant of the integrator, the weight updating part of the CCL is

$$\tau\dot{h}(t)+h(t)= Gu(t)= Ge(t)y^*(t)$$

The performance of the analog CCL can be analyzed by replacing the adaptive weight $h(t)$ by its statistical average, satisfying

$$\tau\dot{h}(t)+h(t)= GE\big[e(t)y^*(t)\big] = GE\big[\big(x(t)-h(t)y(t)\big)y^*(t)\big]$$

or, defining $R = E\big[y(t)y^*(t)\big]$ and $r = E\big[x(t)y^*(t)\big]$,

$$\tau\dot{h}(t)+h(t)= Gr - GRh(t)$$

with solution for $t \geq 0$:

$$h(t)= h_{\text{opt}} + \big(h(0)-h_{\text{opt}}\big)e^{-at}$$

where h_{opt} is the asymptotic value

$$h_{\text{opt}} = (1 + GR)^{-1}Gr$$

Thus, a high gain G is needed to produce an asymptotic value close to the theoretical Wiener solution $R^{-1}r$. The time constant of adaptation is given by

$$\frac{1}{a} = \frac{\tau}{1 + GR}$$

Note that this particular implementation always converges and the speed of convergence is still inversely dependent on R.

16.4 Adaptive Linear Combiner

A straightforward generalization of the correlation canceler loop is the adaptive linear combiner, where one has available a main signal x_n and a number of secondary signals $y_m(n)$, $m = 0, 1, \dots, M$. These $(M + 1)$ secondary signals are to be linearly combined with appropriate weights $h_0, h_1, \dots, h_M$ to form an estimate of x_n:

$$\hat{x}_n = h_0 y_0(n)+h_1 y_1(n)+\cdots + h_M y_M(n)= [h_0, h_1, \dots, h_M]\begin{bmatrix} y_0(n) \\ y_1(n) \\ \vdots \\ y_M(n) \end{bmatrix} = \mathbf{h}^T\mathbf{y}(n)$$

A realization of this is shown in Fig. 16.4.1. The adaptive linear combiner is used in adaptive radar and sonar arrays [1351–1355]. It also encompasses the case of the ordinary FIR, or transversal, Wiener filter [1342].

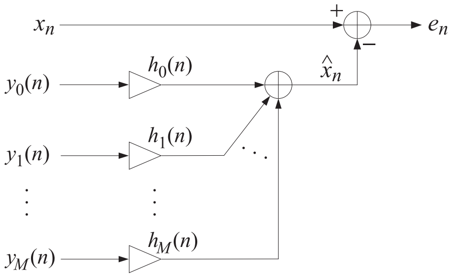


Fig. 16.4.1 Linear combiner.

The optimal weights h_m minimize the estimation error squared

$$\mathcal{E} = E[e_n^2] = \min\,, \quad e_n = x_n - \hat{x}_n$$

The corresponding orthogonality equations state that the estimation error be orthogonal (decorrelated) to each secondary signal $y_m(n)$:

$$\frac{\partial \mathcal{E}}{\partial h_m} = 2E\left[e_n \frac{\partial e_n}{\partial h_m}\right] = -2E[e_n y_m(n)] = 0\,, \quad 0 \le m \le M$$

or, in vector form

$$E[e_n \mathbf{y}(n)] = 0 \quad \Rightarrow \quad E[x_n \mathbf{y}(n)] - E[\mathbf{y}(n)\mathbf{y}^T(n)]\mathbf{h} = \mathbf{r} - R\mathbf{h} = 0$$

with optimum solution $\mathbf{h}_{\text{opt}} = R^{-1}\mathbf{r}$.

The adaptive implementation is easily obtained by allowing the weights to become time-dependent, $\mathbf{h}(n)$, and updating them in time according to the gradient-descent algorithm

$$\mathbf{h}(n+1) = \mathbf{h}(n) - \mu \frac{\partial \mathcal{E}(\mathbf{h}(n))}{\partial \mathbf{h}}$$

with instantaneous gradient

$$\frac{\partial \mathcal{E}}{\partial \mathbf{h}} = -2E[e_n \mathbf{y}(n)] \to -2e_n \mathbf{y}(n)$$

so that

$$\mathbf{h}(n+1) = \mathbf{h}(n) + 2\mu e_n \mathbf{y}(n)$$

or, component-wise

$$h_m(n+1) = h_m(n) + 2\mu e_n y_m(n)\,, \quad 0 \le m \le M \tag{16.4.1}$$

The computational algorithm is summarized below:

1. $\hat{x}_n = h_0(n)y_0(n) + h_1(n)y_1(n) + \cdots + h_M(n)y_M(n)$
2. $e_n = x_n - \hat{x}_n$
3. $h_m(n+1) = h_m(n) + 2\mu e_n y_m(n)\,, \quad 0 \le m \le M$

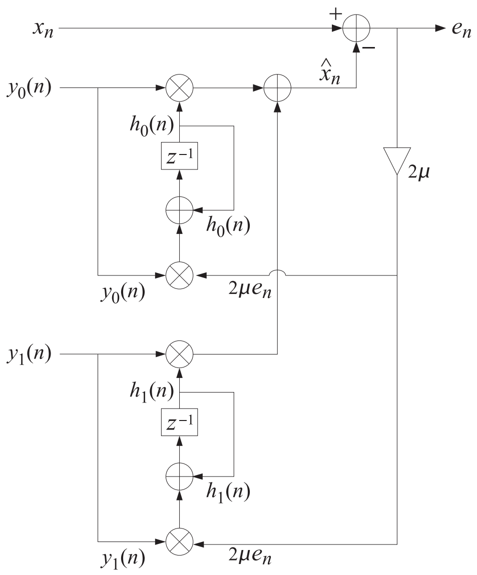


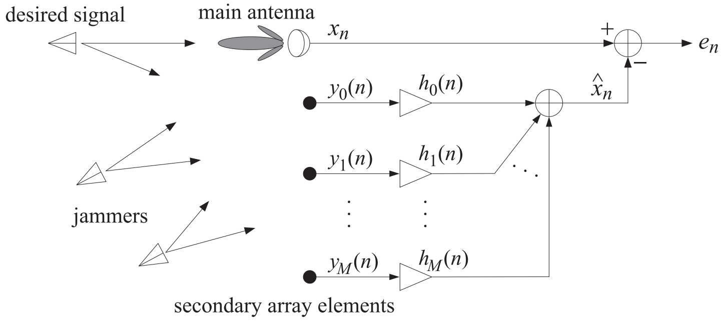
Fig. 16.4.2 Adaptive linear combiner.

It is evident that each weight $h_m(n)$ is being adapted by its own correlation canceler loop, while all weights use the same feedback error e_n to control their loops. The case of two weights $(M = 1)$ is shown in Fig. 16.4.2.

The adaptive linear combiner has two major applications:

1. Adaptive sidelobe canceler.
2. Adaptive FIR Wiener filter.

The two cases differ only in the way the inputs to the linear combiner are supplied. The linear combiner part, performing the optimum processing, is the same in both cases. The time series case is discussed in the next section. The array problem is depicted below.



It consists of a main and a number of secondary antennas. The main antenna is highly directional and oriented toward the desired signal. Jammers picked up by the sidelobes of the main antenna and by the secondary antennas will tend to be canceled because the adaptive linear combiner, acting as a correlation canceler, will adjust itself to cancel that part of the main signal that is correlated with the secondary ones. The desired signal may also be canceled partially if it is picked up by the secondary antennas. Strong jammers, however, will generally dominate and as a result the canceler will configure itself to cancel them. The cancellation of the desired signal can also be prevented by imposing additional constraints on the filter weights that can sustain the beam in the desired look-direction.

The adaptation speed of the adaptive canceler is affected by the relative power levels of the jammers. If there are jammers with greatly differing powers, the overall adaptation speed may be slow. The stronger jammers tend to be canceled faster; the weaker ones more slowly. Qualitatively this may be understood by inspecting, for example, expression (14.2.32). The power levels P_i of the plane waves act as *penalty* factors in the performance index, that is, the minimization of the performance index will tend to favor first the largest terms in the sum. This limitation of the LMS algorithm has led to the development of alternative algorithms, such as adaptive Gram-Schmidt preprocessors or RLS, in which all jammers get canceled equally fast.

16.5 Adaptive FIR Wiener Filter

The adaptive FIR or transversal filter is a special case of the adaptive linear combiner. In this case, there is only one secondary signal y_n. The required $M+1$ signals $y_m(n)$ are provided as delayed replicas of y_n, that is,

$$y_m(n)=y_{n-m} \tag{16.5.1}$$

A realization is shown in Fig. 16.5.1. The estimate of x_n is

$$\hat{x}_n=\sum_{m=0}^{M}h_m(n)y_{n-m}=h_0(n)y_n+h_1(n)y_{n-1}+\cdots+h_M(n)y_{n-M}$$

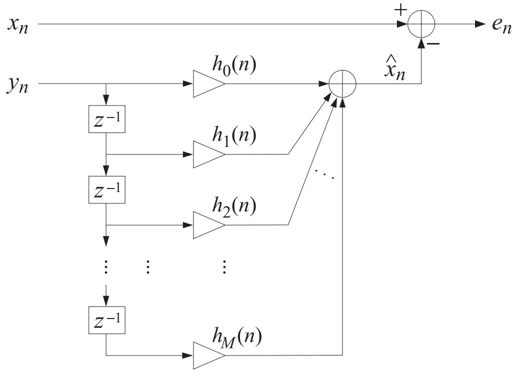


Fig. 16.5.1 Adaptive FIR Wiener filter.

The time-varying filter weights $h_m(n)$ are continuously updated according to the gradient-descent LMS algorithm

$$h_m(n+1)=h_m(n)+2\mu e_n y_m(n)\,,\quad \text{or,}$$
$$h_m(n+1)=h_m(n)+2\mu e_n y_{n-m}\,,\quad 0\le m\le M \tag{16.5.2}$$

Each weight is therefore updated by its own CCL. Again, we summarize the computational steps:

1. Compute the estimate $\hat{x}_n=\sum_{m=0}^{M}h_m(n)y_{n-m}$
2. Compute the error signal $e_n=x_n-\hat{x}_n$
3. Adjust the weights $h_m(n+1)=h_m(n)+2\mu e_n y_{n-m}\,,\quad 0\le m\le M$

The function **lms** is an implementation of the algorithm. With a minor modification it can also be used for the more general adaptive linear combiner. Each call to the function reads a pair of input samples $\{x_n,y_n\}$, performs the filtering operation to produce the output pair $\{\hat{x}_n,e_n\}$, updates the filter coefficients $h_m(n)$ to their new values $h_m(n+1)$ to be used by the next call, and updates the internal state of the filter. It is essentially the function **dwf** with the weight adaptation part added to it.

Next, we present the same simulation example as that given in Section Sec. 16.3, but it is now approached with a two-tap adaptive filter $(M=1)$. The filtering equation is in this case

$$\hat{x}_n=h_0(n)y_n+h_1(n)y_{n-1}$$

The theoretical Wiener solution is found as follows: First note that

$$\begin{aligned}R_{xy}(k)&=E[x_{n+k}y_n]=E\big[(-0.8y_{n+k}+u_{n+k})y_n\big]=-0.8E[y_{n+k}y_n]\\&=-0.8R_{yy}(k)=-0.8R(k)\end{aligned}$$

Thus, the cross correlation vector is

$$\mathbf{r}=\begin{bmatrix}R_{xy}(0)\\R_{xy}(1)\end{bmatrix}=-0.8\begin{bmatrix}R(0)\\R(1)\end{bmatrix}$$

and the Wiener solution becomes:

$$\begin{aligned}\mathbf{h}=R^{-1}\mathbf{r}&=\begin{bmatrix}R(0)&R(1)\\R(1)&R(0)\end{bmatrix}^{-1}\begin{bmatrix}-0.8R(0)\\-0.8R(1)\end{bmatrix}\\&=\frac{-0.8}{R(0)^2-R(1)^2}\begin{bmatrix}R(0)&-R(1)\\-R(1)&R(0)\end{bmatrix}\begin{bmatrix}R(0)\\R(1)\end{bmatrix}=\begin{bmatrix}-0.8\\0\end{bmatrix}\end{aligned}$$

We could have expected that h_1 is zero, since the signal x_n does not depend on y_{n-1}, but only on y_n. The adaptive weights were both initialized to the (arbitrary) value of $h_0(0)=h_1(0)=-0.4$, and the value of μ was 0.03. Fig. 16.5.2 shows the two adaptive weights $h_0(n)$ and $h_1(n)$ as a function of n, converging to their optimal values of $h_0=-0.8$ and $h_1=0$.

How does one select the filter order M? The rule is that the filter must have at least as many delays as that part of x_n which is correlated with y_n. To see this, suppose x_n is related to y_n by

$$x_n=c_0y_n+c_1y_{n-1}+\cdots+c_Ly_{n-L}+u_n \tag{16.5.3}$$

where u_n is uncorrelated with y_n. Then, the filter order must be at least L. If $M\ge L$, we can write:

$$x_n=c_0y_n+c_1y_{n-1}+\cdots+c_My_{n-M}+u_n=\mathbf{c}^T\mathbf{y}(n)+u_n$$

where $\mathbf{c}$ is the extended vector having $c_i=0$ for $L+1\le i\le M$. The cross-correlation between x_n and $\mathbf{y}(n)$ is

$$\mathbf{r}=E\big[x_n\mathbf{y}(n)\big]=E\big[(\mathbf{y}^T(n)\mathbf{c})\mathbf{y}(n)\big]=E\big[\mathbf{y}(n)\mathbf{y}^T(n)\big]\mathbf{c}=R\mathbf{c}$$

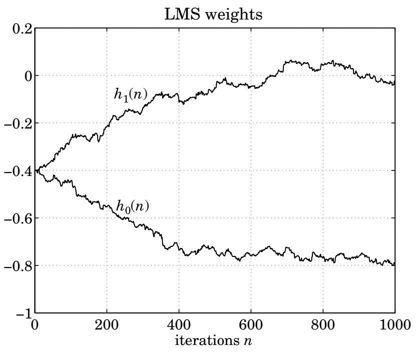


Fig. 16.5.2 Transient behavior of FIR adaptive filter.

Thus, the Wiener solution will be $\mathbf{h}=R^{-1}\mathbf{r}=\mathbf{c}$. This, in turn, implies the complete cancellation of the y-dependent part of x_n. Indeed, $\hat{x}_n=\mathbf{h}^T\mathbf{y}(n)=\mathbf{c}^T\mathbf{y}(n)$ and

$$e_n=x_n-\hat{x}_n=\big(\mathbf{c}^T\mathbf{y}(n)+u_n\big)-\mathbf{c}^T\mathbf{y}(n)=u_n$$

What happens if we underestimate the filter order and choose $M<L$? In this case, we expect to cancel completely the first M terms of Eq. (16.5.3) and to cancel the remaining terms as much as possible. To see this, we separate out the first M terms writing

$$x_n=[c_0,\dots,c_M]\begin{bmatrix}y_n\\ \vdots\\ y_{n-M}\end{bmatrix}+[c_{M+1},\dots,c_L]\begin{bmatrix}y_{n-M-1}\\ \vdots\\ y_{n-L}\end{bmatrix}+u_n\equiv\mathbf{c}_1^T\mathbf{y}_1(n)+\mathbf{c}_2^T\mathbf{y}_2(n)+u_n$$

The problem of estimating x_n using an Mth order filter is equivalent to the problem of estimating x_n from $\mathbf{y}_1(n)$. The cross-correlation between x_n and $\mathbf{y}_1(n)$ is

$$E[x_n\mathbf{y}_1(n)]=E[\mathbf{y}_1(n)\mathbf{y}_1^T(n)]\mathbf{c}_1+E[\mathbf{y}_1(n)\mathbf{y}_2^T(n)]\mathbf{c}_2$$

It follows that the optimum estimate of x_n is

$$\begin{aligned}\hat{x}_n&=E[x_n\mathbf{y}_1^T(n)]E[\mathbf{y}_1(n)\mathbf{y}_1^T(n)]^{-1}\mathbf{y}_1(n)\\&=\big(\mathbf{c}_1^TE[\mathbf{y}_1(n)\mathbf{y}_1^T(n)]+\mathbf{c}_2^TE[\mathbf{y}_2(n)\mathbf{y}_1^T(n)]\big)E[\mathbf{y}_1(n)\mathbf{y}_1^T(n)]^{-1}\mathbf{y}_1(n)\\&=\big(\mathbf{c}_1^T+\mathbf{c}_2^TE[\mathbf{y}_2(n)\mathbf{y}_1^T(n)]E[\mathbf{y}_1(n)\mathbf{y}_1^T(n)]^{-1}\big)\mathbf{y}_1(n)\\&=\mathbf{c}_1^T\mathbf{y}_1(n)+\mathbf{c}_2^T\hat{\mathbf{y}}_{2/1}(n)\end{aligned}$$

where $\hat{\mathbf{y}}_{2/1}(n)=E[\mathbf{y}_2(n)\mathbf{y}_1^T(n)]E[\mathbf{y}_1(n)\mathbf{y}_1^T(n)]^{-1}\mathbf{y}_1(n)$ is recognized as the optimum estimate of $\mathbf{y}_2(n)$ based on $\mathbf{y}_1(n)$. Thus, the estimation error will be

$$\begin{aligned}e_n=x_n-\hat{x}_n&=[\mathbf{c}_1^T\mathbf{y}_1(n)+\mathbf{c}_2^T\mathbf{y}_2(n)+u_n]-[\mathbf{c}_1\mathbf{y}_1(n)+\mathbf{c}_2^T\hat{\mathbf{y}}_{2/1}(n)]\\&=\mathbf{c}_2^T[\mathbf{y}_2(n)-\hat{\mathbf{y}}_{2/1}(n)]+u_n\end{aligned}$$

which shows that the $\mathbf{y}_1(n)$ part is removed completely, and the $\mathbf{y}_2(n)$ part is removed as much as possible.

16.6 Speed of Convergence

The convergence properties of the LMS algorithm [1342,1350,1356] may be discussed by restoring the expectation values where they should be, that is

$$\frac{\partial\mathcal{E}}{\partial\mathbf{h}}=-2E[e_n\mathbf{y}(n)],\quad\mathbf{y}(n)=\begin{bmatrix}y_0(n)\\y_1(n)\\ \vdots\\y_M(n)\end{bmatrix}=\begin{bmatrix}y_n\\y_{n-1}\\ \vdots\\y_{n-M}\end{bmatrix}$$

resulting in the difference equation for the weight vector

$$\begin{aligned}\mathbf{h}(n+1)&=\mathbf{h}(n)-\mu\frac{\partial\mathcal{E}}{\partial\mathbf{h}}\\&=\mathbf{h}(n)+2\mu E[e_n\mathbf{y}(n)]\\&=\mathbf{h}(n)+2\mu\{E[x_n\mathbf{y}(n)]-E[\mathbf{y}(n)\mathbf{y}^T(n)]\mathbf{h}(n)\}\\&=\mathbf{h}(n)+2\mu\mathbf{r}-2\mu R\mathbf{h}(n)\end{aligned}$$

or,

$$\mathbf{h}(n+1)=(I-2\mu R)\mathbf{h}(n)+2\mu\mathbf{r}\tag{16.6.1}$$

where $\mathbf{r}=E[x_n\mathbf{y}(n)]$ and $R=E[\mathbf{y}(n)\mathbf{y}^T(n)]$. The difference equation (16.6.1) has the following solution, where $\mathbf{h}_{\text{opt}}=R^{-1}\mathbf{r}$

$$\mathbf{h}(n)=\mathbf{h}_{\text{opt}}+(I-2\mu R)^n(\mathbf{h}(0)-\mathbf{h}_{\text{opt}})$$

Convergence to $\mathbf{h}_{\text{opt}}$ requires that the quantity $(1-2\mu\lambda)$, for every eigenvalue λ of R, have magnitude less than one (we assume that R has full rank and therefore all its eigenvalues are positive):

$$|1-2\mu\lambda|<1\quad\Leftrightarrow\quad-1<1-2\mu\lambda<1\quad\Leftrightarrow\quad0<\mu<\frac{1}{\lambda}$$

This condition will be guaranteed if we require this inequality for $\lambda_{\max}$, the maximum eigenvalue:

$$0<\mu<\frac{1}{\lambda_{\max}}\tag{16.6.2}$$

Note that $\lambda_{\max}$ can be bounded from above by

$$\lambda_{\max}<\operatorname{tr}(R)=\sum_{i=0}^{M}R_{ii}=\sum_{i=0}^{M}R(0)=(M+1)R(0)$$

and one may require instead $\mu<1/\big((M+1)R(0)\big)$. As for the speed of convergence, suppose that μ is selected half-way within its range (16.6.2), near $0.5/\lambda_{\max}$, then the rate of convergence will depend on the slowest converging term of the form $(1-2\mu\lambda)^n$ that is, the term having $|1-2\mu\lambda|$ as close to one as possible. This occurs for the smallest eigenvalue $\lambda=\lambda_{\min}$. Thus, the slowest converging term is effectively given by $(1-2\mu\lambda_{\min})^n=(1-\lambda_{\min}/\lambda_{\max})^n$. The effective time constant in seconds is obtained by writing $t=nT$, where T is the sampling period, and using the approximation

$$\left(1-\frac{\lambda_{\min}}{\lambda_{\max}}\right)^n\simeq\exp\left(-\frac{\lambda_{\min}}{\lambda_{\max}}n\right)=e^{-t/\tau}$$

where

$$\tau=T\frac{\lambda_{\max}}{\lambda_{\min}}$$

The *eigenvalue spread* $\lambda_{\max}/\lambda_{\min}$ controls, therefore, the *speed of convergence*. The convergence can be as fast as one sampling instant T if the eigenvalue spread is small, i.e., $\lambda_{\max}/\lambda_{\min} \simeq 1$. But, the convergence will be slow if the eigenvalue spread is large. As we shall see shortly, a large spread in the eigenvalues of the covariance matrix R corresponds to a highly self-correlated signal y_n.

Thus, we obtain the general qualitative result that in situations where the secondary signal is strongly self-correlated, the convergence of the gradient-based LMS algorithm will be slow. In many applications, such as channel equalization, the convergence must be as quick as possible. Alternative adaptation schemes exist that combine the computational simplicity of the LMS algorithm with a fast speed of convergence. Examples are the fast RLS and the adaptive lattice algorithms.

The possibility of accelerating the convergence rate may be seen by considering a more general version of the gradient-descent algorithm in which the time update for the weight vector is chosen as

$$\Delta\mathbf{h} = -\mathcal{M}\,\frac{\partial\mathcal{E}}{\partial\mathbf{h}} \tag{16.6.3}$$

where $\mathcal{M}$ is a *positive definite and symmetric* matrix. The LMS steepest descent case is obtained as a special case of this when $\mathcal{M}$ is proportional to the unit matrix I, $\mathcal{M} = \mu I$. This choice guarantees convergence towards the minimum of the performance index $\mathcal{E}(\mathbf{h})$, indeed,

$$\mathcal{E}(\mathbf{h}+\Delta\mathbf{h}) \simeq \mathcal{E}(\mathbf{h}) + \Delta\mathbf{h}^T\left(\frac{\partial\mathcal{E}}{\partial\mathbf{h}}\right) = \mathcal{E}(\mathbf{h}) - \left(\frac{\partial\mathcal{E}}{\partial\mathbf{h}}\right)^T\mathcal{M}\left(\frac{\partial\mathcal{E}}{\partial\mathbf{h}}\right) \le \mathcal{E}(\mathbf{h})$$

Since the performance index is

$$\mathcal{E} = E[e_n^2] = E\big[\big(x_n - \mathbf{h}^T\mathbf{y}(n)\big)^2\big] = E[x_n^2] - 2\mathbf{h}^T\mathbf{r} + \mathbf{h}^T R\mathbf{h}$$

it follows that $\partial\mathcal{E}/\partial\mathbf{h} = -2(\mathbf{r} - R\mathbf{h})$, and the difference equation for the adaptive weights becomes

$$\mathbf{h}(n+1) = \mathbf{h}(n) + \Delta\mathbf{h}(n) = \mathbf{h}(n) + 2\mathcal{M}\big(\mathbf{r} - R\mathbf{h}(n)\big)$$

or,

$$\mathbf{h}(n+1) = (I - 2\mathcal{M}R)\mathbf{h}(n) + 2\mathcal{M}\mathbf{r} \tag{16.6.4}$$

with solution for $n \ge 0$

$$\mathbf{h}(n) = \mathbf{h}_{\text{opt}} + (I - 2\mathcal{M}R)^n\big(\mathbf{h}(0) - \mathbf{h}_{\text{opt}}\big) \tag{16.6.5}$$

where $\mathbf{h}_{\text{opt}} = R^{-1}\mathbf{r}$ is the asymptotic value, and $\mathbf{h}(0)$, the initial value. It is evident from Eq. (16.6.4) or (16.6.5) that the choice of $\mathcal{M}$ can drastically affect the speed of convergence. For example, if $\mathcal{M}$ is chosen as

$$\mathcal{M} = (2R)^{-1} \tag{16.6.6}$$

then $I - 2\mathcal{M}R = 0$, and the convergence occurs in just one step! This choice of $\mathcal{M}$ is equivalent to *Newton's method* of solving the system of equations

$$\mathbf{f}(\mathbf{h}) = \frac{\partial\mathcal{E}}{\partial\mathbf{h}} = 0$$

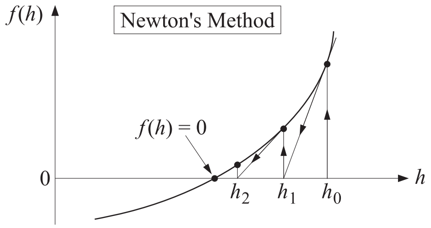
for the optimal weights. Indeed, Newton's method linearizes about each point $\mathbf{h}$ to get the next point, that is, $\Delta\mathbf{h}$ is selected such that

$$\mathbf{f}(\mathbf{h}+\Delta\mathbf{h}) \simeq \mathbf{f}(\mathbf{h}) + \left(\frac{\partial\mathbf{f}}{\partial\mathbf{h}}\right)\Delta\mathbf{h} = 0$$

where we expanded to first order in $\Delta\mathbf{h}$. Solving for $\Delta\mathbf{h}$, we obtain

$$\Delta\mathbf{h} = -\left(\frac{\partial\mathbf{f}}{\partial\mathbf{h}}\right)^{-1}\mathbf{f}(\mathbf{h})$$

But since $\mathbf{f}(\mathbf{h}) = -2(\mathbf{r} - R\mathbf{h})$, we have $\partial\mathbf{f}/\partial\mathbf{h} = 2R$. Therefore, the choice $\mathcal{M} = (2R)^{-1}$ corresponds precisely to Newton's update. Newton's method is depicted below for the one-dimensional case.



Note that the property that Newton's method converges in one step is a well-known property valid for *quadratic* performance indices (in such cases, the gradient $\mathbf{f}(\mathbf{h})$ is already linear in $\mathbf{h}$ and therefore Newton's local linearization is exact). The important property about the choice $\mathcal{M} = (2R)^{-1}$ is that $\mathcal{M}$ is proportional to the inverse of R. An alternative choice could have been $\mathcal{M} = \alpha R^{-1}$. In this case $I - 2\mathcal{M}R$ becomes proportional to the identity matrix:

$$I - 2\mathcal{M}R = (1 - 2\alpha)I$$

having equal eigenvalues. Stability requires that $|1 - 2\alpha| < 1$, or equivalently, $0 < \alpha < 1$, with Newton's choice corresponding exactly to the middle of this interval, $\alpha = 1/2$. Therefore, the disparity between the eigenvalues that could slow down the convergence rate is eliminated, and all eigenmodes converge at the same rate (which is faster the more $\mathcal{M}$ resembles $(2R)^{-1}$).

The implementation of such Newton-like methods requires knowledge of R, which we do not have (if we did, we would simply compute the Wiener solution $\mathbf{h}_{\text{opt}} = R^{-1}\mathbf{r}$.) However, as we shall see later, the so-called *recursive least-squares algorithms* effectively provide an implementation of Newton-type methods, and that is the reason for their extremely fast convergence. Adaptive *lattice* filters also have very fast convergence properties. In that case, because of the orthogonalization of the successive lattice stages of the filter, the matrix R is diagonal (in the decorrelated basis) and the matrix $\mathcal{M}$ can also be chosen to be diagonal so as to equalize and speed up the convergence rate of all the filter coefficients. Recursive least-squares and adaptive lattice filters are discussed in Sections Sec. 16.16 and 16.18, respectively.

Finally, we would like to demonstrate the previous statement that a strongly correlated signal y_n has a large spread in the eigenvalue spectrum of its covariance matrix. For simplicity, consider the 2×2 case

$$R = E\big[\mathbf{y}(n)\mathbf{y}^T(n)\big] = E\left[\begin{bmatrix} y_n \\ y_{n-1}\end{bmatrix}\big[y_n,\ y_{n-1}\big]\right] = \begin{bmatrix} R(0) & R(1) \\ R(1) & R(0)\end{bmatrix}$$

The two eigenvalues are easily found to be

$$\lambda_{\min} = R(0) - |R(1)|$$

$$\lambda_{\max} = R(0) + |R(1)|$$

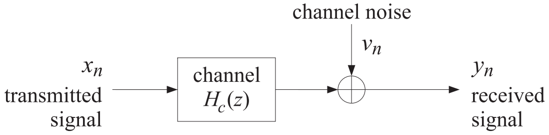
and therefore, the ratio $\lambda_{\min}/\lambda_{\max}$ is given by

$$\frac{\lambda_{\min}}{\lambda_{\max}} = \frac{R(0) - |R(1)|}{R(0) + |R(1)|}$$

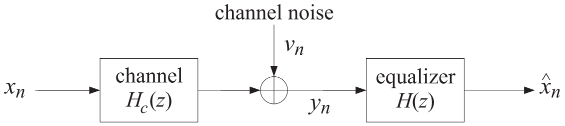
Since for an autocorrelation function we always have $|R(1)| \le R(0)$, it follows that the largest value of $R(1)$ is $\pm R(0)$, implying that for highly correlated signals the ratio $\lambda_{\min}/\lambda_{\max}$ will be very close to zero.

16.7 Adaptive Channel Equalizers

Channels used in digital data transmissions can be modeled very often by linear time-invariant systems. The standard model for such a channel including channel noise is shown here.

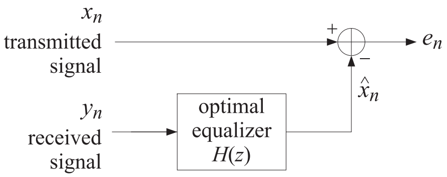


In the Figure, $H_c(z)$ is the transfer function for the channel and v_n, the channel noise, assumed to be additive white gaussian noise. The transfer function $H_c(z)$ incorporates the effects of the modulator and demodulator filters, as well as the channel distortions. The purpose of a channel equalizer is to undo the distorting effects of the channel and recover, from the received waveform y_n, the signal x_n that was transmitted. Typically, a channel equalizer will be an FIR filter with enough taps to approximate the inverse transfer function of the channel. A basic equalizer system is shown below.



In this figure, $H(z)$ is the desired transfer function of the equalizer. In many situations, such in the telephone network, the channel is not known in advance, or it may be time-varying as in the case of multipath channels. Therefore, it is desirable to design equalizers adaptively [1357–1359].

A channel equalizer, adaptive or not, is an optimal filter since it tries to produce as good an estimate $\hat{x}_n$ of the transmitted signal x_n as possible. The Wiener filtering concepts that we developed thus far are ideally suited to this problem. This is shown below.



The design of the optimal filter requires two things: first, the autocorrelation of the received signal y_n, and second, the cross-correlation of the transmitted signal x_n with the received signal. Since the transmitted signal is not available at the receiver, the following procedure is used. After the channel connection is established, a "training" sequence x_n, which is also known to the receiver, is transmitted over the channel. Then, the equalizer may be designed, and then the actual message transmitted. To appreciate the equalizer's action as an inverse filter, suppose that the training sequence x_n is a white-noise sequence of variance σ_x^2. According to the theory developed in Chap. 11, the optimal filter estimating x_n on the basis of y_n is given by

$$H(z)=\frac{1}{\sigma_\epsilon^2 B(z)}\left[\frac{S_{xy}(z)}{B(z^{-1})}\right]_+$$

where $B(z)$ is the spectral factor of $S_{yy}(z)=\sigma_\epsilon^2 B(z)B(z^{-1})$. To simplify the discussion, let us ignore the causal instruction:

$$H(z)=\frac{S_{xy}(z)}{\sigma_\epsilon^2 B(z)B(z^{-1})}=\frac{S_{xy}(z)}{S_{yy}(z)}$$

Since we have $Y(z)=H_c(z)X(z)+V(z)$, we find

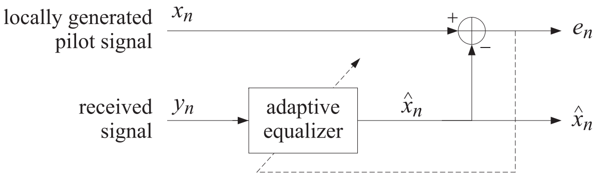
$$\begin{aligned}S_{xy}(z) &= S_{xx}(z)H_c(z^{-1})+S_{xv}(z)=S_{xx}(z)H_c(z^{-1})=\sigma_x^2 H_c(z^{-1})\\ S_{yy}(z) &= H_c(z)H_c(z^{-1})S_{xx}(z)+S_{vv}(z)=\sigma_x^2 H_c(z)H_c(z^{-1})+\sigma_v^2\end{aligned}$$

the equalizer's transfer function is then

$$H(z)=\frac{S_{xy}(z)}{S_{yy}(z)}=\frac{\sigma_x^2 H_c(z^{-1})}{\sigma_x^2 H_c(z)H_c(z^{-1})+\sigma_v^2}$$

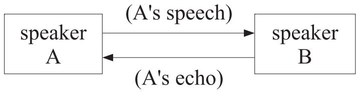
It is seen that when the channel noise is weak (small σ_v^2), the equalizer essentially behaves as the inverse filter $1/H_c(z)$ of the channel.

In an adaptive implementation, we must use a filter with a finite number of weights. These weights are adjusted adaptively until they converge to their optimal values. Again, during this "training mode" a known pilot signal is sent over the channel and is received as y_n. At the receiving end, the pilot signal is locally generated and used in the adaptation algorithm. This implementation is shown below.

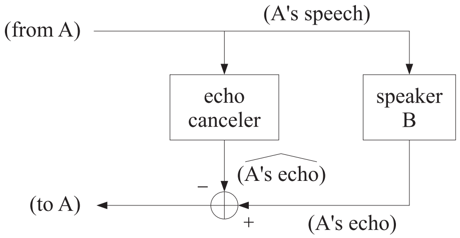


16.8 Adaptive Echo Cancelers

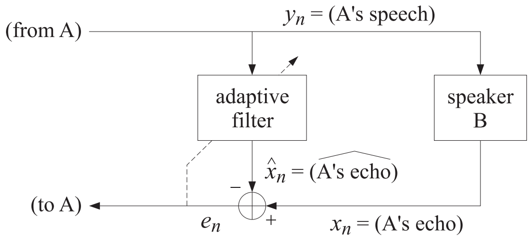
Consider two speakers A and B connected to each other by the telephone network. As a result of various impedance mismatches, when A's speech reaches B, it manages to "leak" through and echoes back to speaker A, as though it were B's speech.



An echo canceler may be placed near B's end, as shown.



It produces an (optimum) estimate of A’s echo through B’s circuits, and then proceeds to cancel it from the signal returning to speaker A. Again, this is another case for which optimal filtering ideas are ideally suited. An adaptive echo canceler is an adaptive FIR filter placed as shown [1360–1365].



As always, the adaptive filter will adjust itself to cancel any correlations that might exist between the secondary signal y_n (A’s speech) and the primary signal x_n (A’s echo).

16.9 Adaptive Noise Canceling

In many applications, two signals are available; one is composed of a desired signal plus undesired noise interference, and the other is composed only of noise interference which, if not identical with the noise part of the first signal, is correlated with it. This is shown in Fig. 16.9.1. An adaptive noise canceler [1350] is an adaptive filter as shown in the Figure. It acts as a correlation canceler. If the signals x_n and y_n are in any way correlated (i.e., the noise component of x_n with y_n), then the filter will respond by adapting its weights until such correlations are canceled from the output e_n. It does so by producing the best possible replica of the noise component of x_n and proceeding to cancel it. The output e_n will now consist mainly of the desired signal.

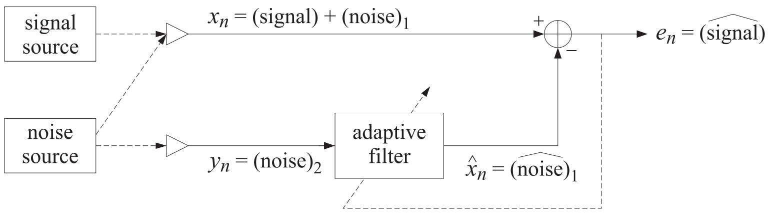

signal source
noise source
x_n = (signal) + (noise)$_1$
+
−
$e_n = \widehat{\text{(signal)}}$
adaptive filter
y_n = (noise)$_2$
$\hat{x}_n = \widehat{\text{(noise)}}_1$

Fig. 16.9.1 Adaptive noise canceler.

There are many applications of adaptive noise canceling, such as adaptive sidelobe cancellation, acoustic noise cancellation [1368–1370], canceling 60 Hz interference in EKG recordings, plasma estimation [1371], and ghost cancellation in television [1372].

An interesting property of the adaptive noise canceler is that when the secondary signal y_n is purely sinusoidal at some frequency ω_0, the adaptive filter behaves as a *notch filter* [1350,1373] at the sinusoid’s frequency, that is, the transfer relationship between the primary input x_n and the output e_n becomes the time-invariant transfer function of a notch filter. This is a surprising property since the adaptation equations for the weights and the filtering I/O equation are in general time-noninvariant. To understand this effect, it proves convenient to work with complex-valued signals using a complex-valued reformulation of the LMS algorithm [1374]. We make a short digression on this, first. We assume that x_n, y_n and the weights $\mathbf{h}(n)$ are complex-valued. The performance index is replaced by

$$\mathcal{E} = E[e_n^* e_n]$$

where the I/O filtering equation is still given by

$$\hat{x}_n = \sum_{m=0}^{M} h_m y_{n-m} = \mathbf{h}^T \mathbf{y}(n)$$

Since the weights $\mathbf{h}$ are complex, the index $\mathcal{E}$ depends on both the real and the imaginary parts of $\mathbf{h}$. Equivalently, we may think of $\mathcal{E}$ as a function of the two independent variables $\mathbf{h}$ and $\mathbf{h}^*$. A complex change in the weights $\Delta\mathbf{h}$ will change the index to

$$\mathcal{E}(\mathbf{h}+\Delta\mathbf{h}, \mathbf{h}^* + \Delta\mathbf{h}^*) = \mathcal{E}(\mathbf{h}, \mathbf{h}^*) + \Delta\mathbf{h}^T \frac{\partial\mathcal{E}}{\partial\mathbf{h}} + \Delta\mathbf{h}^\dagger \frac{\partial\mathcal{E}}{\partial\mathbf{h}^*}$$

Choosing $\Delta\mathbf{h}$ to be proportional to the complex conjugate of the negative gradient, that is,

$$\Delta\mathbf{h} = -2\mu\frac{\partial\mathcal{E}}{\partial\mathbf{h}^*} = 2\mu E[e_n \mathbf{y}(n)^*]$$

will move the index $\mathcal{E}$ towards its minimum value; indeed,

$$\mathcal{E}(\mathbf{h}+\Delta\mathbf{h}, \mathbf{h}^* + \Delta\mathbf{h}^*) = \mathcal{E}(\mathbf{h}, \mathbf{h}^*) - 4\mu\left(\frac{\partial\mathcal{E}}{\partial\mathbf{h}}\right)^\dagger\left(\frac{\partial\mathcal{E}}{\partial\mathbf{h}}\right) \le \mathcal{E}(\mathbf{h}, \mathbf{h}^*)$$

Thus, the complex version of the LMS algorithm consists simply of replacing the instantaneous gradient by its complex conjugate [1374]. We summarize the algorithm as follows:

1. Compute $\hat{x}_n = \mathbf{h}(n)^T \mathbf{y}(n)$.
2. Compute $e_n = x_n - \hat{x}_n$.
3. Update weights $\mathbf{h}(n+1) = \mathbf{h}(n) + 2\mu e_n \mathbf{y}(n)^*$.

Using this complex version, we now discuss the notching behavior of the adaptive filter. Suppose y_n is sinusoidal

$$y_n = A e^{j\omega_0 n}$$

at some frequency ω_0. Then, the weight-update equation becomes:

$$h_m(n+1) = h_m(n) + 2\mu e_n y_{n-m}^* = h_m(n) + 2\mu A^* e^{-j\omega_0(n-m)}$$

for $m = 0, 1, \dots, M$. The factor $e^{-j\omega_0(n-m)}$ suggests that we look for a solution of the form

$$h_m(n) = f_m(n) e^{-j\omega_0(n-m)}$$

Then, $f_m(n)$ must satisfy the difference equation

$$e^{-j\omega_0} f_m(n+1) = f_m(n) + 2\mu A^* e_n$$

As a difference equation in n, this equation has constant coefficients, and, therefore, may be solved by z-transform techniques. Taking z-transforms of both sides we find

$$e^{-j\omega_0} z F_m(z) = F_m(z) + 2\mu A^* E(z)$$

which may be solved for $F_m(z)$ in terms of $E(z)$ to give

$$F_m(z) = E(z)\,\frac{2\mu A^* e^{j\omega_0}}{z - e^{j\omega_0}}$$

On the other hand, the I/O filtering equation from y_n to the output $\hat{x}_n$ is

$$\hat{x}_n = \sum_{m=0}^{M} h_m(n)y_{n-m} = \sum_{m=0}^{M} f_m(n)e^{-j\omega_0(n-m)}Ae^{j\omega_0(n-m)} = \sum_{m=0}^{M} f_m(n)A$$

or, in the z-domain

$$\hat{X}(z) = \sum_{m=0}^{M} F_m(z)A = E(z)\frac{2\mu(M+1)|A|^2e^{j\omega_0}}{z-e^{j\omega_0}}$$

Finally, the I/O equation from x_n to e_n becomes

$$e_n = x_n - \hat{x}_n$$

and, in the z-domain

$$E(z) = X(z) - \hat{X}(z) = X(z) - E(z)\frac{2\mu(M+1)|A|^2e^{j\omega_0}}{z-e^{j\omega_0}}$$

which may be solved for the transfer function $E(z)/X(z)$

$$\frac{E(z)}{X(z)} = \frac{z-e^{j\omega_0}}{z-Re^{j\omega_0}}\,, \quad R \equiv 1 - 2\mu(M+1)|A|^2 \tag{16.9.1}$$

This filter has a zero at $z = e^{j\omega_0}$ which corresponds to the notch at the frequency ω_0. For sufficiently small values of μ and A, the filter is stable; its pole is at $z = Re^{j\omega_0}$ and can be made to lie inside the unit circle $(0 < R < 1)$. If the primary input x_n happens to have a sinusoidal component at frequency ω_0, this component will be completely notched away from the output. This will take place even when the sinusoidal reference signal is very weak (i.e., when A is small). The implications of this property for *jamming by signal cancellation* in adaptive array processing have been discussed in [1375]. The notching behavior of the adaptive noise canceler when the reference signal consists of a sinusoid plus noise has been discussed in [1376].

A related result is that the adaptive noise canceler behaves as a time-invariant *comb filter* whenever its secondary input y_n is a periodic train of impulses separated by some period [1377]. This property can be used to cancel periodic interference. Because the method of signal averaging can be thought of as comb filtering, the above property may also be used as an alternative method to perform signal averaging for pulling weak periodic signals from background noise, such as evoked potentials [1378].

16.10 Adaptive Line Enhancer

A special case of adaptive noise canceling is when there is only one signal x_n available which is contaminated by noise. In such a case, the signal x_n provides its own reference signal y_n, which is taken to be a delayed replica of x_n, that is, $y_n = x_{n-\Delta}$, as shown in Fig. 16.10.1, referred to as the adaptive line enhancer (ALE) [1350,1379–1381].

Will such arrangement be successful? The adaptive filter will respond by canceling any components of the main signal x_n that are in any way correlated with the secondary signal $y_n = x_{n-\Delta}$. Suppose the signal x_n consists of two parts: a narrowband component that has long-range correlations such as a sinusoid, and a broadband component which will tend to have short-range correlations. One of these could represent the desired signal and the other an undesired interfering noise. Pictorially the autocorrelations of the two components could look as follows.

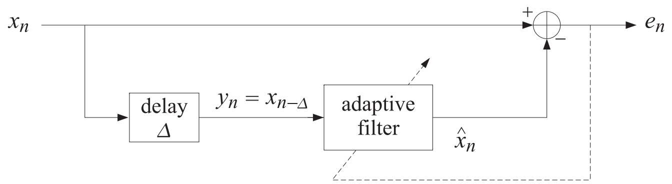
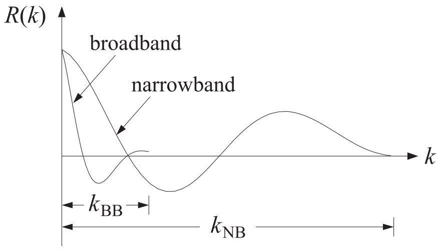


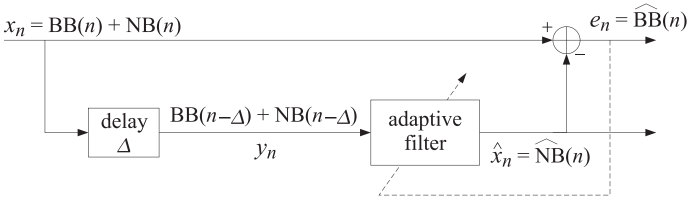
Fig. 16.10.1 Adaptive line enhancer.



where k_{NB} and k_{BB} are effectively the self-correlation lengths of the narrowband and broadband components, respectively. Beyond these lags, the respective correlations die out quickly. Suppose the delay Δ is selected so that

$$k_{\text{BB}} \le \Delta \le k_{\text{NB}}$$

Since Δ is longer than the effective correlation length of the BB component, the delayed replica $\text{BB}(n-\Delta)$ will be entirely uncorrelated with the BB part of the main signal. The adaptive filter will not be able to respond to this component. On the other hand, since Δ is shorter than the correlation length of the NB component, the delayed replica $\text{NB}(n-\Delta)$ that appears in the secondary input will still be correlated with the NB part of the main signal, and the filter will respond to cancel it. Thus, the filter outputs will be as shown.


$x_n = \text{BB}(n) + \text{NB}(n)$
+
−
$e_n = \widehat{\text{BB}}(n)$
delay
Δ
$\text{BB}(n-\Delta) + \text{NB}(n-\Delta)$
y_n
adaptive
filter
$\hat{x}_n = \widehat{\text{NB}}(n)$

Note that if Δ is selected to be longer than both correlation lengths, the secondary input will become uncorrelated with the primary input, and the adaptive filter will turn itself off. In the opposite case, when the delay Δ is selected to be less than both correlation lengths, then both components of the secondary signal will be correlated with the primary signal, and therefore, the adaptive filter will respond to cancel the primary x_n completely. The computational algorithm for the ALE is as follows

1. $\hat{x}_n = \sum_{m=0}^{M} h_m(n)y(n-m) = \sum_{m=0}^{M} h_m(n)x(n-m-\Delta)$

2. $e_n = x_n - \hat{x}_n$
3. $h_m(n+1) = h_m(n) + 2\mu e_n x(n-m-\Delta)\,, \quad m = 0, 1, \dots, M$

The Wiener solution for the steady-state weights is $\mathbf{h} = R^{-1}\mathbf{r}$, where R and $\mathbf{r}$ are both expressible in terms of the autocorrelation of the signal x_n, as follows:

$$R_{ij} = E[y_{n-i}y_{n-j}] = E[x_{n-\Delta-i}\,x_{n-\Delta-j}] = R_{xx}(i-j)$$

$$r_i = E[x_n y_{n-i}] = E[x_n x_{n-\Delta-i}] = R_{xx}(i+\Delta)$$

for $i, j = 0, 1, \dots, M$. When the input signal consists of multiple sinusoids in additive white noise, the inverse R^{-1} may be obtained using the methods of Sec. 14.2, thus resulting in a closed form expression for the steady-state optimal weights [1381].

16.11 Adaptive Linear Prediction

A linear predictor is a special case of the ALE with the delay $\Delta = 1$. It is shown in Fig. 16.11.1, where to be consistent with our past notation on linear predictors we have denoted the main signal by y_n. The secondary signal, the input to the adaptive filter, is then y_{n-1}. Due to the special sign convention used for linear predictors, the adaptation algorithm now reads

1. $\hat{y}_n = -[a_1(n)y_{n-1} + a_2(n)y_{n-2} + \cdots + a_M(n)y_{n-M}]$
2. $e_n = y_n - \hat{y}_n = y_n + a_1(n)y_{n-1} + \cdots + a_M(n)y_{n-M}$
3. $a_m(n+1) = a_m(n) - 2\mu e_n y_{n-m}\,, \quad m = 1, 2 \dots, M$

The realization of Fig. 16.11.1 can be redrawn more explicitly as in Fig. 16.11.2. The **lmsap** is an implementation of the LMS adaptive predictor. At each call, the function reads a sample y_n, computes the filter output e_n, updates the filter coefficients $a_m(n)$ to their new values $a_m(n+1)$ to be used by the next call, and updates the registers of the tapped delay line. With a small modification it can be used in the adaptive array problem (see below).

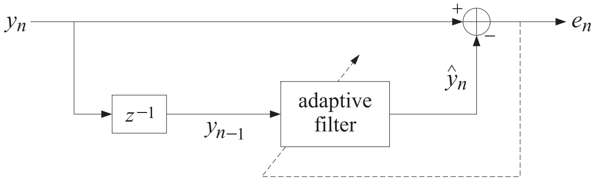


Fig. 16.11.1 Adaptive linear predictor.

Because of the importance of the adaptive predictor, we present a direct derivation of the LMS algorithm as it applies to this case. The weights a_m are chosen optimally to minimize the mean output power of the filter, that is, the mean-square prediction error:

$$\mathcal{E} = E[e_n^2] = \mathbf{a}^T R\mathbf{a} = \min \tag{16.11.1}$$

where $\mathbf{a} = [1, a_1, a_2, \dots, a_M]^T$ is the prediction error filter. The performance index (16.11.1) is minimized with respect to the M weights a_m. The gradient with respect to a_m is the mth component of the vector $2R\mathbf{a}$, namely,

$$\frac{\partial\mathcal{E}}{\partial a_m} = 2(R\mathbf{a})_m = 2\big(E[\mathbf{y}(n)\mathbf{y}(n)^T]\mathbf{a}\big)_m = 2\big(E[\mathbf{y}(n)\mathbf{y}(n)^T\mathbf{a}]\big)_m$$
$$= 2\big(E[\mathbf{y}(n)e_n]\big)_m = 2E[e_n y_{n-m}]$$

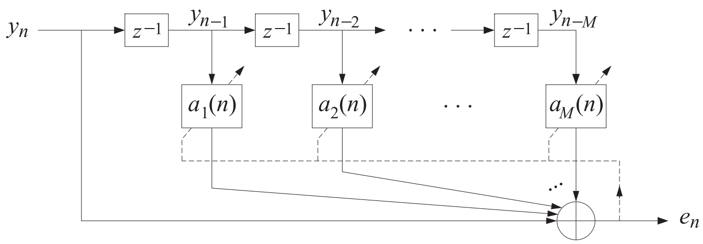


Fig. 16.11.2 Direct-form realization of adaptive predictor.

The instantaneous gradient is obtained by *ignoring* the expectation instruction. This gives for the LMS time-update of the mth weight

$$\Delta a_m(n) = -\mu\frac{\partial\mathcal{E}}{\partial a_m} = -2\mu e_n y_{n-m}\,, \quad m = 1, 2, \dots, M \tag{16.11.2}$$

The adaptive predictor may be thought of as an *adaptive whitening filter*, or an analysis filter which determines the LPC model parameters adaptively. As processing of the signal y_n takes place, the autoregressive model parameters a_m are extracted *on-line*. This is but one example of on-line system identification methods [1384–1392].

The extracted model parameters may be used in any desired way—for example, to provide the *autoregressive spectrum estimate* of the signal y_n. One of the advantages of the adaptive implementation is that it offers the possibility of tracking slow changes in the spectra of non-stationary signals. The only requirement for obtaining meaningful spectrum estimates is that the non-stationary changes of the spectrum be slow enough for the adaptive filter to have a chance to converge between changes. Typical applications are the tracking of sinusoids in noise whose frequencies may be slowly changing [1382,1383,1393], or tracking the time development of the spectra of non-stationary EEG signals [1028]. At each time instant n, the adaptive weights $a_m(n)$, $m = 1, 2, \dots, M$ may be used to obtain an instantaneous autoregressive estimate of the spectrum of y_n in the form

$$S_n(\omega) = \frac{1}{\left|1 + a_1(n)e^{-j\omega} + a_2(n)e^{-2j\omega} + \cdots + a_M(n)e^{-Mj\omega}\right|^2}$$

This is the adaptive implementation of the LP spectrum estimate discussed in Sec. 14.2. The same adaptive approach to LP spectrum estimation may also be used in the problem of multiple source location, discussed in Sec. 14.3. The only difference in the algorithm is to replace y_{n-m} by $y_m(n)$—that is, by the signal recorded at the mth sensor at time n—and to use the complex-valued version of the LMS algorithm. For completeness, we summarize the computational steps in this case, following the notation of Sec. 14.3.

1. $e(n) = y_0(n) + a_1(n)y_1(n) + a_2(n)y_2(n) + \cdots + a_M(n)y_M(n)$
2. $a_m(n+1) = a_m(n) - 2\mu e(n)y_m^*(n)\,, \quad m = 1, 2, \dots, M$

At each time instant n, the corresponding spatial spectrum estimate may be computed by

$$S_n(k) = \frac{1}{\left|1 + a_1(n)e^{-jk} + a_2(n)e^{-2jk} + \cdots + a_M(n)e^{-Mjk}\right|^2}$$

where the wavenumber k and its relationship to the angle of bearing was defined in Sec. 14.3. Fig. 16.11.3 shows the corresponding adaptive array processing configuration.

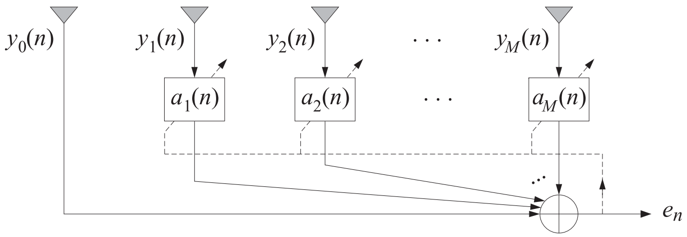


Fig. 16.11.3 Adaptive array processor.

The time-adaptive as well as the block-data adaptive methods of superresolution array processing have been reviewed in [1099,1129]. The above LMS algorithm for the array weights is effectively equivalent to the Howells-Applebaum algorithm [1351–1355]. Adaptive predictors may also be used to improve the performance of spread-spectrum systems [1396–1402].

16.12 Adaptive Implementation of Pisarenko's Method

In Sec. 14.2, we noted that the Pisarenko eigenvalue problem was equivalent to the minimization of the performance index

$$\mathcal{E} = E[e_n^* e_n] = \mathbf{a}^\dagger R\mathbf{a} = \min \tag{16.12.1}$$

subject to the quadratic constraint

$$\mathbf{a}^\dagger\mathbf{a} = 1 \tag{16.12.2}$$

where

$$e_n = \sum_{m=0}^{M} a_m y_{n-m} = [a_0, a_1, \dots, a_M]\begin{bmatrix} y_n \\ y_{n-1} \\ \vdots \\ y_{n-M}\end{bmatrix} = \mathbf{a}^T\mathbf{y}(n)$$

The solution of the minimization problem shown in Eqs. (16.12.1) and (16.12.2) is the eigenvector **a** belonging to the minimum eigenvalue of the covariance matrix R. If there are L sinusoids of frequencies ω_i, $i = 1, 2, \dots, L$, and we use a filter of order M, such that $M \ge L$, then the eigenpolynomial $A(z)$ corresponding to the minimum eigenvector **a** will have L zeros on the unit circle at precisely the desired set of frequencies, that is,

$$A(z_i) = 0\,, \quad \text{where} \quad z_i = e^{j\omega_i}\,, \quad i = 1, 2, \dots, L$$

The adaptive implementation [1403] of the Pisarenko eigenvalue problem is based on the above minimization criterion. The LMS gradient-descent algorithm can be used to update the weights, but some care must be taken to satisfy the essential quadratic constraint (16.12.2) at each iteration of the algorithm. Any infinitesimal change $d\mathbf{a}$ of the weights must respect the constraint. This means the $d\mathbf{a}$ cannot be arbitrary but must satisfy the condition

$$d(\mathbf{a}^\dagger\mathbf{a}) = \mathbf{a}^\dagger(d\mathbf{a}) + (d\mathbf{a})^\dagger\mathbf{a} = 0 \tag{16.12.3}$$

so that the new weight $\mathbf{a} + d\mathbf{a}$ still lies on the quadratic surface $\mathbf{a}^\dagger\mathbf{a} = 1$. The ordinary gradient of the performance index $\mathcal{E}$ is

$$\frac{\partial\mathcal{E}}{\partial\mathbf{a}^*} = R\mathbf{a}$$

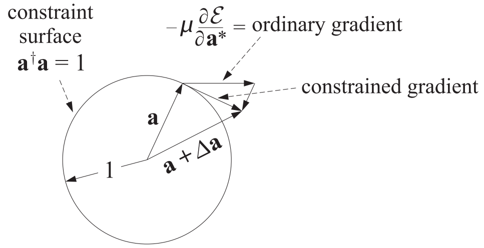
Projecting this onto the surface $\mathbf{a}^\dagger\mathbf{a} = 1$ by the projection matrix $\mathcal{P} = I - \mathbf{a}\mathbf{a}^\dagger$, where I is the $(M+1)$-dimensional unit matrix, we obtain the "constrained" gradient

$$\left(\frac{\partial\mathcal{E}}{\partial\mathbf{a}^*}\right)_c = \mathcal{P}\,\frac{\partial\mathcal{E}}{\partial\mathbf{a}^*} = (I - \mathbf{a}\mathbf{a}^\dagger)(R\mathbf{a}) = R\mathbf{a} - \mathcal{E}\mathbf{a} \tag{16.12.4}$$

which is *tangent* to the constraint surface at the point **a**. The vanishing of the constrained gradient is equivalent to the Pisarenko eigenvalue problem. The weight update can now be chosen to be proportional to the constrained gradient

$$\Delta\mathbf{a} = -\mu\left(\frac{\partial\mathcal{E}}{\partial\mathbf{a}^*}\right)_c = -\mu(R\mathbf{a} - \mathcal{E}\mathbf{a})$$

The projection of the gradient onto the constraint surface is shown below.


constraint
surface
$\mathbf{a}^\dagger\mathbf{a} = 1$
$-\mu\dfrac{\partial\mathcal{E}}{\partial\mathbf{a}^*}$ = ordinary gradient
constrained gradient
$\mathbf{a}$
$\mathbf{a}+\Delta\mathbf{a}$
1

This choice guarantees that $\Delta\mathbf{a}$ satisfies Eq. (16.12.3); indeed, because of the projection matrix in front of the gradient, it follows that $\mathbf{a}^\dagger\Delta\mathbf{a} = 0$. Actually, since $\Delta\mathbf{a}$ is not infinitesimal, it will correspond to a *finite* motion along the tangent to the surface at the point **a**. Thus, the new point $\mathbf{a} + \Delta\mathbf{a}$ will be slightly off the surface and must be renormalized to have unit norm. Using the properties,

$$R\,\mathbf{a} = E[\mathbf{y}(n)^*\mathbf{y}(n)^T]\mathbf{a} = E[\mathbf{y}(n)^* e_n] \quad \text{and} \quad \mathcal{E} = E[e_n^* e_n]$$

we write the update as

$$\Delta\mathbf{a} = -\mu\big(E[e_n\mathbf{y}(n)^*] - E[e_n^* e_n]\mathbf{a}\big)$$

The LMS algorithm is obtained by ignoring the indicated ensemble expectation values. The weight adjustment procedure consists of two steps: first, shift the old weight $\mathbf{a}(n)$ by $\Delta\mathbf{a}(n)$, and then renormalize it to unit norm:

$$\mathbf{a}(n+1) = \frac{\mathbf{a}(n) + \Delta\mathbf{a}(n)}{\|\mathbf{a}(n) + \Delta\mathbf{a}(n)\|} \tag{16.12.5}$$

where the weight update is computed by

$$\Delta\mathbf{a}(n) = -\mu\big[e_n\mathbf{y}(n)^* - e_n^* e_n\mathbf{a}(n)\big] \tag{16.12.6}$$

In summary, the computational steps are as follows:

1. At time n, $\mathbf{a}(n)$ is available and normalized to unit norm.
2. Compute the output $e_n = \sum_{m=0}^{M} a_m(n) y_{n-m} = \mathbf{a}(n)^T\mathbf{y}(n)$.
3. Update the filter weights using Eq. (16.12.5) and (16.12.6).
4. Go to the next time instant, $n \to n+1$.

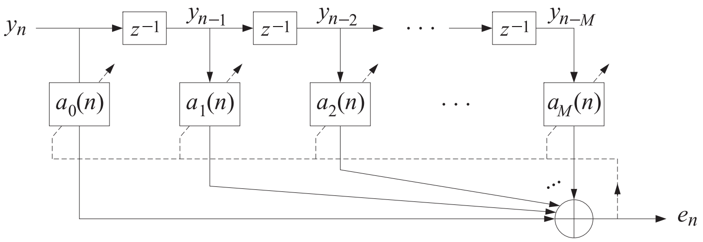
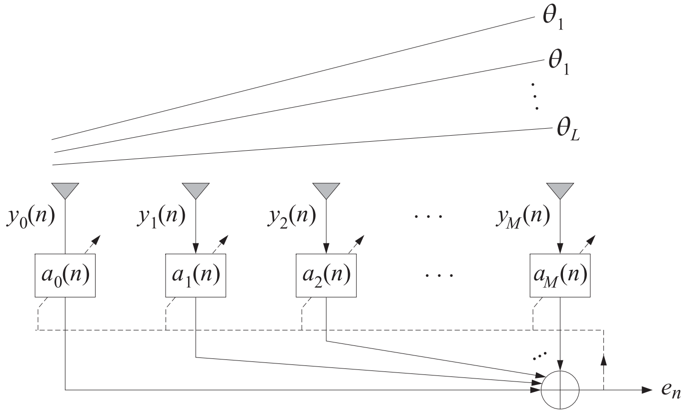


Fig. 16.12.1 Adaptive implementation of Pisarenko's method.

A realization of the adaptive filter is shown in Fig. 16.12.1. After a number of iterations, the algorithm may be stopped and the Pisarenko spectrum estimate computed:

$$S_n(\omega)=\frac{1}{\left|a_0(n)+a_1(n)e^{-j\omega}+a_2(n)e^{-2j\omega}+\cdots+a_M(n)e^{-Mj\omega}\right|^2}$$

After convergence, $S_n(\omega)$ should exhibit very sharp peaks at the sought frequencies ω_i, $i = 1,2\dots,L$. The convergence properties of this algorithm have been studied in [1404]. Alternative adaptation schemes for the weights have been proposed in [1406]. The algorithm may also be applied to the array problem of *multiple source location* [1407]. Again, the only change is to replace y_{n-m} by $y_m(n)$, depicted below.



Both the adaptive prediction and the Pisarenko approaches to the two problems of extracting sinusoids in noise and multiple emitter location have a common aim, namely, to produce an adaptive filter $A(z)$ with zeros very near or on the unit circle at the desired frequency angles. Taking the inverse magnitude response as an estimate of the spectrum of the signal,

$$S(\omega)=\frac{1}{|A(\omega)|^2}$$

is a simple device to obtain a curve that exhibits sharp spectral peaks at the desired frequencies.

A satisfactory alternative approach would be simply to find the roots of the polynomial $A(z)$ and pick those that are closest to the unit circle. The phase angles of these roots are precisely the desired frequencies. In other words, the frequency information we are attempting to extract by means of the adaptive filter is more directly represented by the zeros of the filter than by its weights.

It would be desirable then to develop methods by which these zeros can be estimated directly without having to submit the filter $A(z)$ to root-finding algorithms. In implementing this idea adaptively, we would like to adapt and track the zeros of the adaptive filter as they move about on the complex z-plane, converging to their final destinations which are the desired zeros. In this way, the frequency information can be extracted directly. Such "zero-tracking" adaptation algorithms have been proposed recently [1408,1409].

Even though the representations of the filter in terms of its zeros and in terms of its weights are mathematically equivalent, the zero representation may be more appropriate in some applications in the sense that a better insight into the nature of the underlying processes may be gained from it than from the weight representation.

As an example, we mention the problem of predicting epileptic seizures by LPC modeling of the EEG signal where it was found [1410] that the trajectories of the zeros of the prediction-error filter on the z-plane exhibited an unexpected behavior, namely, prior to the onset of a seizure, one of the zeros became the "most mobile" and moved towards the unit circle, whereas the other zeros did not move much. The trajectory of the most mobile zero could be used as a signature for the onset of the oncoming seizure. Such behavior could not be easily discerned by the frequency response or by the final zero locations.

Next, we describe briefly the *zero-tracking algorithm* as it applies to the Pisarenko problem and present a simulation example. Its application to adaptive prediction and to emitter location has been discussed in [1409]. For simplicity, we assume that the number of sinusoids that are present is the same as the order of the filter $\mathbf{a}$, that is, $L = M$. The case $L < M$ will be discussed later on. The eigenpolynomial of the minimum eigenvector $\mathbf{a}$ may be factored into its zeros as follows:

$$\begin{aligned}A(z)&=a_0+a_1z^{-1}+a_2z^{-2}+\cdots+a_Mz^{-M}\\&=a_0(1-z_1z^{-1})(1-z_2z^{-1})\cdots(1-z_Mz^{-1})\end{aligned}\tag{16.12.7}$$

where a_0 may be thought of as a normalization factor which guarantees the unit norm constraint (16.12.2), and $z_i = e^{j\omega_i}$, $i = 1,2,\dots,M$ are the desired sinusoid zeros on the unit circle.

In the adaptive implementation, the weights a_m become time-dependent $a_m(n)$ and are adapted from each time instant to the next until they converge to the asymptotic values defined by Eq. (16.12.7). At each n, the corresponding polynomial can be factored into its zeros as follows:

$$\begin{aligned}&a_0(n)+a_1(n)z^{-1}+a_2(n)z^{-2}+\cdots+a_M(n)z^{-M}\\&=a_0(n)\bigl(1-z_1(n)z^{-1}\bigr)\bigl(1-z_2(n)z^{-1}\bigr)\cdots\bigl(1-z_M(n)z^{-1}\bigr)\end{aligned}\tag{16.12.8}$$

where again the factor $a_0(n)$ ensures the unit-norm constraint. In the zero-tracking algorithm, the weight update equation (16.12.5) is replaced by a zero-update equation of the form:

$$z_i(n+1)=z_i(n)+\Delta z_i(n)\,,\quad i=1,2,\dots,M\tag{16.12.9}$$

where the zero updates $\Delta z_i(n)$ must be such that to ensure the convergence of the zeros to their asymptotic values z_i. One way to do this is to make the algorithm equivalent to the LMS algorithm. The functional dependence of $z_i(n)$ on $a_m(n)$ defined by Eq. (16.12.8) implies that if the weights $a_m(n)$ are changed by a small amount $\Delta a_m(n)$ given by Eq. (16.12.6), then a small change $\Delta z_i(n)$ will be induced on the corresponding zeros. This is given as follows:

$$\Delta z_i(n)=\sum_{m=0}^{M}\frac{\partial z_i(n)}{\partial a_m}\Delta a_m(n)\tag{16.12.10}$$

where the partial derivatives are given by [12]

$$\frac{\partial z_i(n)}{\partial a_m} = -\frac{1}{a_0(n)}\,\frac{z_i(n)^{M-m}}{\prod\limits_{j\neq i}\big(z_i(n)-z_j(n)\big)}\,, \quad 0 \le m \le M \tag{16.12.11}$$

Equation (16.12.10) is strictly valid for infinitesimal changes, but for small μ, it can be taken to be an adequate approximation for the purpose of computing $\Delta z_i(n)$. The advantage of this expression is that only the current zeros $z_i(n)$ are needed to compute $\Delta z_i(n)$. The complete algorithm is summarized as follows:

1. At time n, the zeros $z_i(n)$, $i = 1, 2, \dots, M$ are available.
2. Using convolution, compute the corresponding filter weights and normalize them to unit norm, that is, first convolve the factors of Eq. (16.12.8) to obtain the vector

   $$\begin{aligned}\mathbf{b}(n)^T &= [1, b_1(n), b_2(n), \dots, b_M(n)]\\ &= [1, -z_1(n)] * [1, -z_2(n)] * \cdots * [1, -z_M(n)]\end{aligned}$$

   and then normalize $\mathbf{b}(n)$ to unit norm:

   $$\mathbf{a}(n) = \frac{\mathbf{b}(n)}{\|\mathbf{b}(n)\|}$$

3. Compute the filter output $e_n = \mathbf{a}(n)^T\mathbf{y}(n)$.
4. Compute the LMS coefficient updates $\Delta a_m(n)$ using Eq. (16.12.6). Compute the zero updates $\Delta z_i(n)$ using Eqs. (16.12.10) and (16.12.11), and update the zeros using Eq. (16.12.9).

The algorithm may be initialized by a random selection of the initial zeros inside the unit circle in the z-plane. Next, we present a simulation example consisting of a fourth order filter and four sinusoids

$$y_n = v_n + e^{j\omega_1 n} + e^{j\omega_2 n} + e^{j\omega_3 n} + e^{j\omega_4 n}$$

with frequencies

$$\omega_1 = 0.25\pi\,, \quad \omega_2 = -0.25\pi\,, \quad \omega_3 = 0.75\pi\,, \quad \omega_4 = -0.75\pi$$

and a zero-mean, unit-variance, white noise sequence v_n (this corresponds to all sinusoids having 0 dB signal to noise ratio). The value of μ was 0.001. Figure 7.14 shows the adaptive trajectories of the four filter zeros as they converge onto the unit circle at the above frequency values. After convergence, the adaptive zeros remain within small neighborhoods about the asymptotic zeros. The diameter of these neighborhoods decreases with smaller μ, but so does the speed of convergence [1409].

The transient behavior of the zeros can be seen by plotting $z_i(n)$ versus iteration number n. Fig. 16.12.3 shows the real and imaginary parts of the adaptive trajectory of the zero $z_2(n)$ converging to the real and imaginary parts of the asymptotic zero $z_2 = e^{j\omega_2} = e^{-j0.25\pi} = (1-j)/\sqrt{2}$.

When the number L of sinusoids is less than the order M of the filter, only L of the M zeros $z_i(n)$ of the filter will be driven to the unit circle at the right frequency angles. The remaining $(M-L)$ zeros correspond to spurious degrees of freedom (the degeneracy of the minimum eigenvalue σ_v^2), and are affected by the adaptation process only insofar as the M zero trajectories are not entirely independent of each other but are mutually coupled through Eq. (16.12.11). Where these spurious zeros converge to depends on the particular initialization. For some special initial conditions it is possible for the spurious zeros to move close to the unit circle, thus causing a confusion as to which are the true sinusoid zeros. To safeguard against such a possibility, the algorithm may be run again with a different choice of initial zeros. Fig. 16.12.4 shows the adaptive trajectory of a single sinusoid, $L = 1$, using a third order filter, $M = 3$. The sinusoid's frequency was $\omega_1 = 0.25\pi$, its SNR was 0 dB, and μ was 0.001. One of the three filter zeros is driven to the unit circle at the desired angle ω_1, while the two spurious zeros traverse fairly short paths which depend on their initial positions.

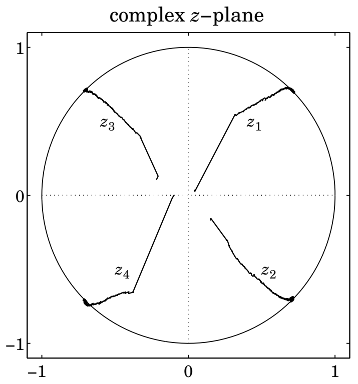


Fig. 16.12.2 z-Plane trajectories of the four adaptive zeros $z_i(n)$, $i = 1, 2, 3, 4$.

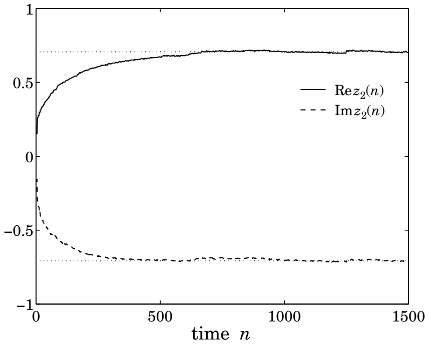


Fig. 16.12.3 Real and imaginary parts of $z_2(n)$ versus n.

16.13 Gradient Adaptive Lattice Filters

In this section we discuss the "gradient adaptive lattice" implementations of linear prediction and lattice Wiener filters [1411–1416]. They are based on a gradient-descent, LMS-like approach applied to the weights of the lattice representations rather than to the weights of the direct-form realization. Taking advantage of the decoupling of the successive stages of the lattice, and properly choosing the adaptation constants μ, all lattice weights can be made to converge fast and, in contrast to the LMS weights, with a convergence rate that is essentially *independent* of the eigenvalue spread of the input covariance matrix. The gradient lattice algorithms are very similar but not identical to the recursive least-squares lattice algorithms (RLSL) [1436–1444], and

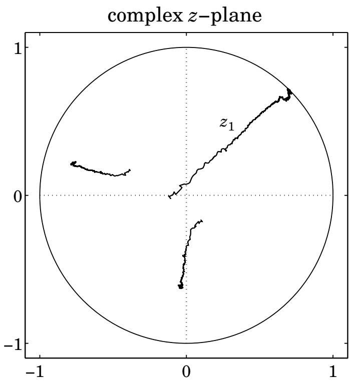


Fig. 16.12.4 Single sinusoid with order-3 adaptive filter.

they share the same properties of fast convergence and computational efficiency with the latter. Typically, the gradient lattice converges somewhat more slowly than RLSL. Some comparisons between the two types of algorithms are given in [1416,1443].

We start by casting the ordinary lattice filter of linear prediction in a gradient-adaptive form, and then discuss the gradient-adaptive form of the lattice Wiener filter, the stationary version of which was presented in Sec. 12.11.

The lattice recursion for an Mth order prediction-error filter of a stationary signal y_n was found in Sec. 12.7 to be

$$\begin{aligned} e_{p+1}^+(n) &= e_p^+(n)-\gamma_{p+1}e_p^-(n-1) \\ e_{p+1}^-(n) &= e_p^-(n-1)-\gamma_{p+1}e_p^+(n) \end{aligned} \tag{16.13.1}$$

for $p = 0,1,\dots,M-1$, and where $e_0^{\pm}(n)= y_n$. The optimal value of the reflection coefficient γ_{p+1} can be obtained by minimizing the performance index

$$\mathcal{E}_{p+1} = E[e_{p+1}^+(n)^2+e_{p+1}^-(n)^2] \tag{16.13.2}$$

Differentiating with respect to γ_{p+1}, we find

$$\frac{\partial\mathcal{E}_{p+1}}{\partial\gamma_{p+1}} = E\left[e_{p+1}^+(n)\frac{\partial e_{p+1}^+(n)}{\partial\gamma_{p+1}} + e_{p+1}^-(n)\frac{\partial e_{p+1}^-(n)}{\partial\gamma_{p+1}}\right]$$

and using Eq. (16.13.1)

$$\frac{\partial\mathcal{E}_{p+1}}{\partial\gamma_{p+1}} = -2E\big[e_{p+1}^+(n)e_p^-(n-1)+e_{p+1}^-(n)e_p^+(n)\big] \tag{16.13.3}$$

Inserting Eq. (16.13.1) into (16.13.3), we rewrite the latter as

$$\frac{\partial\mathcal{E}_{p+1}}{\partial\gamma_{p+1}} = -2(C_{p+1}-\gamma_{p+1}D_{p+1}) \tag{16.13.4}$$

where

$$C_{p+1} = 2E\big[e_p^+(n)e_p^-(n-1)\big] \tag{16.13.5}$$

$$D_{p+1} = E\big[e_p^+(n)^2+e_p^-(n-1)^2\big] \tag{16.13.6}$$

Setting the gradient (16.13.4) to zero, we find the optimal value of γ_{p+1}

$$\gamma_{p+1} = \frac{C_{p+1}}{D_{p+1}} = \frac{2E\big[e_p^+(n)e_p^-(n-1)\big]}{E\big[e_p^+(n)^2+e_p^-(n-1)^2\big]} \tag{16.13.7}$$

which, due to the assumed stationarity, agrees with Eq. (12.7.3). Replacing the numerator and denominator of Eq. (16.13.7) by time averages leads to Burg's method.

The gradient adaptive lattice is obtained by solving $\partial\mathcal{E}_{p+1}/\partial\gamma_{p+1} = 0$ iteratively by the gradient-descent method

$$\gamma_{p+1}(n+1)= \gamma_{p+1}(n)-\mu_{p+1}\frac{\partial\mathcal{E}_{p+1}}{\partial\gamma_{p+1}(n)} \tag{16.13.8}$$

where μ_{p+1} is a small positive adaptation constant. Before we drop the expectation instructions in Eq. (16.13.3), we use the result of Eq. (16.13.4) to discuss qualitatively the convergence rate of the algorithm. Inserting Eq. (16.13.4) into (16.13.8), we find

$$\gamma_{p+1}(n+1)= \gamma_{p+1}(n)+2\mu_{p+1}(C_{p+1}-\gamma_{p+1}(n)D_{p+1})$$

or,

$$\gamma_{p+1}(n+1)= (1-2\mu_{p+1}D_{p+1})\gamma_{p+1}(n)+2\mu_{p+1}C_{p+1} \tag{16.13.9}$$

Actually, if we replace γ_{p+1} by $\gamma_{p+1}(n)$ in Eq. (16.13.1), the stationarity of the lattice is lost, and it is not correct to assume that C_{p+1} and D_{p+1} are independent of n. The implicit dependence of C_{p+1} and D_{p+1} on the (time-varying) reflection coefficients of the previous lattice stages makes Eq. (16.13.9) a nonlinear difference equation in the reflection coefficients. In the analogous discussion of the LMS case in Sec. 16.6, the corresponding difference equation for the weights was linear with constant coefficients. Because of the tapped delay-line structure, the stationarity of the input signal $\mathbf{y}(n)$ was not affected by the time-varying weights. Nevertheless, we will use Eq. (16.13.9) in a qualitative manner, replacing C_{p+1} and D_{p+1} by their constant asymptotic values, but only for the purpose of motivating the final choice of the adaptation parameter μ_{p+1}. The solution of Eq. (16.13.9), then, is

$$\gamma_{p+1}(n)= \gamma_{p+1} + (1-2\mu_{p+1}D_{p+1})^n(\gamma_{p+1}(0)-\gamma_{p+1}) \tag{16.13.10}$$

where γ_{p+1} is the asymptotic value of the weight given in Eq. (16.13.7). The stability of Eqs. (16.13.9) and (16.13.10) requires that

$$|1-2\mu_{p+1}D_{p+1}| < 1 \tag{16.13.11}$$

If we choose μ_{p+1} as

$$2\mu_{p+1} = \frac{\alpha}{D_{p+1}} \tag{16.13.12}$$

then $1-2\mu_{p+1}D_{p+1} = 1-\alpha$ will satisfy Eq. (16.13.11). Note that α was chosen to be independent of the order p. This implies that all reflection coefficients $\gamma_{p+1}(n)$ will essentially converge at the same rate. Using Eqs. (16.13.3) and (16.13.12), we write Eq. (16.13.8) as follows:

$$\gamma_{p+1}(n+1)= \gamma_{p+1}(n)+\frac{\alpha}{D_{p+1}}E\big[e_{p+1}^+(n)e_p^-(n-1)+e_{p+1}^-(n)e_p^+(n)\big] \tag{16.13.13}$$

The practical implementation of this method consists of ignoring the expectation instruction, and using a least-squares approximation for D_{p+1} of the form [1411–1413]

$$D_{p+1}(n)= (1-\lambda)\sum_{k=0}^{n}\lambda^{n-k}\big[e_p^+(k)^2+e_p^-(k-1)^2\big] \tag{16.13.14}$$

where $0 < \lambda < 1$. It may also be computed recursively by

$$D_{p+1}(n)=\lambda D_{p+1}(n-1)+(1-\lambda)\big[e_p^+(n)^2+e_p^-(n-1)^2\big] \tag{16.13.15}$$

This quantity is a measure of D_{p+1} of Eq. (16.13.6); indeed, taking expectations of both sides and assuming stationarity, we find

$$\begin{aligned} E[D_{p+1}(n)] &= (1-\lambda)\sum_{k=0}^{n}\lambda^{n-k}E[e_p^+(k)^2+e_p^-(k-1)^2] \\ &= (1-\lambda)\sum_{k=0}^{n}\lambda^{n-k}D_{p+1} = (1-\lambda^{n+1})D_{p+1} \end{aligned}$$

which converges to D_{p+1} for large n. With the above changes, we obtain the adaptive version of Eq. (16.13.13),

$$\gamma_{p+1}(n+1)=\gamma_{p+1}(n)+\frac{\alpha}{D_{p+1}(n)}\,\big[e_{p+1}^+(n)e_p^-(n-1)+e_{p+1}^-(n)e_p^+(n)\big] \tag{16.13.16}$$

It can be written in a slightly different form by defining the quantity

$$\begin{aligned} d_{p+1}(n) &= \sum_{k=0}^{n}\lambda^{n-k}[e_p^+(k)^2+e_p^-(k-1)^2] \\ &= \lambda d_{p+1}(n-1)+[e_p^+(n)^2+e_p^-(n-1)^2] \end{aligned} \tag{16.13.17}$$

and noting that $D_{p+1}(n)=(1-\lambda)d_{p+1}(n)$. Defining the new parameter $\beta=\alpha/(1-\lambda)$, we rewrite Eq. (16.13.16) in the form

$$\gamma_{p+1}(n+1)=\gamma_{p+1}(n)+\frac{\beta}{d_{p+1}(n)}\,\big[e_{p+1}^+(n)e_p^-(n-1)+e_{p+1}^-(n)e_p^+(n)\big] \tag{16.13.18}$$

This is usually operated with $\beta=1$ or, equivalently, $\alpha=1-\lambda$. This choice makes Eq. (16.13.18) equivalent to a recursive reformulation of Burg's method [1411–1413]. This may be seen as follows. Set $\beta=1$ and define the quantity $c_{p+1}(n)$ by

$$c_{p+1}(n)=\sum_{k=0}^{n}\lambda^{n-k}[2e_p^+(k)e_p^-(k-1)]$$

Then, inserting Eq. (16.13.1), with γ_{p+1} replaced by $\gamma_{p+1}(n)$, into Eq. (16.13.18), we find after some algebra

$$\gamma_{p+1}(n+1)=\frac{c_{p+1}(n)}{d_{p+1}(n)}$$

or, written explicitly

$$\gamma_{p+1}(n+1)=\frac{2\displaystyle\sum_{k=0}^{n}\lambda^{n-k}[e_p^+(k)e_p^-(k-1)]}{\displaystyle\sum_{k=0}^{n}\lambda^{n-k}[e_p^+(k)^2+e_p^-(k-1)^2]} \tag{16.13.19}$$

which corresponds to Burg's method, and also guarantees that $|\gamma_{p+1}(n+1)|$ will remain less than one at each iteration. The adaptive lattice is depicted in Fig. 16.13.1. At each time instant n, the order recursions (16.13.1) are

$$\begin{aligned} e_{p+1}^+(n) &= e_p^+(n)-\gamma_{p+1}(n)e_p^-(n-1) \\ e_{p+1}^-(n) &= e_p^-(n-1)-\gamma_{p+1}(n)e_p^+(n) \end{aligned} \tag{16.13.20}$$

for $p=0,1,\dots,M-1$, with $\gamma_{p+1}(n)$ updated in time using Eq. (16.13.18) or Eq. (16.13.19). Initialize (16.13.20) by $e_0^\pm(n)=y_n$. We summarize the computational steps as follows:

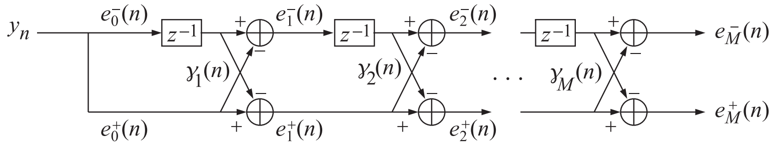


Fig. 16.13.1 Adaptive lattice predictor.

1. At time n, the coefficients $\gamma_{p+1}(n)$ and $d_{p+1}(n-1)$ are available.
2. Iterate Eq. (16.13.20) for $p=0,1,\dots,M-1$.
3. Using Eq. (16.13.17), compute $d_{p+1}(n)$ for $p=0,1,\dots,M-1$.
4. Using Eq. (16.13.18), compute $\gamma_{p+1}(n+1)$ for $p=0,1,\dots,M-1$.
5. Go to $n\to n+1$.

Next, we discuss the adaptive lattice realization of the FIR Wiener filter of Sec. 12.11. We use the same notation as in that section. The time-invariant lattice weights g_p are chosen optimally to minimize the mean-square estimation error

$$\mathcal{E}=E[e_n^2]=\min \tag{16.13.21}$$

where $e_n=x_n-\hat{x}_n$, and

$$\hat{x}_n=\sum_{p=0}^{M}g_pe_p^-(n)=[g_0,g_1,\dots,g_M]\begin{bmatrix} e_0^-(n) \\ e_1^-(n) \\ \vdots \\ e_M^-(n)\end{bmatrix}=\mathbf{g}^T\mathbf{e}^-(n) \tag{16.13.22}$$

The gradient with respect to $\mathbf{g}$ is

$$\frac{\partial\mathcal{E}}{\partial\mathbf{g}}=-2E[e_n\mathbf{e}^-(n)] \tag{16.13.23}$$

Inserting Eq. (16.13.22) into (16.13.23), we rewrite the latter as

$$\frac{\partial\mathcal{E}}{\partial\mathbf{g}}=-2(\mathbf{r}-R\mathbf{g}) \tag{16.13.24}$$

where $\mathbf{r}$ and R are defined in terms of the *backward* lattice signals $e_p^-(n)$ as

$$\mathbf{r}=E[x_n\mathbf{e}^-(n)]\,,\quad R=E[\mathbf{e}^-(n)\mathbf{e}^-(n)^T] \tag{16.13.25}$$

The gradient-descent method applied to the weights $\mathbf{g}$ is

$$\mathbf{g}(n+1)=\mathbf{g}(n)-\mathcal{M}\,\frac{\partial\mathcal{E}}{\partial\mathbf{g}(n)} \tag{16.13.26}$$

where, following the discussion of Sec. 16.6, we have used a positive definite symmetric adaptation matrix $\mathcal{M}$, to be chosen below. Then, Eq. (16.13.26) becomes

$$\mathbf{g}(n+1)=(I-2\mathcal{M}R)\mathbf{g}(n)+2\mathcal{M}\mathbf{r} \tag{16.13.27}$$

The orthogonality of the backward prediction errors $\mathbf{e}^-(n)$ causes their covariance matrix R to be diagonal

$$R=\mathrm{diag}\{E_0,E_1,\dots,E_M\} \tag{16.13.28}$$

where E_p is the variance of $e_p^-(n)$

$$E_p = E[e_p^-(n)^2]\,,\quad p = 0,1,\dots,M \tag{16.13.29}$$

If we choose $\mathcal{M}$ to be diagonal, say, $\mathcal{M} = \text{diag}\{\mu_0,\mu_1,\dots,\mu_M\}$, then the state matrix $(I - 2\mathcal{M}R)$ of Eq. (16.13.27) will also be diagonal and, therefore, Eq. (16.13.27) will decouple into its individual components

$$g_p(n+1) = (1-2\mu_pE_p)g_p(n) + 2\mu_p r_p\,,\quad p = 0,1,\dots,M \tag{16.13.30}$$

where $r_p = E[x_n e_p^-(n)]$. Its solution is

$$g_p(n) = g_p + (1-2\mu_pE_p)^n\big(g_p(0) - g_p\big) \tag{16.13.31}$$

where $g_p = r_p/E_p$ are the *optimal weights*. The convergence rate depends on the quantity $(1 - 2\mu_pE_p)$. Choosing μ_p such that

$$2\mu_p = \frac{\alpha}{E_p}\,,\quad 0 < \alpha < 1 \tag{16.13.32}$$

implies that all lattice weights $g_p(n)$ will have the same rate of convergence. Using Eqs. (16.13.32) and (16.13.23) we can rewrite Eq. (16.13.26) component-wise as follows

$$g_p(n+1) = g_p(n) + \frac{\alpha}{E_p}E[e_n e_p^-(n)]$$

Ignoring the expectation instruction, and replacing E_p by its time average,

$$E_p(n) = (1-\lambda)\sum_{k=0}^{n}\lambda^{n-k}e_p^-(k)^2 = \lambda E_p(n-1) + (1-\lambda)e_p^-(n)^2 \tag{16.13.33}$$

we obtain the adaptation equation for the pth weight

$$g_p(n+1) = g_p(n) + \frac{\alpha}{E_p(n)}\,e_n e_p^-(n)\,,\quad p = 0,1,\dots,M \tag{16.13.34}$$

Defining

$$d_p^-(n) = \sum_{k=0}^{n}\lambda^{n-k}e_p^-(k)^2 = \lambda d_p^-(n-1) + e_p^-(n)^2 \tag{16.13.35}$$

and noting that $E_p(n) = (1-\lambda)d_p^-(n)$, we rewrite Eq. (16.13.34) as

$$g_p(n+1) = g_p(n) + \frac{\beta}{d_p^-(n)}\,e_n e_p^-(n)\,,\quad p = 0,1,\dots,M \tag{16.13.36}$$

where $\beta = \alpha/(1-\lambda)$. Typically, Eq. (16.13.36) is operated with $\beta = 1$, or $\alpha = 1-\lambda$, [1411–1413]. The realization of the adaptive lattice Wiener filter is shown in Fig. 16.13.2.

A slightly different version of the algorithm is obtained by replacing e_n in Eq. (16.13.36) by $e_p(n)$, that is, the estimation error based on a pth order Wiener filter:

$$e_p(n) = x_n - \hat{x}_p(n)\,,\quad \hat{x}_p(n) = \sum_{i=0}^{p}g_ie_i^-(n)$$

It satisfies the recursions (12.11.10) through (12.11.11). This version arises by minimizing the order-p performance index $\mathcal{E}_p = E[e_p(n)^2]$ rather than the order-M performance index (16.13.21). This version is justified by the property that all lower order portions of $\mathbf{g}$ are already optimal. If $\{g_0, g_1,\dots,g_{p-1}\}$ are already optimal, then to go to the next order p it is only necessary to determine the optimal value of the new weight g_p, which is obtained by minimizing $\mathcal{E}_p$ with respect to g_p. The overall algorithm is summarized below:

1. At time n, the quantities $\gamma_p(n)$, $d_p(n-1)$, for $p = 1,2,\dots,M$ and $g_p(n)$, $d_p^-(n-1)$, for $p = 0,1,\dots,M$, are available, as well as the current input samples x_n, y_n.
2. Initialize in order by

$$e_0^\pm(n) = y_n\,,\quad \hat{x}_0(n) = g_0(n)e_0^-(n)\,,\quad e_0(n) = x_n - \hat{x}_0(n)$$
$$d_0^-(n) = \lambda d_0^-(n-1) + e_0^-(n)^2$$
$$g_0(n+1) = g_0(n) + \frac{\beta}{d_0^-(n)}\,e_0(n)e_0^-(n)$$

3. For $p = 1,2,\dots,M$, compute:

$$e_p^+(n) = e_{p-1}^+(n) - \gamma_p(n)e_{p-1}^-(n-1)$$
$$e_p^-(n) = e_{p-1}^-(n-1) - \gamma_p(n)e_{p-1}^+(n)$$
$$d_p(n) = \lambda d_p(n-1) + e_{p-1}^+(n)^2 + e_{p-1}^-(n-1)^2$$
$$\gamma_p(n+1) = \gamma_p(n) + \frac{\beta}{d_p(n)}\big[e_p^+(n)e_{p-1}^-(n-1) + e_p^-(n)e_{p-1}^+(n)\big]$$
$$\hat{x}_p(n) = \hat{x}_{p-1}(n) + g_p(n)e_p^-(n)$$
$$e_p(n) = e_{p-1}(n) - g_p(n)e_p^-(n)$$
$$d_p^-(n) = \lambda d_p^-(n-1) + e_p^-(n)^2$$
$$g_p(n+1) = g_p(n) + \frac{\beta}{d_p^-(n)}\,e_p(n)e_p^-(n)$$

4. Go to the next time instant, $n \to n+1$.

The adaptation of the reflection coefficients $\gamma_p(n)$ provides a gradual orthogonalization of the backward error signals $e_p^-(n)$, which in turn drive the adaptation equations for the lattice weights $g_p(n)$.

The algorithm is initialized in time by setting $\gamma_p(0) = 0$, $d_p(-1) = 0$, $g_p(0) = 0$, $d_p^-(-1) = 0$. Because initially all the γs and the delay registers of the lattice are zero, it follows that the backward output of the pth lattice section, $e_p^-(n)$, will be zero for $n < p$. The corresponding

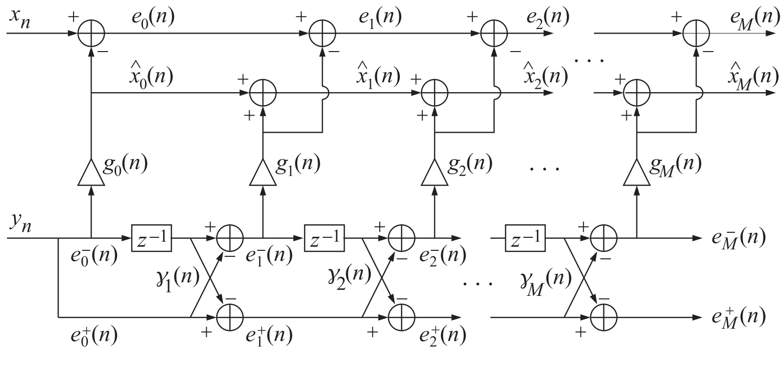


Fig. 16.13.2 Adaptive lattice Wiener filter.

$d_p^-(n)$ will also be zero and thus cannot be used in the updating of $g_p(n)$. During this startup period, we keep $g_p(n)= 0$, $n < p$. A similar problem does not arise for the γs because $d_p(n)$ contains contributions from the forward lattice outputs, which are not zero.

The function **glwf** is an implementation of the gradient lattice Wiener filter. It is the same as **lwf** with the weight adaptation parts added to it. Next, we present a simulation example. The signals x_n and y_n were generated by

$$x_n = y_n + 1.5y_{n-1} - 2y_{n-2} + u_n\,, \quad y_n = 0.75y_{n-1} - 0.5y_{n-2} + \epsilon_n$$

where u_n and ϵ_n were mutually independent, zero-mean, unit-variance, white noises. It follows from our general discussion in Sec. 16.5 that we must use a Wiener filter of order at least $M = 2$ to cancel completely the y-dependent part of x_n. Solving the order-two linear prediction problem for y_n using **bkwlev**, we find the theoretical L matrix and reflection coefficients

$$L = \begin{bmatrix} 1 & 0 & 0 \\ -0.5 & 1 & 0 \\ 0.5 & -0.75 & 1 \end{bmatrix}, \quad \gamma_1 = 0.5\,, \quad \gamma_2 = -0.5 \tag{16.13.37}$$

The direct-form coefficients of the Wiener filter are precisely the coefficients of the y-dependent part of x_n. Thus, we have

$$\mathbf{h} = \begin{bmatrix} 1 \\ 1.5 \\ -2 \end{bmatrix}, \quad \mathbf{g} = L^{-T}\mathbf{h} = \begin{bmatrix} 2 \\ 0 \\ -2 \end{bmatrix} \tag{16.13.38}$$

In the simulation we generated 100 samples of x_n and y_n (after letting the transients of the difference equation of y_n die out). The function **glwf** was run on these samples with $\lambda = 1$ and $\beta = 1$. Fig. 16.13.3 shows the adaptive reflection coefficients $\gamma_1(n)$ and $\gamma_2(n)$ versus iteration number n. The figure shows on the right the three coefficients $g_p(n)$, $p = 0,1,2$, versus n, converging to their theoretical values g_p above. For comparison purposes, we have also included the direct-form weight $h_2(n)$ adapted according to the standard LMS algorithm with $\mu = 0.01$. It should be compared to $g_2(n)$ because by construction the last elements of **g** and **h** are the same; here, $g_2 = h_2$. The LMS algorithm can be accelerated somewhat by using a larger μ, but at the expense of increasing the noisiness of the weights.

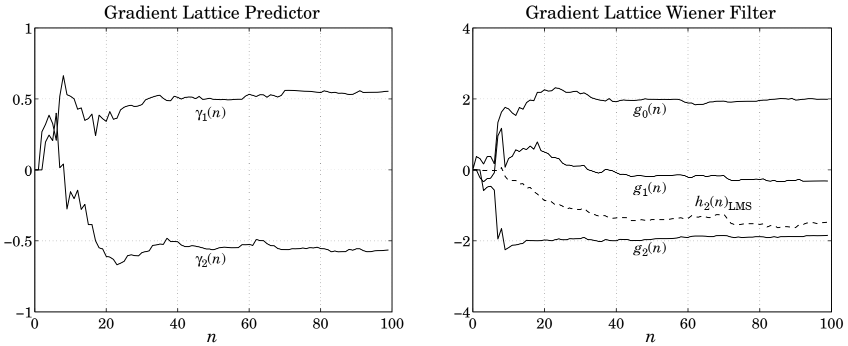


Fig. 16.13.3 Adaptive coefficients $\gamma_p(n)$ and $g_p(n)$.

16.14 Adaptive Gram-Schmidt Preprocessors

In this section we derive the *spatial* analogs of the gradient adaptive lattice algorithms. The main function of the adaptive lattice filter is to decorrelate the tapped delay-line data vector $\mathbf{y}(n)= [y_n, y_{n-1}, \dots, y_{n-M}]^T$. In effect, it carries out the Gram-Schmidt orthogonalization of the components of $\mathbf{y}(n)$ at each time instant n. In array processing problems, because the data vector $\mathbf{y}(n)= [y_0(n), y_1(n), \dots, y_M(n)]^T$ does not have the tapped-delay line property, the Gram-Schmidt orthogonalization cannot be done by a simple a lattice filter. It requires a more complicated structure that basically amounts to carrying out the lower triangular linear transformation $\mathbf{y} = B\boldsymbol{\epsilon}$, which decorrelates the covariance matrix of $\mathbf{y}$.

The Gram-Schmidt construction of an arbitrary random vector $\mathbf{y}$ was discussed in Sec. 1.6. Here, we recast these results in a way that can be used directly in gradient-adaptive implementations. The Gram-Schmidt construction proceeds recursively starting at one end, say, $\epsilon_0 = y_0$. At the mth step of the recursion, we have available the mutually decorrelated components $\{\epsilon_0, \epsilon_1, \dots, \epsilon_{m-1}\}$. The next component ϵ_m is defined by

$$\epsilon_m = y_m - \sum_{i=0}^{m-1} b_{mi}\epsilon_i\,, \quad b_{mi} = \frac{1}{E_i}E[y_m\epsilon_i] \tag{16.14.1}$$

where $E_i = E[\epsilon_i^2]$. By construction, ϵ_m is decorrelated from all the previous ϵ_is, that is, $E[\epsilon_m\epsilon_i]= 0$, $i = 0,1\dots, m-1$. The summation term in Eq. (16.14.1) represents the optimum estimate of y_m based on the previous ϵ_is and ϵ_m represents the estimation error. Therefore, the coefficients b_{mi} can also be derived by the mean-square criterion

$$\mathcal{E}_m = E[\epsilon_m^2]= \min \tag{16.14.2}$$

The gradient with respect to b_{mi} is

$$\frac{\partial\mathcal{E}_m}{\partial b_{mi}} = -2E[\epsilon_m\epsilon_i]= -2\big(E[y_m\epsilon_i]-b_{mi}E_i\big) \tag{16.14.3}$$

where we used the fact that the previous ϵ_is are already decorrelated, so that $E[\epsilon_i\epsilon_j]= \delta_{ij}E_i$, for $i,j = 0,1,\dots, m-1$. Setting the gradient to zero gives the optimum solution (16.14.1) for b_{mi}. In a gradient-adaptive approach, the coefficients b_{mi} will be time-dependent, $b_{mi}(n)$, and updated by

$$b_{mi}(n+1)= b_{mi}(n)-\mu_{mi}\frac{\partial\mathcal{E}_m}{\partial b_{mi}(n)} = b_{mi}(n)+2\mu_{mi}E[\epsilon_m\epsilon_i] \tag{16.14.4}$$

Using the above expression for the gradient, we find the difference equation

$$b_{mi}(n+1)= (1-2\mu_{mi}E_i)b_{mi}(n)+2\mu_{mi}E[y_m\epsilon_i]$$

with solution, for $n \geq 0$

$$b_{mi}(n)= b_{mi} + (1-2\mu_{mi}E_i)^n\big(b_{mi}(0)-b_{mi}\big)$$

where b_{mi} is the optimum solution (16.14.1). As in Sec. 16.13, because of the diagonal nature of the covariance matrix of the previous ϵ_is, the system of difference equations for the b_{mi}s decouples into separate scalar equations. Choosing μ_{mi} by

$$2\mu_{mi} = \frac{\alpha}{E_i}\,, \quad 0 < \alpha < 1$$

implies that all coefficients $b_{mi}(n)$ will converge at the same rate. With this choice, Eq. (16.14.4) becomes

$$b_{mi}(n+1)= b_{mi}(n)+\frac{\alpha}{E_i}E[\epsilon_m\epsilon_i]$$

As before, we may replace E_i by its weighted time average $E_i(n)=(1-\lambda)d_i(n)$, where

$$d_i(n)=\sum_{k=0}^{n}\lambda^{n-k}\epsilon_i(k)^2=\lambda d_i(n-1)+\epsilon_i(n)^2$$

Setting $\beta=\alpha/(1-\lambda)$ and dropping the expectation values, we obtain the adaptive Gram-Schmidt algorithm:

1. At time n, $b_{mi}(n)$ and $d_i(n-1)$ are available, and also the current data vector $\mathbf{y}(n)=[y_0(n),y_1(n),\dots,y_M(n)]^T$. (The algorithm is initialized in time by $b_{mi}(0)=0$ and $d_i(-1)=0$.)
2. Set $\epsilon_0(n)=y_0(n)$.
3. For $m=1,2,\dots,M$, compute:

$$\epsilon_m(n)=y_m(n)-\sum_{i=0}^{m-1}b_{mi}(n)\epsilon_i(n)$$

$$d_{m-1}(n)=\lambda d_{m-1}(n)+\epsilon_{m-1}(n)^2$$

for $i=0,1\dots,m-1$, compute:

$$b_{mi}(n+1)=b_{mi}(n)+\frac{\beta}{d_i(n)}\epsilon_m(n)\epsilon_i(n)$$

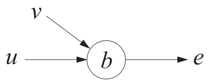
4. Go to the next time instant, $n\to n+1$.

The conventional Gram-Schmidt construction builds up the matrix B row-wise; for example in the case $M=3$

$$B=\begin{bmatrix}1&0&0&0\\b_{10}&1&0&0\\b_{20}&b_{21}&1&0\\b_{30}&b_{31}&b_{32}&1\end{bmatrix}$$

According to Eq. (16.14.1), ϵ_m is constructed from the entries of the mth row of B. This gives rise to the block-diagram realization of the Gram-Schmidt construction shown in Fig. 16.14.1. We will see shortly that each circular block represents an elementary correlation canceling operation of the type [1355,1417–1421]

$$e=u-bv$$



with

$$E[ev]=0\quad\Rightarrow\quad b=\frac{E[uv]}{E[v^2]}$$

Therefore, each block can be replaced by an ordinary adaptive CCL or by an accelerated CCL, as discussed below. This point of view leads to an alternative way of organizing the Gram-Schmidt construction with better numerical properties, known as the *modified Gram-Schmidt* procedure [1166], which builds up the matrix B column-wise. Let $\mathbf{b}_i$ be the ith column of B, so that

$$\mathbf{y}=B\boldsymbol{\epsilon}=[\mathbf{b}_0,\mathbf{b}_1,\dots,\mathbf{b}_M]\begin{bmatrix}\epsilon_0\\\epsilon_1\\\vdots\\\epsilon_M\end{bmatrix}=\sum_{j=0}^{M}\mathbf{b}_j\epsilon_j$$

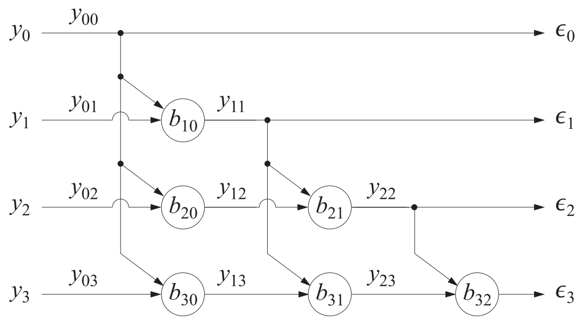


Fig. 16.14.1 Gram-Schmidt array preprocessor.

Removing the contribution of the first i columns, we define for $i=1,2,\dots,M$

$$\mathbf{y}_i=\mathbf{y}-\sum_{j=0}^{i-1}\mathbf{b}_j\epsilon_j=\sum_{j=i}^{M}\mathbf{b}_j\epsilon_j \tag{16.14.5}$$

Component-wise, we write

$$y_{im}=\sum_{j=i}^{M}b_{mj}\epsilon_j\,,\quad m=0,1,\dots,M$$

It follows from the lower-triangular nature of B that $y_{im}=0$ for $m<i$. Moreover, because B has unit diagonal, we have at $m=i$ that $y_{ii}=b_{ii}\epsilon_i=\epsilon_i$. Thus,

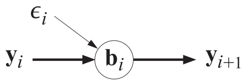
$$\epsilon_i=y_{ii} \tag{16.14.6}$$

Equation (16.14.5) can be written recursively as follows

$$\mathbf{y}_i=\mathbf{b}_i\epsilon_i+\sum_{j=i+1}^{M}\mathbf{b}_j\epsilon_j=\mathbf{b}_i\epsilon_i+\mathbf{y}_{i+1}$$

or,

$$\mathbf{y}_{i+1}=\mathbf{y}_i-\mathbf{b}_i\epsilon_i$$



and component-wise, $y_{i+1,m}=y_{im}-b_{mi}\epsilon_i$. The recursion is initialized by $\mathbf{y}_0=\mathbf{y}$. It is evident by inspecting Fig. 16.14.1 that $\mathbf{y}_i$ represents the output column vector after each column operation. It follows also that each circular block is an elementary correlation canceler. This follows by noting that $\mathbf{y}_{i+1}$ is built out of ϵ_j with $j\geq i+1$, each being uncorrelated with ϵ_i. Thus,

$$E[\epsilon_i\mathbf{y}_{i+1}]=E[\epsilon_i\mathbf{y}_i]-\mathbf{b}_iE_i=0\quad\Rightarrow\quad\mathbf{b}_i=\frac{1}{E_i}E[\epsilon_i\mathbf{y}_i]$$

or, component-wise

$$b_{mi}=\frac{1}{E_i}E[\epsilon_iy_{im}]\,,\quad m=i+1,i+2,\dots,M \tag{16.14.7}$$

An adaptive implementation can be obtained easily by writing

$$\mathbf{b}_i(n+1)=\mathbf{b}_i(n)+2\mu_iE[\epsilon_i\mathbf{y}_{i+1}]=(1-2\mu_iE_i)\mathbf{b}_i(n)+2\mu_iE[\epsilon_i\mathbf{y}_i]$$

As usual, we set $2\mu_i=\alpha/E_i$, replace E_i by $E_i(n)=(1-\lambda)d_i(n)$, and drop the expectation values to obtain the following algorithm, which adapts the matrix elements of B column-wise:

1. At time n, $b_{mi}(n)$ and $d_i(n-1)$ are available, and also the current data vector $\mathbf{y}(n)=[y_0(n),y_1(n),\dots,y_M(n)]^T$.
2. Define $y_{0m}(n)=y_m(n)$, for $m=0,1,\dots,M$.
3. For $i=0,1,\dots,M$, compute:

$$\epsilon_i(n)=y_{ii}(n)$$

$$d_i(n)=\lambda d_i(n-1)+\epsilon_i(n)^2$$

For $i+1\le m\le M$, compute:

$$y_{i+1,m}(n)=y_{im}(n)-b_{mi}(n)\epsilon_i(n)$$

$$b_{mi}(n+1)=b_{mi}(n)+\frac{\beta}{d_i(n)}\epsilon_i(n)y_{i+1,m}(n)$$

4. Go to the next time instant, $n\to n+1$.

The algorithm may be appended to provide an overall Gram-Schmidt implementation of the adaptive linear combiner of Sec. 16.4. In the decorrelated basis, the estimate of x_n and estimation error may be written order recursively as

$$\hat{x}_i(n)=\hat{x}_{i-1}(n)+g_i(n)\epsilon_i(n)\,,\quad e_i(n)=e_{i-1}(n)-g_i(n)\epsilon_i(n) \tag{16.14.8}$$

with the weights $g_i(n)$ adapted by

$$g_i(n+1)=g_i(n)+\frac{\beta}{d_i(n)}e_i(n)\epsilon_i(n)\,,\quad i=0,1,\dots,M \tag{16.14.9}$$

The function **mgs** is an implementation of the adaptive modified Gram-Schmidt procedure. At each call, the function reads the snapshot vector $\mathbf{y}$, computes the decorrelated vector $\boldsymbol{\epsilon}$, and updates the matrix elements of B in preparation for the next call. An LMS-like version can be obtained by replacing the accelerated CCLs by ordinary CCLs [1355]

$$b_{mi}(n+1)=b_{mi}(n)+2\mu\epsilon_i(n)y_{i+1,m}(n) \tag{16.14.10}$$

An exact recursive least squares version of the modified Gram-Schmidt algorithm can also be derived [1421]. It bears the same relationship to the above gradient-based version that the exact RLS lattice filter bears to the gradient lattice filter. The computational complexity of the algorithm is high because there are $M(M+1)/2$ coefficients to be adapted at each time instant, namely, the matrix elements in the strictly lower triangular part of B. By contrast, in the lattice structure there are only M reflection coefficients to be adapted. Despite its computational complexity, the algorithm is quite modular, built out of elementary CCLs.

Next, we present a simulation example of order $M=2$. The vectors $\mathbf{y}$ were constructed by

$$\mathbf{y}=\begin{bmatrix}1&0&0\\-2&1&0\\1&2&1\end{bmatrix}\begin{bmatrix}\epsilon_0\\\epsilon_1\\\epsilon_2\end{bmatrix}=B\boldsymbol{\epsilon}$$

with the components of $\boldsymbol{\epsilon}$ having variances $E_0=1$, $E_1=4$, and $E_2=9$. We generated 100 independent snapshots $\boldsymbol{\epsilon}$ and computed the corresponding $\mathbf{y}=B\boldsymbol{\epsilon}$. Fig. 16.14.2 shows the two matrix elements $b_{10}(n)$ and $b_{21}(n)$ adapted by running **mgs** on the 100 snapshots with $\lambda=1$ and $\beta=1$. They are converging to the theoretical values $b_{10}=-2$ and $b_{21}=2$. On the right, the figure shows the same two matrix elements adapted by the LMS algorithm (16.14.10) with $\mu=0.01$.

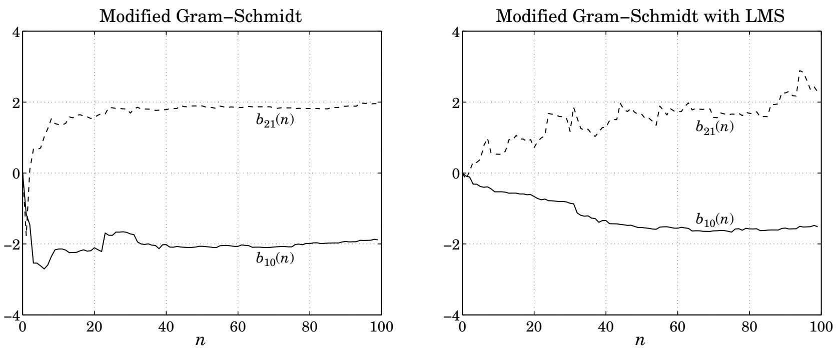


Fig. 16.14.2 Modified Gram-Schmidt algorithm and its LMS version.

16.15 Rank-One Modification of Covariance Matrices

All recursive least-squares (RLS) algorithms, conventional, lattice, and fast direct-form structures, can be derived from the rank-one updating properties of covariance matrices. In this section we discuss these properties and derive all the necessary algebraic steps and computational reductions that make the fast RLS versions possible. In the succeeding sections, we couple these results with the so-called *shift-invariance* property to close the loop, as it were, and complete the derivation of the fast RLS algorithms.

The rank-one modification of a covariance matrix R_0 is obtained by adding the rank-one term

$$R_1=R_0+\mathbf{y}\mathbf{y}^T \tag{16.15.1}$$

where $\mathbf{y}$ is a vector of the same dimension as R_0. Similarly, the modification of a cross-correlation vector $\mathbf{r}_0$ will be defined as follows, where x is a scalar

$$\mathbf{r}_1=\mathbf{r}_0+x\mathbf{y} \tag{16.15.2}$$

We define the *Wiener solutions* based on the pairs $R_0,\mathbf{r}_0$ and $R_1,\mathbf{r}_1$ by

$$\mathbf{h}_0=R_0^{-1}\mathbf{r}_0\,,\quad \mathbf{h}_1=R_1^{-1}\mathbf{r}_1 \tag{16.15.3}$$

and the corresponding *estimates* of x and estimation errors

$$\hat{x}_0=\mathbf{h}_0^T\mathbf{y}\,,\quad e_0=x-\hat{x}_0\quad\text{and}\quad \hat{x}_1=\mathbf{h}_1^T\mathbf{y}\,,\quad e_1=x-\hat{x}_1 \tag{16.15.4}$$

Similarly, using the notation of Sec. 1.8, we will consider the solution of the forward and backward prediction problems

$$R_0\mathbf{a}_0=E_{0a}\mathbf{u}\,,\quad R_1\mathbf{a}_1=E_{1a}\mathbf{u} \tag{16.15.5}$$

and

$$R_0\mathbf{b}_0=E_{0b}\mathbf{v}\,,\quad R_1\mathbf{b}_1=E_{1b}\mathbf{v} \tag{16.15.6}$$

and the corresponding forward and backward prediction errors

$$e_{0a}=\mathbf{a}_0^T\mathbf{y}\,,\quad e_{1a}=\mathbf{a}_1^T\mathbf{y}\quad\text{and}\quad e_{0b}=\mathbf{b}_0^T\mathbf{y}\,,\quad e_{1b}=\mathbf{b}_1^T\mathbf{y} \tag{16.15.7}$$

The basic question that we pose is how to construct the solution of the filtering and prediction problems 1 from the solution of the corresponding problems 0; that is, to construct $\mathbf{h}_1$ from $\mathbf{h}_0$, $\mathbf{a}_1$ from $\mathbf{a}_0$, and $\mathbf{b}_1$ from $\mathbf{b}_0$. We will generally refer to the various quantities of problem-0 as *a priori* and to the corresponding quantities of problem-1 as *a posteriori*. The constructions are carried out with the help of the so-called a priori and a posteriori *Kalman gain* vectors defined by

$$\mathbf{k}_0 = R_0^{-1}\mathbf{y}\,,\quad \mathbf{k}_1 = R_1^{-1}\mathbf{y} \tag{16.15.8}$$

We also define the so-called *likelihood variables*

$$\nu = \mathbf{y}^T R_0^{-1}\mathbf{y}\,,\quad \mu = \frac{1}{1+\nu} = \frac{1}{1+\mathbf{y}^T R_0^{-1}\mathbf{y}} \tag{16.15.9}$$

Note that the positivity condition $\nu > 0$ is equivalent to $0 < \mu < 1$. Multiplying Eq. (16.15.1) from the left by R_1^{-1} and from the right by R_0^{-1}, we obtain

$$R_0^{-1} = R_1^{-1} + R_1^{-1}\mathbf{y}\mathbf{y}^T R_0^{-1} = R_1^{-1} + \mathbf{k}_1\mathbf{k}_0^T \tag{16.15.10}$$

Acting on $\mathbf{y}$ and using the definitions (16.15.8) and (16.15.9), we find

$$R_0^{-1}\mathbf{y} = R_1^{-1}\mathbf{y} + \mathbf{k}_1\mathbf{k}_0^T\mathbf{y} \quad\Rightarrow\quad \mathbf{k}_0 = \mathbf{k}_1 + \mathbf{k}_1\nu = (1+\nu)\mathbf{k}_1 = \frac{1}{\mu}\,\mathbf{k}_1$$

or,

$$\mathbf{k}_1 = \mu\mathbf{k}_0 \tag{16.15.11}$$

It follows that

$$\mathbf{y}^T R_1^{-1}\mathbf{y} = \mathbf{k}_1^T\mathbf{y} = \mu\,\mathbf{k}_0^T\mathbf{y} = \mu\nu = \frac{\nu}{1+\nu} = 1 - \frac{1}{1+\nu} = 1-\mu$$

Thus, solving for μ

$$\mu = 1 - \mathbf{y}^T R_1^{-1}\mathbf{y} = \frac{1}{1+\mathbf{y}^T R_0^{-1}\mathbf{y}} \tag{16.15.12}$$

Solving Eq. (16.15.10) for R_1^{-1}, we obtain

$$R_1^{-1} = R_0^{-1} - \mathbf{k}_1\mathbf{k}_0^T = R_0^{-1} - \mu\,\mathbf{k}_0\mathbf{k}_0^T = R_0^{-1} - \frac{1}{1+\mathbf{y}^T R_0^{-1}\mathbf{y}}\,R_0^{-1}\mathbf{y}\mathbf{y}^T R_0^{-1} \tag{16.15.13}$$

which is recognized as the application of the matrix inversion lemma to Eq. (16.15.1). It provides the rank-one update of the inverse matrices. Denoting $P_0 = R_0^{-1}$ and $P_1 = R_1^{-1}$, we may rewrite Eq. (16.15.13) in the form

$$P_1 = P_0 - \mu\,\mathbf{k}_0\mathbf{k}_0^T\,,\quad \mathbf{k}_0 = P_0\mathbf{y}\,,\quad \mu = \frac{1}{1+\mathbf{y}^T P_0\mathbf{y}} \tag{16.15.14}$$

Before we derive the relationship between the Wiener solutions Eq. (16.15.3), we may obtain the relationship between the a priori and a posteriori estimation errors. Noting that the estimates can be written as,

$$\hat{x}_0 = \mathbf{h}_0^T\mathbf{y} = \mathbf{r}_0^T R_0^{-1}\mathbf{y} = \mathbf{r}_0^T\mathbf{k}_0$$

$$\hat{x}_1 = \mathbf{h}_1^T\mathbf{y} = \mathbf{r}_1^T R_1^{-1}\mathbf{y} = \mathbf{r}_1^T\mathbf{k}_1$$

and using Eq. (16.15.2), we obtain

$$\hat{x}_1 = \mathbf{k}_1^T\mathbf{r}_1 = (\mu\mathbf{k}_0)^T(\mathbf{r}_0 + x\mathbf{y}) = \mu\hat{x}_0 + \mu\nu x = \mu\hat{x}_0 + (1-\mu)x = x - \mu e_0$$

from which it follows that

$$e_1 = \mu e_0 \tag{16.15.15}$$

The simplest method of determining the relationship between the $\mathbf{h}_1$ and $\mathbf{h}_0$ is to act on $\mathbf{h}_0$ by the covariance matrix R_1 of problem-1, and then use the recursions (16.15.1) and (16.15.2), that is,

$$R_1\mathbf{h}_0 = (R_0 + \mathbf{y}\mathbf{y}^T)\mathbf{h}_0 = \mathbf{r}_0 + \hat{x}_0\mathbf{y} = (\mathbf{r}_1 - x\mathbf{y}) + \hat{x}_0\mathbf{y} = \mathbf{r}_1 - e_0\mathbf{y}$$

Multiplying by R_1^{-1}, we find

$$\mathbf{h}_0 = R_1^{-1}\mathbf{r}_1 - e_0 R_1^{-1}\mathbf{y} = \mathbf{h}_1 - e_0\mathbf{k}_1$$

or, solving for $\mathbf{h}_1$ and using Eqs. (16.15.11) and (16.15.15)

$$\mathbf{h}_1 = \mathbf{h}_0 + e_0\mathbf{k}_1 = \mathbf{h}_0 + \mu e_0\mathbf{k}_0 = \mathbf{h}_0 + e_1\mathbf{k}_0 \tag{16.15.16}$$

Note that the update term can be expressed either in terms of the a priori estimation error e_0 and a posteriori Kalman gain $\mathbf{k}_1$, or the a posteriori error e_1 and a priori Kalman gain $\mathbf{k}_0$. Next, we summarize what may be called the *conventional RLS* computational sequence:

1. $\mathbf{k}_0 = P_0\mathbf{y}$
2. $\nu = \mathbf{k}_0^T\mathbf{y}\,,\quad \mu = \dfrac{1}{1+\nu}$
3. $\mathbf{k}_1 = \mu\mathbf{k}_0$
4. $P_1 = P_0 - \mathbf{k}_1\mathbf{k}_0^T$
5. $\hat{x}_0 = \mathbf{h}_0^T\mathbf{y}\,,\quad e_0 = x - \hat{x}_0\,,\quad e_1 = \mu e_0\,,\quad \hat{x}_1 = x - e_1$
6. $\mathbf{h}_1 = \mathbf{h}_0 + e_0\mathbf{k}_1$

Because in step 4 an entire matrix is updated, the computational complexity of the algorithm grows quadratically with the matrix order; that is, $O(M^2)$ operations.

Next, we consider the forward and backward prediction solutions. Equations (1.8.28) and (1.8.35) applied to R_0 become

$$R_0^{-1} = \begin{bmatrix} 0 & \mathbf{0}^T \\ \mathbf{0} & \tilde{R}_0^{-1} \end{bmatrix} + \frac{1}{E_{0a}}\mathbf{a}_0\mathbf{a}_0^T = \begin{bmatrix} \bar{R}_0^{-1} & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix} + \frac{1}{E_{0b}}\mathbf{b}_0\mathbf{b}_0^T$$

Acting on $\mathbf{y}$ and using Eq. (16.15.7), we find

$$\mathbf{k}_0 = \begin{bmatrix} 0 \\ \tilde{\mathbf{k}}_0 \end{bmatrix} + \frac{e_{0a}}{E_{0a}}\,\mathbf{a}_0 = \begin{bmatrix} \bar{\mathbf{k}}_0 \\ 0 \end{bmatrix} + \frac{e_{0b}}{E_{0b}}\,\mathbf{b}_0 \tag{16.15.17}$$

where $\tilde{\mathbf{k}}_0 = \tilde{R}_0^{-1}\tilde{\mathbf{y}}$ and $\bar{\mathbf{k}}_0 = \bar{R}_0^{-1}\bar{\mathbf{y}}$, where we recall the decompositions (1.8.2) and (1.8.3)

$$\mathbf{y} = \begin{bmatrix} y_a \\ \tilde{\mathbf{y}} \end{bmatrix} = \begin{bmatrix} \bar{\mathbf{y}} \\ y_b \end{bmatrix}$$

Similarly, we obtain for the a posteriori gains

$$\mathbf{k}_1 = \begin{bmatrix} 0 \\ \tilde{\mathbf{k}}_1 \end{bmatrix} + \frac{e_{1a}}{E_{1a}}\,\mathbf{a}_1 = \begin{bmatrix} \bar{\mathbf{k}}_1 \\ 0 \end{bmatrix} + \frac{e_{1b}}{E_{1b}}\,\mathbf{b}_1 \tag{16.15.18}$$

Because $\mathbf{b}_0$ and $\mathbf{b}_1$ have last coefficients of unity, it follows that the *last* coefficients of the Kalman gains will be

$$k_{0b} = \frac{e_{0b}}{E_{0b}}\,,\quad k_{1b} = \frac{e_{1b}}{E_{1b}} \tag{16.15.19}$$

Similarly, the *first* coefficients will be

$$k_{0a} = \frac{e_{0a}}{E_{0a}}\,,\quad k_{1a} = \frac{e_{1a}}{E_{1a}} \tag{16.15.20}$$

Taking the dot product of Eq. (16.15.17) with $\mathbf{y}$ and using the definition (16.15.9) and (16.15.7), we obtain

$$\nu = \tilde{\nu} + \frac{e_{0a}^2}{E_{0a}} = \bar{\nu} + \frac{e_{0b}^2}{E_{0b}}$$

or,

$$\nu = \tilde{\nu} + e_{0a}k_{0a} = \bar{\nu} + e_{0b}k_{0b} \tag{16.15.21}$$

where $\tilde{\nu} = \tilde{\mathbf{k}}_0^T\tilde{\mathbf{y}}$ and $\bar{\nu} = \bar{\mathbf{k}}_0^T\bar{\mathbf{y}}$. Similarly, using $\mathbf{k}_1^T\mathbf{y} = 1 - \mu$ and taking the dot product of Eq. (16.15.18) with $\mathbf{y}$, we find

$$1 - \mu = 1 - \tilde{\mu} + \frac{e_{1a}^2}{E_{1a}} = 1 - \bar{\mu} + \frac{e_{1b}^2}{E_{1b}}$$

or,

$$\mu = \tilde{\mu} - \frac{e_{1a}^2}{E_{1a}} = \bar{\mu} - \frac{e_{1b}^2}{E_{1b}} \tag{16.15.22}$$

This is equivalent to Eq. (16.15.21). To relate $\mathbf{a}_1$ and $\mathbf{a}_0$, we apply the usual method of acting on the a priori solution $\mathbf{a}_0$ by the a posteriori covariance matrix R_1:

$$R_1\mathbf{a}_0 = (R_0 + \mathbf{y}\mathbf{y}^T)\mathbf{a}_0 = R_0\mathbf{a}_0 + \mathbf{y}(\mathbf{y}^T\mathbf{a}_0) = E_{0a}\mathbf{u} + e_{0a}\mathbf{y}$$

Multiplying by R_1^{-1} and using $R_1^{-1}\mathbf{u} = \mathbf{a}_1/E_{1a}$, we obtain

$$\mathbf{a}_0 = \frac{E_{0a}}{E_{1a}}\,\mathbf{a}_1 + e_{0a}\mathbf{k}_1 \tag{16.15.23}$$

This has five useful consequences. First, equating first coefficients and using Eq. (16.15.20), we obtain

$$1 = \frac{E_{0a}}{E_{1a}} + e_{0a}k_{1a} = \frac{E_{0a}}{E_{1a}} + \frac{e_{0a}e_{1a}}{E_{1a}} \tag{16.15.24}$$

or,

$$E_{1a} = E_{0a} + e_{0a}e_{1a} \tag{16.15.25}$$

Second, writing Eq. (16.15.24) in the form $E_{0a}/E_{1a} = 1 - e_{0a}k_{1a}$, we rewrite Eq. (16.15.23) as

$$\mathbf{a}_0 = (1 - e_{0a}k_{1a})\mathbf{a}_1 + e_{0a}\mathbf{k}_1 = \mathbf{a}_1 + e_{0a}(\mathbf{k}_1 - k_{1a}\mathbf{a}_1) = \mathbf{a}_1 + e_{0a}\begin{bmatrix} 0 \\ \tilde{\mathbf{k}}_1 \end{bmatrix}$$

where we used Eq. (16.15.18). Thus,

$$\mathbf{a}_1 = \mathbf{a}_0 - e_{0a}\begin{bmatrix} 0 \\ \tilde{\mathbf{k}}_1 \end{bmatrix} \tag{16.15.26}$$

Third, taking the dot product with $\mathbf{y}$ and using $\tilde{\mathbf{k}}_1^T\tilde{\mathbf{y}} = 1 - \tilde{\mu}$, we find

$$e_{1a} = \mathbf{a}_1^T\mathbf{y} = \mathbf{a}_0^T\mathbf{y} - e_{0a}(\tilde{\mathbf{k}}_1^T\tilde{\mathbf{y}}) = e_{0a} - (1 - \tilde{\mu})e_{0a} = \tilde{\mu}e_{0a} \qquad \text{or,}$$

$$e_{1a} = \tilde{\mu}e_{0a} \tag{16.15.27}$$

This is analogous to Eq. (16.15.15). Fourth, writing $e_{0a} = e_{1a}/\tilde{\mu} = (1+\tilde{\nu})e_{1a}$, it follows by adding one to Eq. (16.15.21) that

$$(1 + \nu) = (1 + \tilde{\nu}) + (1 + \tilde{\nu})e_{1a}\,\frac{e_{0a}}{E_{0a}} = (1 + \tilde{\nu})\,\frac{E_{0a} + e_{0a}e_{1a}}{E_{0a}} = (1 + \tilde{\nu})\,\frac{E_{1a}}{E_{0a}}$$

and inverting,

$$\mu = \tilde{\mu}\,\frac{E_{0a}}{E_{1a}} \tag{16.15.28}$$

This, in turn, is equivalent to Eq. (16.15.22) as can be seen by

$$\mu = \tilde{\mu}\,\frac{E_{1a} - e_{0a}e_{1a}}{E_{1a}} = \tilde{\mu} - (\tilde{\mu}e_{0a})\,\frac{e_{1a}}{E_{1a}} = \tilde{\mu} - \frac{e_{1a}^2}{E_{1a}}$$

Fifth, using Eq. (16.15.27) and the result $\tilde{\mathbf{k}}_1 = \tilde{\mu}\tilde{\mathbf{k}}_0$, we may rewrite Eq. (16.15.26) in terms of the a posteriori error e_{1a} and the a priori gain $\tilde{\mathbf{k}}_0$ as follows

$$\mathbf{a}_1 = \mathbf{a}_0 - e_{1a}\begin{bmatrix} 0 \\ \tilde{\mathbf{k}}_0 \end{bmatrix} \tag{16.15.29}$$

Defining the inverse matrices $\tilde{P}_0 = \tilde{R}_0^{-1}$ and $\tilde{P}_1 = \tilde{R}_1^{-1}$, we summarize the conventional RLS computational sequence for the *forward predictor:*

1. $\tilde{\mathbf{k}}_0 = \tilde{P}_0\tilde{\mathbf{y}}$
2. $\tilde{\nu} = \tilde{\mathbf{k}}_0^T\tilde{\mathbf{y}}, \quad \tilde{\mu} = \dfrac{1}{1 + \tilde{\nu}}$
3. $\tilde{\mathbf{k}}_1 = \tilde{\mu}\,\tilde{\mathbf{k}}_0$
4. $\tilde{P}_1 = \tilde{P}_0 - \tilde{\mathbf{k}}_1\tilde{\mathbf{k}}_0^T$
5. $e_{0a} = \mathbf{a}_0^T\mathbf{y}, \quad e_{1a} = \tilde{\mu}e_{0a}$
6. $\mathbf{a}_1 = \mathbf{a}_0 - e_{0a}\begin{bmatrix} 0 \\ \tilde{\mathbf{k}}_1 \end{bmatrix}$

The fast RLS algorithms make use also of the backward predictors. Starting with $R_1\mathbf{b}_0 = (R_0 + \mathbf{y}\mathbf{y}^T)\mathbf{b}_0 = E_{0b}\mathbf{v} + e_{0b}\mathbf{y}$, and following similar steps as for the forward case, we obtain parallel results for the backward predictor, that is,

$$\mathbf{b}_0 = \frac{E_{0b}}{E_{1b}}\,\mathbf{b}_1 + e_{0b}\mathbf{k}_1 \tag{16.15.30}$$

from which it follows that

$$1 = \frac{E_{0b}}{E_{1b}} + e_{0b}k_{1b} = \frac{E_{0b}}{E_{1b}} + \frac{e_{0b}e_{1b}}{E_{1b}} \tag{16.15.31}$$

or,

$$E_{1b} = E_{0b} + e_{0b}e_{1b} \tag{16.15.32}$$

Similarly, we have $\bar{\mathbf{k}}_1 = \bar{\mu}\,\bar{\mathbf{k}}_0$, and

$$e_{1b} = \bar{\mu}e_{0b} \tag{16.15.33}$$

and the equivalencies

$$\nu = \bar{\nu} + \frac{e_{0b}^2}{E_{0b}} \quad\Leftrightarrow\quad \mu = \bar{\mu} - \frac{e_{1b}^2}{E_{1b}} \quad\Leftrightarrow\quad \mu = \bar{\mu}\,\frac{E_{0b}}{E_{1b}} \tag{16.15.34}$$

Finally, the update equations of $\mathbf{b}_1$ are

$$\mathbf{b}_1 = \mathbf{b}_0 - e_{0b}\begin{bmatrix} \bar{\mathbf{k}}_1 \\ 0 \end{bmatrix} = \mathbf{b}_0 - e_{1b}\begin{bmatrix} \bar{\mathbf{k}}_0 \\ 0 \end{bmatrix} \tag{16.15.35}$$

Writing Eq. (16.15.31) in the form $E_{1b}/E_{0b} = 1/(1 - e_{0b}k_{1b})$, and solving Eq. (16.15.30) for $\mathbf{b}_1$, we have the alternative expression

$$\mathbf{b}_1 = \frac{E_{1b}}{E_{0b}}(\mathbf{b}_0 - e_{0b}\mathbf{k}_1) = \frac{\mathbf{b}_0 - e_{0b}\mathbf{k}_1}{1 - e_{0b}k_{1b}} \tag{16.15.36}$$

This is used in the so-called *fast Kalman* (FK) [1422,1423] computational sequence, which we summarize below

1. $e_{0a} = \mathbf{a}_0^T\mathbf{y}$
2. $\mathbf{a}_1 = \mathbf{a}_0 - e_{0a}\begin{bmatrix} 0 \\ \tilde{\mathbf{k}}_1 \end{bmatrix}$
3. $e_{1a} = \mathbf{a}_1^T\mathbf{y}$
4. $E_{1a} = E_{0a} + e_{0a}e_{1a}$
5. Compute the first element of $\mathbf{k}_1$, $k_{1a} = \dfrac{e_{1a}}{E_{1a}}$
6. $\mathbf{k}_1 = \begin{bmatrix} 0 \\ \tilde{\mathbf{k}}_1 \end{bmatrix} + k_{1a}\mathbf{a}_1$, and extract the last element of $\mathbf{k}_1$, k_{1b}
7. $e_{0b} = \mathbf{b}_0^T\mathbf{y}$
8. $\mathbf{b}_1 = \dfrac{\mathbf{b}_0 - e_{0b}\mathbf{k}_1}{1 - e_{0b}k_{1b}}$
9. $\begin{bmatrix} \bar{\mathbf{k}}_1 \\ 0 \end{bmatrix} = \mathbf{k}_1 - k_{1b}\mathbf{b}_1$
10. $\hat{x}_0 = \mathbf{h}_0^T\mathbf{y}\,,\quad e_0 = x - \hat{x}_0\,,\quad \mathbf{h}_1 = \mathbf{h}_0 + e_0\mathbf{k}_1\,,\quad \hat{x}_1 = \mathbf{h}_1^T\mathbf{y}\,,\quad e_1 = x - \hat{x}_1$

Step 9 is obtained from Eq. (16.15.18). Steps 1–9 perform the calculation and update of the Kalman gain vector $\mathbf{k}_1$, which is used in step 10 for the Wiener filtering part. This algorithm avoids the updating of the inverse autocorrelation matrices P_0 and P_1. The computationally intensive parts of the algorithm are the computation of the inner products and the vector updates. Steps 1, 2, 3, 6, 7, and 9 require M operations each, and step 8 requires $2M$ operations. Thus, the gain calculation in steps 1–9 requires a total of $8M$ operations. The Wiener filtering and updating part in step 10 require an additional $3M$ operations. Thus, the overall complexity grows like $8M + 3M = 11M$ operations; that is, linearly in the order M.

Several of the above operations can be avoided. In particular, the computation of the error e_{1a} in step 3 can be done by Eq. (16.15.27), thus, avoiding the inner product. Similarly, the inner product in step 7 can be avoided by solving Eq. (16.15.19) for e_{0b}, that is, $e_{0b} = k_{0b}E_{0b}$. Also, the division by the overall scalar factor $1/(1-e_{0b}k_{1b})$ in step 8 can be avoided by using Eq. (16.15.35) instead. This saves $3M$ out of the $8M$ computations—a 40% reduction. Similarly, the operation $\hat{x}_1 = \mathbf{h}_1^T\mathbf{y}$ in the Wiener filtering part can be avoided by $e_1 = \mu e_0$ and $\hat{x}_1 = x - e_1$. The resulting computational sequence is the so-called *fast a posteriori error sequential technique* (FAEST) [1424]. It uses the a posteriori errors and the a priori Kalman gains, and is summarized below

1. $e_{0a} = \mathbf{a}_0^T\mathbf{y}$
2. $e_{1a} = \tilde{\mu}e_{0a} = e_{0a}/(1+\tilde{\nu})$
3. Compute the first element of $\mathbf{k}_0$, $k_{0a} = \dfrac{e_{0a}}{E_{0a}}$
4. $E_{1a} = E_{0a} + e_{0a}e_{1a}$
5. $\mathbf{k}_0 = \begin{bmatrix} 0 \\ \tilde{\mathbf{k}}_0 \end{bmatrix} + k_{0a}\mathbf{a}_0$, and extract the last element of $\mathbf{k}_0$, k_{0b}
6. $e_{0b} = k_{0b}E_{0b}$
7. $\begin{bmatrix} \bar{\mathbf{k}}_0 \\ 0 \end{bmatrix} = \mathbf{k}_0 - k_{0b}\mathbf{b}_0$
8. $\nu = \tilde{\nu} + e_{0a}k_{0a}\,,\quad \bar{\nu} = \nu - e_{0b}k_{0b}$
9. $e_{1b} = \bar{\mu}e_{0b} = e_{0b}/(1+\bar{\nu})$
10. $E_{1b} = E_{0b} + e_{0b}e_{1b}$

11. $\mathbf{a}_1 = \mathbf{a}_0 - e_{1a}\begin{bmatrix} 0 \\ \tilde{\mathbf{k}}_0 \end{bmatrix}$
12. $\mathbf{b}_1 = \mathbf{b}_0 - e_{1b}\begin{bmatrix} \bar{\mathbf{k}}_0 \\ 0 \end{bmatrix}$
13. $\hat{x}_0 = \mathbf{h}_0^T\mathbf{y}\,,\quad e_0 = x - \hat{x}_0\,,\quad e_1 = \mu e_0 = e_0/(1+\nu)\,,\quad \hat{x}_1 = x - e_1$
14. $\mathbf{h}_1 = \mathbf{h}_0 + e_1\mathbf{k}_0$

Step 8 was obtained from Eq. (16.15.21). Steps l, 5, 7, 11, and 12 require M operations each. Therefore, the gain calculation can be done with $5M$ operations. The last two Wiener filtering steps require an additional $2M$ operations. Thus, the total operation count grows like $5M + 2M = 7M$. The so-called *fast transversal filter* (FTF) [1425] computational sequence is essentially identical to FAEST, but works directly with the variables μ instead of ν. The only change is to replace step 8 by the following:

8. $\mu = \tilde{\mu}\dfrac{E_{0a}}{E_{1a}}\,,\quad \bar{\mu} = \dfrac{\mu}{1 - e_{0b}k_{0b}\mu}\qquad$ (FTF)

The second equation is obtained from (16.15.34), (16.15.31), and the proportionality $\mathbf{k}_1 = \mu\,\mathbf{k}_0$, which implies the same for the last elements of these vectors, $k_{1b} = \mu k_{0b}$. We have

$$\bar{\mu} = \mu\,\frac{E_{1b}}{E_{0b}} = \frac{\mu}{1 - e_{0b}k_{1b}} = \frac{\mu}{1 - e_{0b}k_{0b}\mu}$$

The above computational sequences are organized to start with the tilde quantities, such as $\tilde{\nu}$ and $\tilde{\mathbf{k}}_0$, and end up with the bar quantities such as $\bar{\nu}$ and $\bar{\mathbf{k}}_0$. The reason has to do with the shift-invariance property, which implies that all bar quantities computed at the present iteration become the corresponding tilde quantities of the *next* iteration; for example,

$$\tilde{\nu}(n+1) = \bar{\nu}(n)\,,\quad \tilde{\mathbf{k}}_0(n+1) = \bar{\mathbf{k}}_0(n)$$

This property allows the repetition of the computational cycle from one time instant to the next. As we have seen, the computational savings of FAEST over FK, and FK over conventional RLS, have nothing to do with shift invariance but rather are consequences of the rank-one updating properties.

The FAEST, FTF, and FK algorithms are the fastest known RLS algorithms. Unfortunately, they can exhibit numerically unstable behavior and require the use of rescue devices and re-initializations for continuous operation [1426–1435]. Next, we consider the lattice formulations. Equations (1.8.50) can be applied to the *a priori lattice*

$$\begin{aligned} e_{0a} &= \bar{e}_{0a} - \gamma_{0b}\tilde{e}_{0b} \\ e_{0b} &= \tilde{e}_{0b} - \gamma_{0a}\bar{e}_{0a} \end{aligned} \tag{16.15.37}$$

and *a posteriori lattice*

$$\begin{aligned} e_{1a} &= \bar{e}_{1a} - \gamma_{1b}\tilde{e}_{1b} \\ e_{1b} &= \tilde{e}_{1b} - \gamma_{1a}\bar{e}_{1a} \end{aligned} \tag{16.15.38}$$

with the reflection coefficients computed by

$$\gamma_{0a} = \frac{\Delta_0}{\bar{E}_{0a}}\,,\quad \gamma_{0b} = \frac{\Delta_0}{\tilde{E}_{0b}}\quad \text{and}\quad \gamma_{1a} = \frac{\Delta_1}{\bar{E}_{1a}}\,,\quad \gamma_{1b} = \frac{\Delta_1}{\tilde{E}_{1b}} \tag{16.15.39}$$

To find the relationship between Δ_1 and Δ_0, we use Eq. (1.8.44) applied to R_1

$$R_1\begin{bmatrix} 0 \\ \tilde{\mathbf{b}}_1 \end{bmatrix} = \Delta_1\mathbf{u} + \tilde{E}_{1b}\mathbf{v}\,,\quad R_1\begin{bmatrix} \bar{\mathbf{a}}_1 \\ 0 \end{bmatrix} = \Delta_1\mathbf{v} + \bar{E}_{1a}\mathbf{u} \tag{16.15.40}$$

Applying Eq. (1.8.44) also to R_0, we obtain

$$R_1\begin{bmatrix}\bar{\mathbf{a}}_0\\0\end{bmatrix} = (R_0+\mathbf{y}\mathbf{y}^T)\begin{bmatrix}\bar{\mathbf{a}}_0\\0\end{bmatrix} = \Delta_0\mathbf{v}+\bar{E}_{0a}\mathbf{u}+\bar{e}_{0a}\mathbf{y} \tag{16.15.41}$$

and

$$R_1\begin{bmatrix}0\\\tilde{\mathbf{b}}_0\end{bmatrix} = (R_0+\mathbf{y}\mathbf{y}^T)\begin{bmatrix}0\\\tilde{\mathbf{b}}_0\end{bmatrix} = \Delta_0\mathbf{u}+\tilde{E}_{0b}\mathbf{v}+\tilde{e}_{0b}\mathbf{y} \tag{16.15.42}$$

Forming the dot products,

$$[0,\,\tilde{\mathbf{b}}_1^T]R_1\begin{bmatrix}\bar{\mathbf{a}}_0\\0\end{bmatrix}\quad\text{and}\quad[0,\,\tilde{\mathbf{b}}_0^T]R_1\begin{bmatrix}\bar{\mathbf{a}}_1\\0\end{bmatrix}$$

we obtain the two alternative expressions

$$\Delta_1=\Delta_0+\bar{e}_{0a}\tilde{e}_{1b}\,,\quad \Delta_1=\Delta_0+\bar{e}_{1a}\tilde{e}_{0b} \tag{16.15.43}$$

They represent the least-squares modifications of the partial correlation (1.8.53). The two expressions are equivalent. Applying Eq. (16.15.33) to $\tilde{e}_{1b}$, we have $\tilde{e}_{1b}=\bar{\tilde{\mu}}\tilde{e}_{0b}$. Applying Eq. (16.15.27) to $\bar{e}_{1a}$, we have $\bar{e}_{1a}=\tilde{\bar{\mu}}\bar{e}_{0a}$. But, $\bar{\tilde{\nu}}=\tilde{\bar{\nu}}$ because, as is evident from Eq. (1.8.51), the tilde part of $\bar{\mathbf{y}}$ is the same as the bar part of $\tilde{\mathbf{y}}$, namely, $\mathbf{y}_c$. Thus, $\bar{\tilde{\nu}}=\tilde{\bar{\nu}}=\mathbf{y}_c^TR_{0c}^{-1}\mathbf{y}_c$, which implies $\bar{\tilde{\mu}}=\tilde{\bar{\mu}}$. Applying Eq. (16.15.34), we have the updating equation $\tilde{\mu}=\bar{\tilde{\mu}}-\tilde{e}_{1b}^2/\tilde{E}_{1b}$.

As for the Wiener filtering part, we can apply the order-updating equations (1.8.24) through (1.8.27) to the a priori and a posteriori problems to get

$$\begin{aligned}\hat{x}_0&=\bar{x}_0+g_{0b}e_{0b}\,,\quad e_0=\bar{e}_0-g_{0b}e_{0b}\\ \hat{x}_1&=\bar{x}_1+g_{1b}e_{1b}\,,\quad e_1=\bar{e}_1-g_{1b}e_{1b}\end{aligned} \tag{16.15.44}$$

where g_{0b} and g_{1b} are the *last* components of the lattice weight vectors $\mathbf{g}_0$ and $\mathbf{g}_1$. Because of the relationship $\mathbf{h}=L^T\mathbf{g}$, it follows that the last component of $\mathbf{h}$ is equal to the last component of $\mathbf{g}$. Thus, extracting the last components of the relationship $\mathbf{h}_1=\mathbf{h}_0+e_0\mathbf{k}_1$, we find

$$g_{1b}=g_{0b}+e_0k_{1b}=g_{0b}+e_0\,\frac{e_{1b}}{E_{1b}} \tag{16.15.45}$$

This provides a *direct* way to update the gs. The more conventional updating method is indirect; it is obtained by writing

$$g_{0b}=\frac{\rho_{0b}}{E_{0b}}\,,\quad g_{1b}=\frac{\rho_{1b}}{E_{1b}} \tag{16.15.46}$$

Using Eq. (16.15.44), we can find a recursion for the ρs as follows

$$\rho_{1b}=E_{1b}g_{1b}=E_{1b}g_{0b}+(\bar{e}_0-g_{0b}e_{0b})e_{1b}=(E_{1b}-e_{0b}e_{1b})g_{0b}+\bar{e}_0e_{1b}$$

or, using $E_{1b}-e_{0b}e_{1b}=E_{0b}$ and $\rho_{0b}=E_{0b}g_{0b}$, we obtain

$$\rho_{1b}=\rho_{0b}+\bar{e}_0e_{1b}=\rho_{0b}+\frac{1}{\bar{\mu}}\,\bar{e}_1e_{1b} \tag{16.15.47}$$

The *conventional RLS lattice* (RLSL) computational sequence is summarized below [1436–1444]:

1. $\Delta_1=\Delta_0+\tilde{e}_{1b}\bar{e}_{0a}=\Delta_0+\tilde{e}_{1b}\bar{e}_{1a}/\bar{\tilde{\mu}}$
2. $\gamma_{1a}=\dfrac{\Delta_1}{\bar{E}_{1a}}\,,\quad\gamma_{1b}=\dfrac{\Delta_1}{\tilde{E}_{1b}}$

3. $e_{1a}=\bar{e}_{1a}-\gamma_{1b}\tilde{e}_{1b}\,,\quad e_{1b}=\tilde{e}_{1b}-\gamma_{1a}\bar{e}_{1a}$
4. $E_{1a}=\bar{E}_{1a}-\gamma_{1b}\Delta_1\,,\quad E_{1b}=\tilde{E}_{1b}-\gamma_{1a}\Delta_1$
5. $\tilde{\mu}=\bar{\tilde{\mu}}-\dfrac{\tilde{e}_{1b}^2}{\tilde{E}_{1b}}$
6. $\rho_{1b}=\rho_{0b}+\bar{e}_1e_{1b}/\bar{\mu}$
7. $g_{1b}=\dfrac{\rho_{1b}}{E_{1b}}$
8. $e_1=\bar{e}_1-g_{1b}e_{1b}\,,\quad\hat{x}_1=x-e_1$

This is referred to as the *a posteriori* RLS lattice because it uses the a posteriori lattice equations (16.15.38). There are 14 multiplication/division operations in this sequence. We will see later that the use of the so-called forgetting factor λ requires 2 more multiplications. Thus, the total number of operations is 16. Because this sequence must be performed once per order, it follows that, for an order-M problem, the computational complexity of the RLS lattice will be $16M$ operations per time update. This is to be compared with $7M$ for the FAEST direct-form version. However, as we have already mentioned, the direct-form versions can exhibit numerical instabilities. By contrast, the lattice algorithms are numerically stable [1431,1445].

Many other variations of the RLS lattice are possible. For example, there is a version based on Eq. (16.15.37), called the *a priori* RLS lattice algorithm [1358,1444], or a version called the *double* (a priori/a posteriori) RLS algorithm [1441,1444] that uses Eqs. (16.15.37) and (16.15.38) simultaneously. This version avoids the computation of the likelihood parameter μ. Like Eq. (16.15.45), we can also obtain direct updating formulas for the reflection coefficients, thereby avoiding the recursion (16.15.43) for the partial correlations Δ. Using the second term of Eqs. (16.15.43) and (16.15.25) applied to $\bar{E}_{1a}$, that is, $\bar{E}_{1a}+\bar{E}_{0a}+\bar{e}_{0a}\bar{e}_{1a}$. we find

$$\begin{aligned}\gamma_{1a}&=\frac{\Delta_1}{\bar{E}_{1a}}=\frac{\Delta_0+\bar{e}_{1a}\tilde{e}_{0b}}{\bar{E}_{1a}}=\frac{\gamma_{0a}\bar{E}_{0a}+\bar{e}_{1a}\tilde{e}_{0b}}{\bar{E}_{1a}}\\ &=\frac{\gamma_{0a}(\bar{E}_{1a}-\bar{e}_{0a}\bar{e}_{1a})+\bar{e}_{1a}\tilde{e}_{0b}}{\bar{E}_{1a}}=\gamma_{0a}+\frac{\bar{e}_{1a}}{\bar{E}_{1a}}(\tilde{e}_{0b}-\gamma_{0a}\bar{e}_{0a})\end{aligned}$$

and using Eq. (16.15.37), we obtain

$$\gamma_{1a}=\gamma_{0a}+e_{0b}\,\frac{\bar{e}_{1a}}{\bar{E}_{1a}} \tag{16.15.48}$$

Similarly, working with the first term of Eq. (16.15.43), we find

$$\gamma_{1b}=\gamma_{0b}+e_{0a}\,\frac{\tilde{e}_{1b}}{\tilde{E}_{1b}} \tag{16.15.49}$$

Replacing $\bar{e}_{1a}=\tilde{\bar{\mu}}\bar{e}_{0a}$ and $\tilde{e}_{1b}=\bar{\tilde{\mu}}\tilde{e}_{0b}$ in the above equations gives rise to the so-called *a priori direct-updating* RLS lattice [1445], also called the *a priori error-feedback* lattice because the outputs e_{0a} and e_{0b} of the a priori lattice equations (16.15.37) are used to update the reflection coefficients.

An *a posteriori* direct or error-feedback algorithm [1445] can also be obtained by working with the a posteriori lattice Eq. (16.15.38). In this case, we must express e_{0a} and e_{0b} in terms of the a posteriori quantities as follows:

$$e_{0a}=\bar{e}_{0a}-\gamma_{0b}\tilde{e}_{0b}=(\bar{e}_{1a}-\gamma_{0b}\tilde{e}_{1b})/\tilde{\bar{\mu}}\quad\text{and}\quad e_{0b}=(\tilde{e}_{1b}-\gamma_{0a}\bar{e}_{1a})/\bar{\tilde{\mu}}$$

The a priori and a posteriori error-feedback lattice algorithms are computationally somewhat more expensive—requiring $O(20M)$ operations—than the conventional RLS lattice. But, they have much better *numerical accuracy* under quantization [1445] and, of course, their long-term behavior is numerically stable.

Below we list the computational sequence of what may be called the *double/direct* RLS lattice algorithm that, on the one hand, uses direct-updating for increased numerical accuracy, and on the other, has the same computational complexity as the conventional a posteriori RLS lattice, namely, $16M$ operations [1487]:

1. $e_{0a} = \bar{e}_{0a} - \gamma_{0b}\tilde{e}_{0b}\,, \quad e_{0b} = \tilde{e}_{0b} - \gamma_{0a}\bar{e}_{0a}$
2. $\gamma_{1a} = \gamma_{0a} + e_{0b}\,\dfrac{\bar{e}_{1a}}{\bar{E}_{1a}}\,, \quad \gamma_{1b} = \gamma_{0b} + e_{0a}\,\dfrac{\tilde{e}_{1b}}{\tilde{E}_{1b}}$
3. $e_{1a} = \bar{e}_{1a} - \gamma_{1b}\tilde{e}_{1b}\,, \quad e_{1b} = \tilde{e}_{1b} - \gamma_{1a}\bar{e}_{1a}$
4. $E_{1a} = E_{0a} + e_{1a}e_{0a}\,, \quad E_{1b} = E_{0b} + e_{1b}e_{0b}$
5. $e_0 = \bar{e}_0 - g_{0b}e_{eb}$
6. $g_{1b} = g_{0b} + e_0\,\dfrac{e_{1b}}{E_{1b}}$
7. $e_1 = \bar{e}_1 - g_{1b}e_{1b}\,, \quad \hat{x}_1 = x - e_1$

It uses simultaneously the a priori and a posteriori lattice equations (16.15.37) and (16.15.38). There are 14 operations (plus 2 for the forgetting factor) per order per time update, that is, a total of $16M$ per time update.

Finally, we discuss the sense in which the a priori and a posteriori backward errors e_{0b} and e_{1b} provide a decorrelation of the covariance matrices R_0 and R_1. Following Eqs. (1.8.13) and (1.8.17), we write the LU factorizations of the a priori and a posteriori problems

$$L_0R_0L_0^T = D_{0b}\,, \quad L_1R_1L_1^T = D_{1b} \tag{16.15.50}$$

where L_0 and L_1 have as rows the backward predictors $\mathbf{b}_0^T = [\boldsymbol{\beta}_0^T, 1]$ and $\mathbf{b}_1^T = [\boldsymbol{\beta}_1^T, 1]$.

$$L_0 = \begin{bmatrix} \bar{L}_0 & \mathbf{0} \\ \boldsymbol{\beta}_0^T & 1 \end{bmatrix}, \quad L_1 = \begin{bmatrix} \bar{L}_1 & \mathbf{0} \\ \boldsymbol{\beta}_1^T & 1 \end{bmatrix} \tag{16.15.51}$$

The corresponding backward basis vectors are constructed by

$$\mathbf{e}_{0b} = L_0\mathbf{y} = \begin{bmatrix} \bar{L}_0 & \mathbf{0} \\ \boldsymbol{\beta}_0^T & 1 \end{bmatrix}\begin{bmatrix} \bar{\mathbf{y}} \\ y_b \end{bmatrix} = \begin{bmatrix} \bar{L}_0\bar{\mathbf{y}} \\ \mathbf{b}_0^T\mathbf{y} \end{bmatrix} = \begin{bmatrix} \bar{\mathbf{e}}_{0b} \\ e_{0b} \end{bmatrix} \tag{16.15.52}$$

and

$$\mathbf{e}_{1b} = L_1\mathbf{y} = \begin{bmatrix} \bar{L}_1 & \mathbf{0} \\ \boldsymbol{\beta}_1^T & 1 \end{bmatrix}\begin{bmatrix} \bar{\mathbf{y}} \\ y_b \end{bmatrix} = \begin{bmatrix} \bar{L}_1\bar{\mathbf{y}} \\ \mathbf{b}_1^T\mathbf{y} \end{bmatrix} = \begin{bmatrix} \bar{\mathbf{e}}_{1b} \\ e_{1b} \end{bmatrix} \tag{16.15.53}$$

The rank-one updating property (16.15.1) for the Rs can be translated into an updating equation for the LU factorizations[112–114], in the following form:

$$L_1 = LL_0 \tag{16.15.54}$$

It turns out that the unit lower triangular matrix L can be built entirely out of the a priori backward errors $\mathbf{e}_{0b}$, as we show below. The determining equation for L may be found by

$$D_{1b} = L_1R_1L_1^T = LL_0(R_0 + \mathbf{y}\mathbf{y}^T)L_0^TL^T = L(D_{0b} + \mathbf{e}_{0b}\mathbf{e}_{0b}^T)L^T \tag{16.15.55}$$

Thus, L performs the LU factorization of the rank-one update of a diagonal matrix, namely, $D_{0b} + \mathbf{e}_{0b}\mathbf{e}_{0b}^T$. The solution is easily found by introducing the block decompositions

$$L = \begin{bmatrix} \bar{L} & \mathbf{0} \\ \boldsymbol{\beta}^T & 1 \end{bmatrix}, \quad D_{1b} = \begin{bmatrix} \bar{D}_{1b} & \mathbf{0} \\ \mathbf{0}^T & E_{1b} \end{bmatrix}, \quad D_{0b} + \mathbf{e}_{0b}\mathbf{e}_{0b}^T = \begin{bmatrix} \bar{D}_{0b} + \bar{\mathbf{e}}_{0b}\bar{\mathbf{e}}_{0b}^T & e_{0b}\bar{\mathbf{e}}_{0b} \\ e_{0b}\bar{\mathbf{e}}_{0b}^T & E_{0b} + e_{0b}^2 \end{bmatrix}$$

Using the methods of Sec. 1.8, e.g., Eqs. (1.8.7) and (1.8.11) applied to this problem, we find the solution

$$\boldsymbol{\beta} = -\bar{\mu}e_{0b}\bar{D}_{0b}^{-1}\bar{\mathbf{e}}_{0b}\,, \quad \bar{\mu} = \frac{1}{1 + \bar{\mathbf{e}}_{0b}^T\bar{D}_{0b}^{-1}\bar{\mathbf{e}}_{0b}} \tag{16.15.56}$$

Using $\bar{R}_0^{-1} = \bar{L}_0^T\bar{D}_{0b}^{-1}\bar{L}_0$, we recognize

$$\bar{\mathbf{e}}_{0b}^T\bar{D}_{0b}^{-1}\bar{\mathbf{e}}_{0b} = \bar{\mathbf{y}}^T\bar{L}_0^T\bar{D}_{0b}^{-1}\bar{L}_0\bar{\mathbf{y}} = \bar{\mathbf{y}}^T\bar{R}_0^{-1}\bar{\mathbf{y}} = \bar{\nu}$$

Therefore, the quantity $\bar{\mu}$ defined above is the usual one. Similarly, we find

$$E_{1b} = (E_{0b} + e_{0b}^2) + e_{0b}\bar{\mathbf{e}}_{0b}^T\,\boldsymbol{\beta} = E_{0b} + e_{0b}^2 - \bar{\mu}e_{0b}^2\bar{\nu}$$

Noting that $1 - \bar{\mu}\bar{\nu} = \bar{\mu}$, this reduces to Eq. (16.15.32). Writing $\bar{D}_{0b}^{-1}\bar{\mathbf{e}}_{0b} = \bar{L}_0^{-T}\bar{R}_0^{-1}\bar{\mathbf{y}} = \bar{L}_0^{-T}\bar{\mathbf{k}}_0$, we may express $\boldsymbol{\beta}$ in terms of the Kalman gain vector:

$$\boldsymbol{\beta} = -\bar{\mu}e_{0b}L_0^{-T}\bar{\mathbf{k}}_0 \tag{16.15.57}$$

It easy to verify that the block-decomposed form of Eq. (16.15.54) is equivalent to

$$\bar{L}_1 = \bar{L}\bar{L}_0\,, \quad \boldsymbol{\beta}_1 = \boldsymbol{\beta}_0 + \bar{L}_0^T\boldsymbol{\beta} \tag{16.15.58}$$

Because of Eq. (16.15.57), the updating equation for the $\boldsymbol{\beta}$s is equivalent to Eq. (16.15.35). Using this formalism, we may show the proportionality between the a posteriori and a priori backward errors. We have $\mathbf{e}_{1b} = L_1\mathbf{y} = LL_0\mathbf{y} = L\mathbf{e}_{0b}$, and in block form

$$\mathbf{e}_{1b} = \begin{bmatrix} \bar{L} & \mathbf{0} \\ \boldsymbol{\beta}^T & 1 \end{bmatrix}\begin{bmatrix} \bar{\mathbf{e}}_{0b} \\ e_{0b} \end{bmatrix} = \begin{bmatrix} \bar{L}\mathbf{e}_{0b} \\ e_{0b} + \boldsymbol{\beta}^T\bar{\mathbf{e}}_{0b} \end{bmatrix}$$

Therefore, $e_{1b} = e_{0b} + \boldsymbol{\beta}^T\bar{\mathbf{e}}_{0b} = e_{0b} - \bar{\mu}e_{0b}\bar{\nu} = \bar{\mu}e_{0b}$. It follows that L acting on $\mathbf{e}_{0b}$ can be replaced by the diagonal matrix of $\bar{\mu}$s acting on $\mathbf{e}_{0b}$. The double/direct lattice algorithm effectively provides the error signals required to build L. For example, Eq. (16.15.56) can be written in a form that avoids the computation of the μs

$$\boldsymbol{\beta} = -\bar{\mu}e_{0b}\bar{D}_{0b}^{-1}\bar{\mathbf{e}}_{0b} = -e_{1b}\bar{D}_{0b}^{-1}\bar{\mathbf{e}}_{0b} \tag{16.15.59}$$

The a priori and a posteriori estimates $\hat{x}_0$ and $\hat{x}_1$ may also be expressed in the backward bases. Defining $\mathbf{g}_0 = L_0^{-T}\mathbf{h}_0$, we find $\hat{x}_0 = \mathbf{h}_0^T\mathbf{y} = \mathbf{g}_0^TL_0\mathbf{y} = \mathbf{g}_0^T\mathbf{e}_{0b}$, and similarly, defining $\mathbf{g}_1 = L_1^{-T}\mathbf{h}_1$, we find $\hat{x}_1 = \mathbf{g}_1^T\mathbf{e}_{1b}$. Thus,

$$\mathbf{g}_1 = L_1^{-T}\mathbf{h}_1\,, \quad \mathbf{g}_0 = L_0^{-T}\mathbf{h}_0 \tag{16.15.60}$$

and

$$\hat{x}_1 = \mathbf{g}_1^T\mathbf{e}_{1b}\,, \quad \hat{x}_0 = \mathbf{g}_0^T\mathbf{e}_{0b} \tag{16.15.61}$$

Finally, the updating equation (16.15.16) for the direct-form weights translates into an updating equation for the lattice weights:

$$\mathbf{g}_1 = L_1^{-T}\mathbf{h}_1 = L_1^{-T}(\mathbf{h}_0 + e_0\mathbf{k}_1) = L^{-T}L_0^{-T}\mathbf{h}_0 + e_0L_1^{-T}\mathbf{k}_1$$

where we used the factorization (16.15.54) for the first term. Using $R_1^{-1} = L_1^TD_{1b}^{-1}L_1$, we find for the second term $L_1^{-T}\mathbf{k}_1 = L_1^{-T}R_1^{-1}\mathbf{y} = D_{1b}^{-1}L_1\mathbf{y} = D_{1b}^{-1}\mathbf{e}_{1b}$. Therefore,

$$\mathbf{g}_1 = L^{-T}\mathbf{g}_0 + e_0D_{1b}^{-1}\mathbf{e}_{1b} \tag{16.15.62}$$

Extracting the last elements we obtain Eq. (16.15.45).

16.16 RLS Adaptive Filters

The LMS and gradient lattice adaptation algorithms, based on the steepest descent method, provide a gradual, iterative, minimization of the performance index. The adaptive weights are not optimal at each time instant, but only after convergence. In this section, we discuss *recursive least-squares* (RLS) adaptation algorithms that are based on the *exact minimization* of least-squares criteria. The filter weights are optimal at each time instant *n*.

Adaptive RLS algorithms are the time-recursive analogs of the block processing methods of linear prediction and FIR Wiener filtering that we discussed in Sections 12.12 and 12.14. They may be used, in place of LMS, in any adaptive filtering application. Because of their fast convergence they have been proposed for use in fast start-up channel equalizers [1448–1451]. They are also routinely used in real-time system identification applications. Their main disadvantage is that they require a fair amount of computation, $O(M^2)$ operations per time update. In biomedical applications, they can be easily implemented on minicomputers. In other applications, such as the equalization of rapidly varying channels or adaptive arrays [1355,1453–1455], they may be too costly for implementation.

The fast reformulations of RLS algorithms, such as the RLSL, FK, FAEST, and FTF, have $O(M)$ computational complexity. The fast RLS algorithms combine the best of the LMS and RLS, namely, the computational efficiency of the former and the fast convergence of the latter. Among the fast RLS algorithms, the RLS lattice has better numerical stability properties than the direct-form versions.

We start with the RLS formulation of the FIR Wiener filtering problem. The estimation criterion, $\mathcal{E} = E[e(n)^2] = \min$, is replaced with a least-squares weighted time-average that includes all estimation errors from the initial time instant to the current time n, that is, $e(k)$, $k = 0, 1, \dots, n$:

$$\mathcal{E}_n = \sum_{k=0}^{n} e^2(k) = \min \tag{16.16.1}$$

where

$$e(k) = x(k) = \hat{x}(k)$$

and $\hat{x}(k)$ is the estimate of $x(k)$ produced by the order-M Wiener filter

$$\hat{x}(k) = \sum_{m=0}^{M} h_m y_{k-m} = [h_0, h_1, \dots, h_M]\begin{bmatrix} y_k \\ y_{k-1} \\ \vdots \\ y_{k-M} \end{bmatrix} = \mathbf{h}^T \mathbf{y}(k)$$

Note that in adaptive array problems, $\mathbf{y}(k)$ represents the vector of measurements at the array elements, namely, $\mathbf{y}(k) = [y_0(k), y_1(k), \dots, y_M(k)]$. To better track possible non-stationarities in the signals, the performance index may be modified by introducing exponential weighting

$$\mathcal{E}_n = \sum_{k=0}^{n} \lambda^{n-k} e^2(k) = e^2(n) + \lambda e^2(n-1) + \lambda^2 e^2(n-2) + \cdots + \lambda^n e^2(0) \tag{16.16.2}$$

where the *forgetting factor* λ is positive and less than one. This performance index emphasizes the most recent observations and exponentially ignores the older ones. We will base our discussion on this criterion. Setting the derivative with respect to $\mathbf{h}$ to zero, we find the least-square analogs of the *orthogonality equations*

$$\frac{\partial \mathcal{E}_n}{\partial \mathbf{h}} = -2 \sum_{k=0}^{n} \lambda^{n-k} e(k) \mathbf{y}(k) = 0$$

which may be cast in a *normal equation* form

$$\sum_{k=0}^{n} \lambda^{n-k} [x(k) - \mathbf{h}^T \mathbf{y}(k)] \mathbf{y}(k) = 0, \quad \text{or,}$$

$$\left[\sum_{k=0}^{n} \lambda^{n-k} \mathbf{y}(k) \mathbf{y}(k)^T\right] \mathbf{h} = \sum_{k=0}^{n} \lambda^{n-k} x(k) \mathbf{y}(k)$$

Defining the quantities

$$\begin{aligned} R(n) &= \sum_{k=0}^{n} \lambda^{n-k} \mathbf{y}(k) \mathbf{y}(k)^T \\ \mathbf{r}(n) &= \sum_{k=0}^{n} \lambda^{n-k} x(k) \mathbf{y}(k) \end{aligned} \tag{16.16.3}$$

we write the normal equations as $R(n)\mathbf{h} = \mathbf{r}(n)$, with solution $\mathbf{h} = R(n)^{-1}\mathbf{r}(n)$. Note that the n-dependence of $R(n)$ and $\mathbf{r}(n)$ makes $\mathbf{h}$ depend on n; we shall write, therefore,

$$\mathbf{h}(n) = R(n)^{-1} \mathbf{r}(n) \tag{16.16.4}$$

These are the least-squares analogs of the ordinary Wiener solution, with $R(n)$ and $\mathbf{r}(n)$ playing the role of the covariance matrix $R = E[\mathbf{y}(n)\mathbf{y}^T(n)]$ and cross-correlation vector $\mathbf{r} = E[x(n)\mathbf{y}(n)]$. These quantities satisfy the rank-one updating properties

$$R(n) = \lambda R(n-1) + \mathbf{y}(n)\mathbf{y}(n)^T \tag{16.16.5}$$

$$\mathbf{r}(n) = \lambda \mathbf{r}(n-1) + x(n)\mathbf{y}(n) \tag{16.16.6}$$

Thus, the general results of the previous section can be applied. We have the correspondences:

$\mathbf{y}$	→	$\mathbf{y}(n)$	x	→	$x(n)$
R_1	→	$R(n)$	R_0	→	$\lambda R(n-1)$
P_1	→	$P(n) = R(n)^{-1}$	P_0	→	$\lambda^{-1} P(n-1) = \lambda^{-1} R(n-1)^{-1}$
$\mathbf{r}_1$	→	$\mathbf{r}(n)$	$\mathbf{r}_0$	→	$\lambda \mathbf{r}(n-1)$
$\mathbf{h}_1$	→	$\mathbf{h}(n) = R(n)^{-1}\mathbf{r}(n)$	$\mathbf{h}_0$	→	$\mathbf{h}(n-1) = R(n-1)^{-1}\mathbf{r}(n)$
$\hat{x}_1$	→	$\hat{x}(n) = \mathbf{h}(n)^T \mathbf{y}(n)$	$\hat{x}_0$	→	$\hat{x}(n/n-1) = \mathbf{h}(n-1)^T \mathbf{y}(n)$
e_1	→	$e(n) = x(n) - \hat{x}(n)$	e_0	→	$e(n/n-1) = x(n) - \hat{x}(n/n-1)$
$\mathbf{k}_1$	→	$\mathbf{k}(n) = R(n)^{-1}\mathbf{y}(n)$	$\mathbf{k}_0$	→	$\mathbf{k}(n/n-1) = \lambda^{-1} R(n-1)^{-1} \mathbf{y}(n)$
ν	→	$\nu(n) = \mathbf{k}(n/n-1)^T \mathbf{y}(n)$	μ	→	$\mu(n) = 1/\big(1 + \nu(n)\big)$

We used the notation $\hat{x}(n/n-1)$, $e(n/n-1)$, and $\mathbf{k}(n/n-1)$ to denote the *a priori* estimate, estimation error, and Kalman gain. Note that $R_0, \mathbf{r}_0$ are the quantities $R(n-1)$, $\mathbf{r}(n-1)$ scaled by the forgetting factor λ. In the a priori solution $\mathbf{h}_0 = R_0^{-1}\mathbf{r}_0$, the factors λ cancel to give $\big[\lambda R(n-1)\big]^{-1}\big[\lambda \mathbf{r}(n-1)\big] = R(n-1)^{-1}\mathbf{r}(n-1) = \mathbf{h}(n-1)$. Thus, the a priori Wiener solution is the solution at the *previous* time instant $n-1$. With the above correspondences, the conventional RLS algorithm listed in the previous section becomes

1. $\mathbf{k}(n/n-1) = \lambda^{-1} P(n-1) \mathbf{y}(n)$

2. $\nu(n) = \mathbf{k}(n/n-1)^T \mathbf{y}(n)\,, \quad \mu(n) = \dfrac{1}{1 + \nu(n)}$

3. $\mathbf{k}(n)= \mu(n)\mathbf{k}(n/n-1)$
4. $P(n)= \lambda^{-1}P(n-1)-\mathbf{k}(n)\mathbf{k}(n/n-1)^T$
5. $\hat{x}(n/n-1)= \mathbf{h}(n-1)^T\mathbf{y}(n)\,, \quad e(n/n-1)= x(n)-\hat{x}(n/n-1)$
6. $e(n)= \mu(n)e(n/n-1)\,, \quad \hat{x}(n)= x(n)-e(n)$
7. $\mathbf{h}(n)= \mathbf{h}(n-1)+e(n/n-1)\mathbf{k}(n)$

The algorithm may be initialized in time by taking $R(-1)= 0$, which would imply $P(-1)= \infty$. Instead, we may use $P(-1)= \delta^{-1}I$, where δ is a very small number, and I the identity matrix. The algorithm is quite insensitive to the choice of δ. Typical values are $\delta = 0.1$, or $\delta = 0.01$.

The function **rls** is an implementation of the algorithm. Because the algorithm can also be used in array problems, we have designed the function so that its inputs are the old weights $\mathbf{h}(n-1)$, the current sample $x(n)$, and the entire data vector $\mathbf{y}(n)$ (in time series problems only the current time sample y_n is needed, the past samples y_{n-i}, $i = 1, 2, \dots, M$ being stored in the tapped delay line). The outputs of the function are $\mathbf{h}(n)$, $\hat{x}(n)$, and $e(n)$. A simulation example will be presented in the next section.

The term *Kalman gain* arises by interpreting $\mathbf{h}(n)= \mathbf{h}(n-1)+e(n/n-1)\mathbf{k}(n)$ as a Kalman predictor/corrector algorithm, where the first term $\mathbf{h}(n-1)$ is a prediction of the weight $\mathbf{h}(n)$ based on the past, $e(n/n-1)= x(n)-\mathbf{h}(n-1)^T\mathbf{y}(n)$ is the tentative estimation error made on the basis of the prediction $\mathbf{h}(n-1)$, and the second term $e(n/n-1)\mathbf{k}(n)$ is the correction of the prediction. The fast convergence properties of the algorithm can be understood by making the replacement $\mathbf{k}(n)= R(n)^{-1}\mathbf{y}(n)$ in the update equation

$$\mathbf{h}(n)= \mathbf{h}(n-1)+R(n)^{-1}\mathbf{y}(n)e(n/n-1) \tag{16.16.7}$$

It differs from the LMS algorithm by the presence of $R(n)^{-1}$ in the weight update term. Because $R(n)$ is an estimate of the covariance matrix $R = E[\mathbf{y}(n)\mathbf{y}(n)^T]$, the presence of $R(n)^{-1}$ makes the RLS algorithm behave like Newton's method, hence its fast convergence properties [1456,1457]. Another important conceptual difference with the LMS algorithm is that in the RLS algorithm the filters $\mathbf{h}(n)$ and $\mathbf{h}(n-1)$ are the *exact* Wiener solutions of two different minimization criteria; namely, $\mathcal{E}_n = \min$ and $\mathcal{E}_{n-1} = \min$, whereas in the LMS algorithm they are successive gradient-descent approximations to the optimum solution.

The role of the forgetting factor λ may be understood qualitatively, by considering the quantity

$$n_\lambda = \frac{\sum_{n=0}^{\infty} n\lambda^n}{\sum_{n=0}^{\infty}\lambda^n} = \frac{\lambda}{1-\lambda}$$

to be a *measure* of the effective memory of the performance index $\mathcal{E}_n$. Smaller λs correspond to shorter memory n_λ, and can track better the non-stationary changes of the underlying signals. The memory n_λ of the performance index should be as short as the effective duration of the non-stationary segments, but not shorter because the performance index will not be taking full advantage of all the available samples (which could extend over the entire non-stationary segment); as a result, the computed weights $\mathbf{h}(n)$ will exhibit more noisy behavior. In particular, if the signals are stationary, the best value of λ is unity.

In Sec. 16.12, we considered the adaptive implementation of eigenvector methods based on an LMS gradient-projection method. Adaptive eigenvector methods can also be formulated based on the rank-one updating property (16.16.5). For example, one may use standard numerical methods for the rank-one updating of the entire eigenproblem of $R(n)$ [1166,1458,1459].

If one is interested only in a few largest or smallest eigenvalues and corresponding eigenvectors, one can use the more efficient power method or inverse power method and their generalizations, such as the simultaneous and subspace iterations, or Lanczos methods, which are essentially the subspace iteration improved by Rayleigh-Ritz methods [1246,1460].

The basic procedure for making these numerical methods adaptive is as follows [1461–1467]. The power method generates the maximum eigenvector by the iteration $\mathbf{e}(n)= R\mathbf{e}(n-1)$, followed by normalization of $\mathbf{e}(n)$ to unit norm. Similarly, the minimum eigenvector may be generated by the inverse power iteration $\mathbf{e}(n)= R^{-1}\mathbf{e}(n-1)$. Because R and R^{-1} are not known, they may be replaced by their estimates $R(n)$ and $P(n)= R(n)^{-1}$, which are being updated from one time instant to the next by Eq. (16.16.5) or by step 4 of the RLS algorithm, so that one has $\mathbf{e}(n)= R(n)\mathbf{e}(n-1)$ for the power iteration, or $\mathbf{e}(n)= P(n)\mathbf{e}(n-1)$ for the inverse power case.

This can be generalized to the simultaneous iteration case. For example, to generate adaptively the K minimum eigenvectors spanning the noise subspace one starts at each iteration n with K mutually orthonormalized vectors $\mathbf{e}_i(n-1)$, $i = 0, 1, \dots, K-1$. Each is subjected to the inverse power iteration $\mathbf{e}_i(n)= P(n)\mathbf{e}_i(n-1)$ and finally, the K updated vectors $\mathbf{e}_i(n)$ are mutually orthonormalized using the Gram-Schmidt or modified Gram-Schmidt procedure for vectors. Similar simultaneous iteration methods can also be applied to the gradient-projection method of Sec. 16.12. The main limitation of applying the simultaneous iteration methods is that one must know in advance the dimension K of the noise subspace.

16.17 Fast RLS Filters

In this section, we present fast RLS algorithms based on a direct-form realization [1422–1424,1436–1445,1468–1477]. Fast RLS lattice filters are discussed in the next section. The fast direct-form RLS algorithms make use of the forward and backward predictors. The subblock decompositions of the $(M+1)$-dimensional data vector $\mathbf{y}(n)$ are

$$\mathbf{y}(n)= \begin{bmatrix} y_n \\ y_{n-1} \\ \vdots \\ y_{n-M} \end{bmatrix} = \begin{bmatrix} \bar{\mathbf{y}}(n) \\ y_{n-M} \end{bmatrix} = \begin{bmatrix} y_n \\ \tilde{\mathbf{y}}(n) \end{bmatrix} \tag{16.17.1}$$

Therefore, the two M-dimensional parts of $\mathbf{y}(n)$ are

$$\bar{\mathbf{y}}(n)=\begin{bmatrix} y_n \\ y_{n-1} \\ \vdots \\ y_{n-M+1} \end{bmatrix}, \quad \tilde{\mathbf{y}}(n)=\begin{bmatrix} y_{n-1} \\ y_{n-2} \\ \vdots \\ y_{n-M} \end{bmatrix} \tag{16.17.2}$$

The covariance matrices of these subvectors will be

$$\bar{R}(n)=\sum_{k=0}^{n}\lambda^{n-k}\bar{\mathbf{y}}(k)\bar{\mathbf{y}}(k)^T\,, \quad \tilde{R}(n)=\sum_{k=0}^{n}\lambda^{n-k}\tilde{\mathbf{y}}(k)\tilde{\mathbf{y}}(k)^T \tag{16.17.3}$$

The definitions (16.17.2) imply the *shift-invariance* property

$$\tilde{\mathbf{y}}(n+1)=\bar{\mathbf{y}}(n) \tag{16.17.4}$$

Using this property, we find

$$\begin{aligned}\tilde{R}(n+1) &= \sum_{k=0}^{n+1}\lambda^{n+1-k}\tilde{\mathbf{y}}(k)\tilde{\mathbf{y}}(k)^T = \sum_{k=-1}^{n}\lambda^{n-k}\tilde{\mathbf{y}}(k+1)\tilde{\mathbf{y}}(k+1)^T \\ &= \sum_{k=-1}^{n}\lambda^{n-k}\bar{\mathbf{y}}(k)\bar{\mathbf{y}}(k)^T = \bar{R}(n)+\lambda^{n+1}\bar{\mathbf{y}}(-1)\bar{\mathbf{y}}(-1)^T\end{aligned}$$

If we make the *prewindowing* assumption that $\bar{\mathbf{y}}(-1)=0$, we obtain the shift-invariance property for the covariance matrices

$$\tilde{R}(n+1)=\bar{R}(n) \tag{16.17.5}$$

Before we use the shift-invariance properties, we make some additional correspondences from the previous section:

$\bar{\mathbf{y}}$	→	$\bar{\mathbf{y}}(n)$
$\tilde{\mathbf{y}}$	→	$\tilde{\mathbf{y}}(n)$
$R_1\mathbf{a}_1 = E_{1a}\mathbf{u}$	→	$R(n)\mathbf{a}(n)=E^+(n)\mathbf{u}$
$R_1\mathbf{b}_1 = E_{1b}\mathbf{v}$	→	$R(n)\mathbf{b}(n)=E^-(n)\mathbf{v}$
$R_0\mathbf{a}_0 = E_{0a}\mathbf{u}$	→	$\lambda R(n-1)\mathbf{a}(n-1)=\lambda E^+(n-1)\mathbf{u}$
$R_0\mathbf{b}_0 = E_{0b}\mathbf{v}$	→	$\lambda R(n-1)\mathbf{b}(n-1)=\lambda E^-(n-1)\mathbf{v}$
$e_{1a} = \mathbf{a}_1^T\mathbf{y}$	→	$e^+(n)=\mathbf{a}(n)^T\mathbf{y}(n)$
$e_{1b} = \mathbf{b}_1^T\mathbf{y}$	→	$e^-(n)=\mathbf{b}(n)^T\mathbf{y}(n)$
$e_{0a} = \mathbf{a}_0^T\mathbf{y}$	→	$e^+(n/n-1)=\mathbf{a}(n-1)^T\mathbf{y}(n)$
$e_{0b} = \mathbf{b}_0^T\mathbf{y}$	→	$e^-(n/n-1)=\mathbf{b}(n-1)^T\mathbf{y}(n)$
$E_{1a} = E_{0a} + e_{1a}e_{0a}$	→	$E^+(n)=\lambda E^+(n-1)+e^+(n)e^+(n/n-1)$
$E_{1b} = E_{0b} + e_{1b}e_{0b}$	→	$E^-(n)=\lambda E^-(n-1)+e^-(n)e^-(n/n-1)$
$\tilde{\mathbf{k}}_1 = \tilde{R}_1^{-1}\tilde{\mathbf{y}}$	→	$\tilde{\mathbf{k}}(n)=\tilde{R}(n)^{-1}\tilde{\mathbf{y}}(n)$
$\bar{\mathbf{k}}_1 = \bar{R}_1^{-1}\bar{\mathbf{y}}$	→	$\bar{\mathbf{k}}(n)=\bar{R}(n)^{-1}\bar{\mathbf{y}}(n)$
$\tilde{\mathbf{k}}_0 = \tilde{R}_0^{-1}\tilde{\mathbf{y}}$	→	$\tilde{\mathbf{k}}(n/n-1)=\lambda^{-1}\tilde{R}(n-1)^{-1}\tilde{\mathbf{y}}(n)$
$\bar{\mathbf{k}}_0 = \bar{R}_0^{-1}\bar{\mathbf{y}}$	→	$\bar{\mathbf{k}}(n/n-1)=\lambda^{-1}\bar{R}(n-1)^{-1}\bar{\mathbf{y}}(n)$
$\tilde{\nu} = \tilde{\mathbf{k}}_0^T\tilde{\mathbf{y}}$	→	$\tilde{\nu}(n)=\tilde{\mathbf{k}}(n/n-1)^T\tilde{\mathbf{y}}(n)$
$\bar{\nu} = \bar{\mathbf{k}}_0^T\bar{\mathbf{y}}$	→	$\bar{\nu}(n)=\bar{\mathbf{k}}(n/n-1)^T\bar{\mathbf{y}}(n)$
$\tilde{\mu} = 1/(1+\tilde{\nu})$	→	$\tilde{\mu}(n)=1/\big(1+\tilde{\nu}(n)\big)$
$\bar{\mu} = 1/(1+\bar{\nu})$	→	$\bar{\mu}(n)=1/\big(1+\bar{\nu}(n)\big)$

We have used the superscripts ± to indicate the forward and backward quantities. Again, note the cancellation of the factors λ from the a priori normal equations, which implies that the a priori predictors are the predictors of the previous time instant; that is, $\mathbf{a}_0 \to \mathbf{a}(n-1)$ and $\mathbf{b}_0 \to \mathbf{b}(n-1)$.

Using the shift-invariance properties (16.17.4) and (16.17.5), we find that all the tilde quantities at the next time instant $n+1$ are equal to the bar quantities at the present instant n; for example,

$$\tilde{\mathbf{k}}(n+1)=\tilde{R}(n+1)^{-1}\tilde{\mathbf{y}}(n+1)=\bar{R}(n)^{-1}\bar{\mathbf{y}}(n)=\bar{\mathbf{k}}(n)$$

Similarly,

$$\tilde{\mathbf{k}}(n+1/n)=\lambda^{-1}\tilde{R}(n)^{-1}\tilde{\mathbf{y}}(n+1)=\lambda^{-1}\bar{R}(n-1)^{-1}\bar{\mathbf{y}}(n)=\bar{\mathbf{k}}(n/n-1)$$

and for the likelihood variables

$$\tilde{\nu}(n+1)=\tilde{\mathbf{k}}(n+1/n)^T\tilde{\mathbf{y}}(n+1)=\bar{\mathbf{k}}(n/n-1)^T\bar{\mathbf{y}}(n)=\bar{\nu}(n)$$

and similarly for the μs. We summarize:

$$\begin{aligned}\tilde{\mathbf{k}}(n+1) &= \bar{\mathbf{k}}(n)\,, \quad \tilde{\mathbf{k}}(n+1/n)=\bar{\mathbf{k}}(n/n-1) \\ \tilde{\nu}(n+1) &= \bar{\nu}(n)\,, \quad \tilde{\mu}(n+1)=\bar{\mu}(n)\end{aligned} \tag{16.17.6}$$

These equations can be added at the ends of the computational sequences of the previous section to complete the computational cycle at each time instant. In the present notation, the complete *fast Kalman algorithm* [1422,1423] is:

0. At time n, we have available the quantities $\mathbf{h}(n-1)$, $\mathbf{a}(n-1)$, $\mathbf{b}(n-1)$, $\tilde{\mathbf{k}}(n)$, $E^+(n-1)$, $x(n)$, and $\mathbf{y}(n)$

1 . $e^+(n/n-1)=\mathbf{a}(n-1)^T\mathbf{y}(n)$

2. $\mathbf{a}(n)=\mathbf{a}(n-1)-e^+(n/n-1)\begin{bmatrix} 0 \\ \tilde{\mathbf{k}}(n) \end{bmatrix}$

3. $e^+(n)=\mathbf{a}(n)^T\mathbf{y}(n)$

4. $E^+(n)=\lambda E^+(n-1)+e^+(n)e^+(n/n-1)$

5. Compute the first element of $\mathbf{k}(n)$, $k_0(n)=\dfrac{e^+(n)}{E^+(n)}$

6. $\mathbf{k}(n)=\begin{bmatrix} 0 \\ \tilde{\mathbf{k}}(n) \end{bmatrix}+k_0(n)\mathbf{a}(n)$, extract the last element of $\mathbf{k}(n)$, $k_M(n)$

7. $e^-(n/n-1)=\mathbf{b}(n-1)^T\mathbf{y}(n)$

8. $\mathbf{b}(n)=\dfrac{\mathbf{b}(n-1)-e^-(n/n-1)\mathbf{k}(n)}{1-e^-(n/n-1)k_M(n)}$

9. $\begin{bmatrix} \bar{\mathbf{k}}(n) \\ 0 \end{bmatrix}=\mathbf{k}(n)-k_M(n)\mathbf{b}(n)$

10. $\hat{x}(n/n-1)=\mathbf{h}(n-1)^T\mathbf{y}(n)\,,\quad e(n/n-1)=x(n)-\hat{x}(n/n-1)$

11. $\mathbf{h}(n)=\mathbf{h}(n-1)+e(n/n-1)\mathbf{k}(n)$

12. $\hat{x}(n)=\mathbf{h}(n)^T\mathbf{y}(n)\,,\quad e(n)=x(n)-\hat{x}(n)$

13. $\tilde{\mathbf{k}}(n+1)=\bar{\mathbf{k}}(n)$

14. Go to the next time instant, $n\to n+1$

The first and last entries of the a posteriori Kalman gain vector $\mathbf{k}(n)$ were denoted by $k_0(n)$ and $k_M(n)$, that is, $\mathbf{k}(n)=[k_0(n),k_1(n),\dots,k_M(n)]^T$. Similarly, we obtain the complete *FAEST algorithm* [1424]:

0. At time n, we have available the quantities $\mathbf{h}(n-1)$, $\mathbf{a}(n-1)$, $\mathbf{b}(n-1)$, $\tilde{\mathbf{k}}(n/n-1)$, $\tilde{\nu}(n)$, $E^{\pm}(n-1)$, $x(n)$, and $\mathbf{y}(n)$

1 . $e^+(n/n-1)=\mathbf{a}(n-1)^T\mathbf{y}(n)$

2. $e^+(n)=e^+(n/n-1)/\big(1+\tilde{\nu}(n)\big)=\tilde{\mu}(n)e^+(n/n-1)$

3. Compute the first element of $\mathbf{k}(n/n-1)$, $k_0(n/n-1)=\dfrac{e^+(n/n-1)}{\lambda E^+(n-1)}$

4. $E^+(n)=\lambda E^+(n-1)+e^+(n)e^+(n/n-1)$

5. $\mathbf{k}(n/n-1)=\begin{bmatrix}0\\ \tilde{\mathbf{k}}(n/n-1)\end{bmatrix}+k_0(n/n-1)\mathbf{a}(n-1)$

6. Extract the last element of $\mathbf{k}(n/n-1)$, $k_M(n/n-1)$

7. $e^-(n/n-1)=k_M(n/n-1)\big[\lambda E^-(n-1)\big]$

8. $\begin{bmatrix}\bar{\mathbf{k}}(n/n-1)\\ 0\end{bmatrix}=\mathbf{k}(n/n-1)-k_M(n/n-1)\mathbf{b}(n-1)$

9. $\nu(n)=\tilde{\nu}(n)+e^+(n/n-1)k_0(n/n-1)\,,\quad \bar{\nu}(n)=\nu(n)-e^-(n/n-1)k_M(n/n-1)$

10. $e^-(n)=e^-(n/n-1)/\big(1+\bar{\nu}(n)\big)=\bar{\mu}(n)e^-(n/n-1)$

11. $E^-(n)=\lambda E^-(n-1)+e^-(n)e^-(n/n-1)$

12. $\mathbf{a}(n)=\mathbf{a}(n-1)-e^+(n)\begin{bmatrix}0\\ \tilde{\mathbf{k}}(n/n-1)\end{bmatrix}$

13. $\mathbf{b}(n)=\mathbf{b}(n-1)-e^-(n)\begin{bmatrix}\bar{\mathbf{k}}(n/n-1)\\ 0\end{bmatrix}$

14. $\hat{x}(n/n-1)=\mathbf{h}(n-1)^T\mathbf{y}(n)\,,\quad e(n/n-1)=x(n)-\hat{x}(n/n-1)$

15. $e(n)=e(n/n-1)/\big(1+\nu(n)\big)=\mu(n)e(n/n-1)\,,\quad \hat{x}(n)=x(n)-e(n)$

16. $\mathbf{h}(n)=\mathbf{h}(n-1)+e(n)\mathbf{k}(n/n-1)$

17. $\tilde{\mathbf{k}}(n+1/n)=\bar{\mathbf{k}}(n)\,,\quad \tilde{\nu}(n+1)=\bar{\nu}(n)$

19. Go to the next time instant, $n\to n+1$

The algorithm is initialized in time by clearing the tapped delay line of the filter and setting $\mathbf{h}(-1)=0$, $\mathbf{a}(-1)=\mathbf{u}=[1,\mathbf{0}^T]^T$, $\mathbf{b}(-1)=\mathbf{v}=[\mathbf{0}^T,1]^T$, $\tilde{\mathbf{k}}(0/-1)=0$, $\tilde{\nu}(0)=0$, and $E^{\pm}(-1)=\delta$, where δ is a small constant. Exact initialization procedures have been discussed in [1426]. The *FTF algorithm* [1426] is obtained by replacing step 9 by the following:

$$\mu(n)=\tilde{\mu}(n)\,\frac{\lambda E^+(n-1)}{E^+(n)}\,,\quad \bar{\mu}(n)=\frac{\mu(n)}{1-e^-(n/n-1)k_M(n/n-1)\mu(n)}\qquad\text{(FTF)}$$

The function **faest** is an implementation of the FAEST algorithm. The function transforms an input pair of samples $\{x,y\}$ into an output pair $\{\hat{x},e\}$, updates the tapped delay line of the filter, and updates the filter $\mathbf{h}(n)$.

Next, we present a simulation example comparing the FAEST and LMS algorithms. The example is the same as that discussed in Sec. 16.13 and defined theoretically by Eqs. (16.13.37) and (16.13.38). Fig. 16.17.1 shows two of the adaptive weights, $h_1(n)$ and $h_2(n)$, adapted by FAEST and LMS. The weights are converging to their theoretical values of $h_1=1.5$ and $h_2=-2$. The RLS parameters were $\lambda=1$ and $\delta=0.01$; the LMS parameter was $\mu=0.01$.

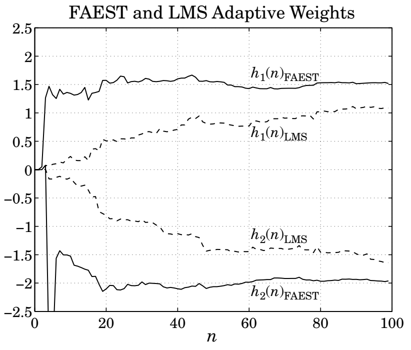


Fig. 16.17.1 Comparison of FAEST and LMS adaptive weights.

16.18 RLS Lattice Filters

The fast direct-form RLS filters were fixed-order filters. By contrast, the RLS lattice algorithms [1436–1445], for each time instant n, do a recursion in the order, $p=0,1,\dots,M$. Therefore, it is necessary to indicate the order p by using an extra index in all the quantities of the past two sections. For example, the order-p data vector and its bar and tilde parts will be denoted by

$$\mathbf{y}_p(n)=\begin{bmatrix} y_n \\ y_{n-1} \\ \vdots \\ y_{n-p}\end{bmatrix}, \quad \bar{\mathbf{y}}_p(n)=\begin{bmatrix} y_n \\ y_{n-1} \\ \vdots \\ y_{n-p+1}\end{bmatrix}, \quad \tilde{\mathbf{y}}_p(n)=\begin{bmatrix} y_{n-1} \\ y_{n-2} \\ \vdots \\ y_{n-p}\end{bmatrix} \tag{16.18.1}$$

Therefore, we have

$$\bar{\mathbf{y}}_p(n)=\mathbf{y}_{p-1}(n)\,, \quad \tilde{\mathbf{y}}_p(n)=\bar{\mathbf{y}}_p(n-1)=\mathbf{y}_{p-1}(n-1) \tag{16.18.2}$$

Similarly, the covariance matrices will be

$$\bar{R}_p(n)=R_{p-1}(n)\,, \quad \tilde{R}_p(n)=R_{p-1}(n-1) \tag{16.18.3}$$

The order-p predictors will be denoted by $\mathbf{a}_p(n)$ and $\mathbf{b}_p(n)$, with error signals $e_p^+(n)=\mathbf{a}_p(n)^T\mathbf{y}_p(n)$ and $e_p^-(n)=\mathbf{b}_p(n)^T\mathbf{y}_p(n)$ The corresponding mean-square errors will be denoted by $E_p^\pm(n)$. Similarly, the a priori estimation errors are denoted by $e_p^+(n/n-1)=\mathbf{a}_p(n-1)^T\mathbf{y}_p(n)$ and $e_p^-(n/n-1)=\mathbf{b}_p(n-1)^T\mathbf{y}_p(n)$. Using Eq. (16.18.3), we find the following correspondences between the order-$(p-1)$ and order-p problems:

$$\begin{array}{llllllllll}
\bar{R}_1 & \to & R_{p-1}(n), & \bar{\mathbf{a}}_1 & \to & \mathbf{a}_{p-1}(n), & \bar{E}_{1a} & \to & E_{p-1}^+(n) \\
\bar{R}_0 & \to & \lambda R_{p-1}(n-1), & \bar{\mathbf{a}}_0 & \to & \mathbf{a}_{p-1}(n-1), & \bar{E}_{0a} & \to & \lambda E_{p-1}^+(n-1) \\
\tilde{R}_1 & \to & R_{p-1}(n-1), & \tilde{\mathbf{b}}_1 & \to & \mathbf{b}_{p-1}(n-1), & \tilde{E}_{1b} & \to & E_{p-1}^-(n-1) \\
\tilde{R}_0 & \to & \lambda R_{p-1}(n-2), & \tilde{\mathbf{b}}_0 & \to & \mathbf{b}_{p-1}(n-2), & \tilde{E}_{0b} & \to & \lambda E_{p-1}^-(n-1)
\end{array}$$

$$\begin{array}{rcl}
\bar{e}_{1a}=\bar{\mathbf{a}}_1^T\bar{\mathbf{y}} & \to & e_{p-1}^+(n)=\mathbf{a}_{p-1}(n)^T\mathbf{y}_{p-1}(n) \\
\tilde{e}_{1b}=\tilde{\mathbf{b}}_1^T\tilde{\mathbf{y}} & \to & e_{p-1}^-(n-1)=\mathbf{b}_{p-1}(n-1)^T\mathbf{y}_{p-1}(n-1) \\
\bar{e}_{0a}=\bar{\mathbf{a}}_0^T\bar{\mathbf{y}} & \to & e_{p-1}^+(n/n-1)=\mathbf{a}_{p-1}(n-1)^T\mathbf{y}_{p-1}(n) \\
\tilde{e}_{0b}=\tilde{\mathbf{b}}_0^T\tilde{\mathbf{y}} & \to & e_{p-1}^-(n-1/n-2)=\mathbf{b}_{p-1}(n-2)^T\mathbf{y}_{p-1}(n-1)
\end{array}$$

$$\begin{array}{rcl}
\gamma_{1a} & \to & \gamma_p^+(n) \\
\gamma_{0a} & \to & \gamma_p^+(n-1) \\
\gamma_{1b} & \to & \gamma_p^-(n) \\
\gamma_{0b} & \to & \gamma_p^-(n-1)
\end{array}$$

$$\begin{array}{rcl}
e_{1a}=\bar{e}_{1a}-\gamma_{1b}\tilde{e}_{1b} & \to & e_p^+(n)=e_{p-1}^+(n)-\gamma_p^-(n)e_{p-1}^-(n-1) \\
e_{1b}=\tilde{e}_{1b}-\gamma_{1a}\bar{e}_{1a} & \to & e_p^-(n)=e_{p-1}^-(n-1)-\gamma_p^+(n)e_{p-1}^+(n) \\
e_{0a}=\bar{e}_{0a}-\gamma_{0b}\tilde{e}_{0b} & \to & e_p^+(n/n-1)=e_{p-1}^+(n/n-1)-\gamma_p^-(n-1)e_{p-1}^-(n-1/n-2) \\
e_{0b}=\tilde{e}_{0b}-\gamma_{0a}\bar{e}_{0a} & \to & e_p^-(n/n-1)=e_{p-1}^-(n-1/n-2)-\gamma_p^+(n-1)e_{p-1}^+(n/n-1)
\end{array}$$

$$\mathbf{a}_1=\begin{bmatrix}\bar{\mathbf{a}}_1\\0\end{bmatrix}-\gamma_{1b}\begin{bmatrix}0\\\tilde{\mathbf{b}}_1\end{bmatrix} \quad \to \quad \mathbf{a}_p(n)=\begin{bmatrix}\mathbf{a}_{p-1}(n)\\0\end{bmatrix}-\gamma_p^-(n)\begin{bmatrix}0\\\mathbf{b}_{p-1}(n-1)\end{bmatrix}$$

$$\mathbf{b}_1=\begin{bmatrix}0\\\tilde{\mathbf{b}}_1\end{bmatrix}-\gamma_{1a}\begin{bmatrix}\bar{\mathbf{a}}_1\\0\end{bmatrix} \quad \to \quad \mathbf{b}_p(n)=\begin{bmatrix}0\\\mathbf{b}_{p-1}(n-1)\end{bmatrix}-\gamma_p^+(n)\begin{bmatrix}\mathbf{a}_{p-1}(n)\\0\end{bmatrix}$$

$$\mathbf{a}_0=\begin{bmatrix}\bar{\mathbf{a}}_0\\0\end{bmatrix}-\gamma_{0b}\begin{bmatrix}0\\\tilde{\mathbf{b}}_0\end{bmatrix} \quad \to \quad \mathbf{a}_p(n-1)=\begin{bmatrix}\mathbf{a}_{p-1}(n-1)\\0\end{bmatrix}-\gamma_p^-(n-1)\begin{bmatrix}0\\\mathbf{b}_{p-1}(n-2)\end{bmatrix}$$

$$\mathbf{b}_0=\begin{bmatrix}0\\\tilde{\mathbf{b}}_0\end{bmatrix}-\gamma_{0a}\begin{bmatrix}\bar{\mathbf{a}}_0\\0\end{bmatrix} \quad \to \quad \mathbf{b}_p(n-1)=\begin{bmatrix}0\\\mathbf{b}_{p-1}(n-2)\end{bmatrix}-\gamma_p^+(n-1)\begin{bmatrix}\mathbf{a}_{p-1}(n-1)\\0\end{bmatrix}$$

$$\begin{array}{lcl}
\gamma_{1a}=\gamma_{0a}+e_{0b}\dfrac{\bar{e}_{1a}}{\bar{E}_{1a}} & \to & \gamma_p^+(n)=\gamma_p^+(n-1)+e_p^-(n/n-1)\dfrac{e_{p-1}^+(n)}{E_{p-1}^+(n)} \\
\gamma_{1b}=\gamma_{0b}+e_{0a}\dfrac{\tilde{e}_{1b}}{\tilde{E}_{1b}} & \to & \gamma_p^-(n)=\gamma_p^-(n-1)+e_p^+(n/n-1)\dfrac{e_{p-1}^-(n-1)}{E_{p-1}^-(n-1)} \\
e_0=\bar{e}_0-g_{0b}e_{0b} & \to & e_p(n/n-1)=e_{p-1}(n/n-1)-g_p(n-1)e_p^-(n/n-1) \\
g_{1b}=g_{0b}+e_0\dfrac{e_{1b}}{E_{1b}} & \to & g_p(n)=g_p(n-1)+e_p(n/n-1)\dfrac{e_p^-(n)}{E_p^-(n)} \\
e_1=\bar{e}_1-g_{1b}e_{1b} & \to & e_p(n)=e_{p-1}(n)-g_p(n)e_p^-(n)
\end{array}$$

We have denoted the forward/backward reflection coefficients by $\gamma_p^\pm(n)$, and the lattice Wiener weights by $g_p(n)$. The order-p a priori and a posteriori estimation errors are $e_p(n/n-1)=x(n)-\hat{x}_p(n/n-1)$ and $e_p(n)=x(n)-\hat{x}_p(n)$. The likelihood variable $\mu=1-\mathbf{y}^TR_1^{-1}\mathbf{y}$ is

$$\mu_p(n)=1-\mathbf{y}_p(n)^TR_p(n)^{-1}\mathbf{y}_p(n) \tag{16.18.4}$$

and can also be written as

$$\mu_p(n)=\frac{1}{1+\nu_p(n)}=\frac{1}{1+\lambda^{-1}\mathbf{y}_p(n)^TR_p(n-1)^{-1}\mathbf{y}_p(n)}$$

Similarly, we have

$$\begin{aligned}
\tilde{\mu}_p(n) &= 1-\tilde{\mathbf{y}}_p(n)^T\tilde{R}_p(n)^{-1}\tilde{\mathbf{y}}_p(n) \\
&= 1-\mathbf{y}_{p-1}(n-1)^TR_{p-1}(n-1)^{-1}\mathbf{y}_{p-1}(n-1) \\
&= \mu_{p-1}(n-1)
\end{aligned}$$

and

$$\begin{aligned}
\bar{\mu}_p(n) &= 1-\bar{\mathbf{y}}_p(n)^T\bar{R}_p(n)^{-1}\bar{\mathbf{y}}_p(n) \\
&= 1-\mathbf{y}_{p-1}(n)^TR_{p-1}(n)^{-1}\mathbf{y}_{p-1}(n) \\
&= \mu_{p-1}(n)
\end{aligned}$$

Therefore,

$$\tilde{\mu}_p(n)=\mu_{p-1}(n-1)\,, \quad \bar{\mu}_p(n)=\mu_{p-1}(n) \tag{16.18.5}$$

Thus, the proportionality between a posteriori and a priori errors will be

$$e_p^+(n)=\tilde{\mu}_p(n)e_p^+(n/n-1)\,, \quad e_p^-(n)=\bar{\mu}_p(n)e_p^-(n/n-1) \tag{16.18.6}$$

Using either of Eq. (16.18.5), we find for the quantity $\bar{\tilde{\mu}}=\tilde{\bar{\mu}}$

$$\bar{\tilde{\mu}}_p(n)=\bar{\mu}_{p-1}(n-1)=\tilde{\mu}_{p-1}(n)=\mu_{p-2}(n-1) \tag{16.18.7}$$

Based on the above correspondences, we can obtain all versions of RLS lattice algorithms, such as the conventional a posteriori, a priori, double, and a priori and a posteriori error-feedback. In particular, we summarize the complete *double/direct RLS lattice algorithm* [156]:

0. At time n, we have available the quantities $\gamma_p^\pm(n-1)$, $g_p(n-1)$, $E_p^\pm(n-1)$, and $x(n), y(n)$.

1. Initialize in order by

$$e_0^{\pm}(n/n-1) = e_0^{\pm}(n) = y(n)$$
$$E_0^{\pm}(n) = \lambda E_0^{\pm}(n-1) + e_0^{\pm}(n)e_0^{\pm}(n/n-1)$$

$$e_0(n/n-1) = x(n) - g_0(n-1)e_0^-(n/n-1)$$
$$g_0(n) = g_0(n-1) + e_0(n/n-1)\frac{e_0^-(n)}{E_0^-(n)}$$
$$e_0(n) = x(n) - g_0(n)e_0^-(n)$$

2. For $p = 1, 2, \dots, M$, compute

$$e_p^+(n/n-1) = e_{p-1}^+(n/n-1) - \gamma_p^-(n-1)e_{p-1}^-(n-1/n-2)$$
$$e_p^-(n/n-1) = e_{p-1}^-(n-1/n-2) - \gamma_p^+(n-1)e_{p-1}^+(n/n-1)$$
$$\gamma_p^+(n) = \gamma_p^+(n-1) + e_p^-(n/n-1)\frac{e_{p-1}^+(n)}{E_{p-1}^+(n)}$$
$$\gamma_p^-(n) = \gamma_p^-(n-1) + e_p^+(n/n-1)\frac{e_{p-1}^-(n-1)}{E_{p-1}^-(n-1)}$$
$$e_p^+(n) = e_{p-1}^+(n) - \gamma_p^-(n)e_{p-1}^-(n-1)$$
$$e_p^-(n) = e_{p-1}^-(n-1) - \gamma_p^+(n)e_{p-1}^+(n)$$
$$E_p^{\pm}(n) = \lambda E_p^{\pm}(n-1) + e_p^{\pm}(n)e_p^{\pm}(n/n-1)$$
$$e_p(n/n-1) = e_{p-1}(n/n-1) - g_p(n-1)e_p^-(n/n-1)$$
$$g_p(n) = g_p(n-1) + e_p(n/n-1)\frac{e_p^-(n)}{E_p^-(n)}$$
$$e_p(n) = e_{p-1}(n) - g_p(n)e_p^-(n)$$

3. $\hat{x}_M(n) = x(n) - e_M(n)$, and go to the next time instant, $n \to n+1$.

The algorithm is initialized in time by clearing the delay registers of both lattices and setting $\gamma_p^{\pm}(-1) = 0$, $E_p^{\pm}(-1) = 0$, and $g_p(-1) = 0$. As in the case of the gradient lattice, it follows that the backward outputs from the pth lattice section, $e_p^-(n/n-1)$, will be zero for $n < p$; therefore, we must keep $\gamma_p^-(n) = g_p(n) = 0$ for $n < p$ because these quantities require divisions by $E_p^-(n)$. There are 16 multiplications/divisions in step 2; therefore, the complexity of the algorithm grows like $16M$ per time update.

The **rlsl** is an implementation of the above algorithm. It is essentially the same as **lwf** used twice for the a priori and a posteriori lattices and with the weight adaptation parts added to it.

Fig. 16.18.1 shows the reflection coefficients $\gamma_1^{\pm}(n)$ and $\gamma_2^{\pm}(n)$ adapted by the RLS lattice algorithm, for the same example presented in Sec. 16.13, which was also used in

the FAEST simulation. Note that, after some initial transients, the forward and backward reflection coefficients become more or less the same as they converge to their theoretical values. Compare also with the reflection coefficients of Fig. 16.13.3 adapted by the gradient lattice. The version of the gradient lattice that we presented uses one set of reflection coefficients, which may be thought of as some sort of average combination of the forward/backward ones. Indeed, the curves for the gradient lattice reflection coefficients fall mostly between the curves of the forward and backward ones. Similarly, the lattice Wiener weights $g_p(n)$ have almost the same behavior as those of Fig. 16.13.3. We finish this section by discussing LU factorizations. Equations (16.15.20) become

$$L_p(n)R_p(n)L_p(n)^T = D_p^-(n)\,, \quad \lambda L_p(n-1)R_p(n-1)L_p(n-1)^T = \lambda D_p^-(n-1) \tag{16.18.8}$$

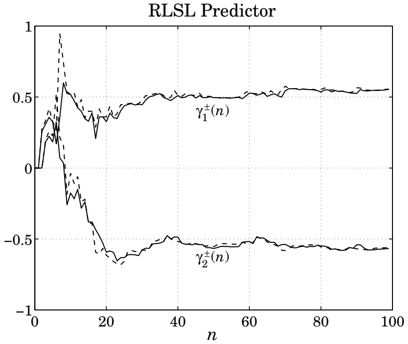


Fig. 16.18.1 Reflection coefficients adapted by the double/direct RLSL algorithm.

where

$$D_p^-(n) = \text{diag}\{E_0^-(n), E_1^-(n), \dots, E_p^-(n)\}$$

The vectors of a posteriori and a priori backward error signals are constructed by

$$\mathbf{e}_p^-(n) = \begin{bmatrix} e_0^-(n) \\ e_1^-(n) \\ \vdots \\ e_p^-(n) \end{bmatrix} = L_p(n)\mathbf{y}_p(n)\,,$$

$$\mathbf{e}_p^-(n/n-1) = \begin{bmatrix} e_0^-(n/n-1) \\ e_1^-(n/n-1) \\ \vdots \\ e_p^-(n/n-1) \end{bmatrix} = L_p(n-1)\mathbf{y}_p(n)$$

This follows from the fact that the rows of the matrices $L_p(n)$ are the backward predictors of successive orders. The $L_p(n)$ matrices are related by Eq. (16.15.54), which reads

$$L_p(n) = L_p(n/n-1)L_p(n-1) \tag{16.18.9}$$

The rows of the unit lower triangular updating matrix $L_p(n/n-1)$ are constructed by (16.15.59), that is,

$$\boldsymbol{\beta}_p = -e_p^-(n)\,\big[\lambda D_{p-1}^-(n-1)\big]^{-1}\mathbf{e}_{p-1}^-(n/n-1) \tag{16.18.10}$$

or, component-wise

$$\beta_{pi} = -e_p^-(n)\,\frac{e_i^-(n/n-1)}{\lambda E_i^-(n-1)} = -\bar{\mu}_p(n)\,e_p^-(n/n-1)\,\frac{e_i^-(n/n-1)}{\lambda E_i^-(n-1)}\,, \quad i = 0,1,\dots,p-1$$

The direct and lattice Wiener weights are related by Eq. (16.15.60), i.e., $\mathbf{g}_p(n) = L_p(n)^{-T}\mathbf{h}_p(n)$, and the a posteriori and a priori estimation errors are given by (16.15.61)

$$\hat{x}_p(n) = \mathbf{g}_p(n)^T\mathbf{e}_p(n)\,, \quad \hat{x}_p(n/n-1) = \mathbf{g}_p(n-1)^T\mathbf{e}_p^-(n/n-1) \tag{16.18.11}$$

and satisfy the recursions in order

$$\hat{x}_p(n) = \hat{x}_{p-1}(n) + g_p(n)e_p^-(n)\,, \quad \hat{x}_p(n/n-1) = \hat{x}_{p-1}(n/n-1) + g_p(n-1)e_p^-(n/n-1)$$

This implies the following recursions for the estimation errors

$$e_p(n) = e_{p-1}(n) - g_p(n)e_p^-(n)\,, \quad e_p(n/n-1) = e_{p-1}(n/n-1) - g_p(n-1)e_p^-(n/n-1)$$

Finally, the time updating equation (16.15.62) for the lattice weights takes the form

$$\mathbf{g}_p(n) = L_p(n/n-1)^{-T}\mathbf{g}_p(n-1) + e_p(n/n-1)D_p^-(n)^{-1}\mathbf{e}_p^-(n)$$

and extracting the last component, we obtain

$$g_p(n) = g_p(n-1) + e_p(n/n-1)\,\frac{e_p^-(n)}{E_p^-(n)}$$

RLS lattice and gradient adaptive lattice filters may be used in any Wiener filtering application. Their attractive features are: (a) computational *efficiency*; (b) very *fast* rate of convergence, which is essentially independent of the eigenvalue spread of the input covariance matrix; (c) *modularity* of structure admitting parallel VLSI implementations; and (d) numerical *stability* and accuracy under quantization.

16.19 Computer Project – Adaptive Wiener Filters

It is desired to design an adaptive Wiener filter to enhance a sinusoidal signal buried in noise. The noisy sinusoidal signal is given by

$$x_n = s_n + v_n\,, \qquad \text{where} \quad s_n = \sin(\omega_0 n)$$

with $\omega_0 = 0.075\pi$. The noise v_n is related to the secondary signal y_n by

$$v_n = y_n + y_{n-1} + y_{n-2} + y_{n-3} + y_{n-4} + y_{n-5} + y_{n-6}$$

The signal y_n is assumed to be an order-4 AR process with reflection coefficients:

$$\{\gamma_1, \gamma_2, \gamma_3, \gamma_4\} = \{0.5, -0.5, 0.5, -0.5\}$$

The variance σ_ϵ^2 of the driving white noise of the model must be chosen in such a way as to make the variance σ_v^2 of the noise component v_n approximately unity.

a. For a Wiener filter of order $M = 6$, determine the theoretical direct-form Wiener filter coefficient vector:

$$\mathbf{h} = [h_0, h_1, \dots, h_6]$$

for estimating x_n (or, rather v_n) from y_n. Determine also the theoretical lattice/ladder realization coefficients:

$$\boldsymbol{\gamma} = [\gamma_1, \gamma_2, \dots, \gamma_6]\,, \qquad \mathbf{g} = [g_0, g_1, \dots, g_6]$$

b. Generate input pairs $\{x_n, y_n\}$ (making sure that the transients introduced by the filter have died out), and filter them through the LMS algorithm to generate the filter output pairs $\{\hat{x}_n, e_n\}$. On the same graph, plot e_n together with the desired signal s_n.

Plot also a few of the adaptive filter coefficients such as $h_4(n)$, $h_5(n)$, and $h_6(n)$. Observe their convergence to the theoretical Wiener solution.

You must generate enough input pairs in order to achieve convergence of the LMS algorithm and observe the steady-state converged output of the filter.

Experiment with the choice of the adaptation parameter μ. Start by determining $\lambda_{\max}$, $\lambda_{\min}$, the eigenvalue spread $\lambda_{\max}/\lambda_{\min}$ of R and the corresponding time constant.

c. Repeat (b), using the gradient lattice adaptive filter. Plot all of the adaptive reflection coefficients $\gamma_p(n)$ versus n, and a few of the ladder coefficients, such as $g_4(n)$, $g_5(n)$, and definitely $g_6(n)$.

(Because theoretically $g_6 = h_6$ (why?), plotting $h_6(n)$ and $g_6(n)$ will let you compare the convergence speeds of the LMS and lattice adaptive filters.)

You must experiment with a couple of values of λ (use $\beta = 1$). You must work of course with exactly the same set of input pairs as in part (b).

d. Next, we change this experiment into a non-stationary one. Suppose the total number of input pairs that you used in parts (b) and (c) is N. And suppose that at time $n = N$, the input statistics changes suddenly so that the primary signal is given now by the model:

$$x_n = s_n + v_n\,, \qquad \text{where} \quad v_n = y_n + y_{n-1} + y_{n-2} + y_{n-3}$$

and y_n changes from a fourth-order AR model to a second-order model with reflection coefficients (use the same σ_ϵ^2

$$\{\gamma_1, \gamma_2\} = \{0.5, -0.5\}$$

Repeat parts (a,b,c), keeping the filter order the same, $M = 6$. Use $2N$ input pairs, such that the first N follow the original statistics and the second N follow the changed statistics. Compare the capability of the LMS and lattice adaptive filters in tracking such changes.

Here, the values of μ for the LMS case and λ for the lattice case, will make more of a difference in balancing the requirements of learning speed and quality of estimates.

e. Finally, feel free to “tweak” the statements of all of the above parts as well as the definition of the models in order to show more clearly and more dramatically the issues involved, namely, LMS versus lattice, learning speed versus quality, and the effect of the adaptation parameters, eigenvalue spread, and time constants. One other thing to notice in this experiment is that, while the adaptive weights tend to fluctuate a lot as they converge, the actual filter outputs $\hat{x}_n, e_n$ behave better and are closer to what one might expect.

16.20 Problems

16.1 *Computer Experiment.* (a) Reproduce the results of Fig. 16.3.2.

(b) On the same graph of part (a), plot the theoretical convergence curve of the weight $h(n)$ obtained by using Eq. (16.2.8).

(c) Using 10 different realizations of x_n and y_n, compute 10 different realizations of the adaptive weight of Eq. (16.3.2). Compute the average weight over the 10 realizations and plot it versus n, together with the theoretical weight of Eq. (16.2.8). Use $\mu = 0.03$.

(d) Reproduce the results of Fig. 16.5.2.

16.2 In steered adaptive arrays [1093] and other applications, one has to solve a constrained Wiener filtering problem. Suppose the $(M+1)$-dimensional weight vector $\mathbf{h} = [h_0, h_1, \dots, h_M]^T$ satisfies the L linear constraints $\mathbf{c}_i^T\mathbf{h} = f_i$, $i = 1, 2, \dots, L$, where $L \le M$ and the $\mathbf{c}_i$ are given $(M+1)$-dimensional vectors, and f_i are given scalars. The set of constraints may be written compactly as $C^T\mathbf{h} = \mathbf{f}$, where $C = [\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_L]$ and $\mathbf{f} = [f_1, f_2, \dots, f_L]^T$.

(a) Show that the solution of the minimization problem $\mathcal{E} = E[e_n^2] = \min$, subject to the constraint $C^T\mathbf{h} = \mathbf{f}$, is given by

$$\mathbf{h} = \mathbf{h}_u + R^{-1}C(C^TR^{-1}C)^{-1}(\mathbf{f} - C^T\mathbf{h}_u)$$

where $\mathbf{h}_u = R^{-1}\mathbf{r}$ is the unconstrained Wiener solution and $R = E[\mathbf{y}(n)\mathbf{y}(n)^T]$, $\mathbf{r} = E[x_n\mathbf{y}(n)]$.

(b) In an adaptive implementation, $\mathbf{h}(n+1) = \mathbf{h}(n) + \Delta\mathbf{h}(n)$, the constraint must be satisfied at each iteration. The time update term, therefore, must satisfy $C^T\Delta\mathbf{h}(n) = 0$. Show that the following (gradient projection) choice satisfies this condition

$$\Delta\mathbf{h}(n) = -\mu\mathcal{P}\frac{\partial\mathcal{E}}{\partial\mathbf{h}(n)}, \quad \mathcal{P} = I - C(C^TC)^{-1}C^T$$

Moreover, show that this choice moves the performance index closer to its minimum at each iteration.

(c) Show that the resulting difference equation can be written as

$$\mathbf{h}(n+1) = \mathcal{P}\big[\mathbf{h}(n) - 2\mu R\mathbf{h}(n) + 2\mu\mathbf{r}\big] + \mathbf{h}_{LS}$$

where $\mathbf{h}_{LS} = C(C^TC)^{-1}\mathbf{f}$ is recognized as the least-squares solution of the linear equation $C^T\mathbf{h} = \mathbf{f}$. And, show that $C^T\mathbf{h}(n+1) = \mathbf{f}$.

(d) Show that the LMS adaptive algorithm resulting by dropping the expectation values is, with $e_n = x_n - \hat{x}_n = x_n - \mathbf{h}(n)^T\mathbf{y}(n)$

$$\mathbf{h}(n+1) = \mathcal{P}\big[\mathbf{h}(n) + 2\mu e_n\mathbf{y}(n)\big] + \mathbf{h}_{LS}$$

16.3 Rederive the results in parts (c) and (d) of Problem 16.2 using the following approach. Introduce a Lagrange multiplier vector $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \dots, \lambda_L]^T$ into the performance index enforcing the constraint equations; that is, $\mathcal{E} = E[e_n^2] + \boldsymbol{\lambda}^T(\mathbf{f} - C^T\mathbf{h})$. Show that the ordinary unconstrained gradient descent method $\mathbf{h}(n+1) = \mathbf{h}(n) - \mu\partial\mathcal{E}/\partial\mathbf{h}(n)$ gives rise to the difference equation

$$\mathbf{h}(n+1) = (I - 2\mu R)\mathbf{h}(n) + 2\mu\mathbf{r} - \mu C\boldsymbol{\lambda}(n)$$

Impose the constraint $C^T\mathbf{h}(n+1) = \mathbf{f}$, eliminate $\boldsymbol{\lambda}(n)$, and show that this equation is equivalent to that in part (c) of the previous problem.

16.4 Verify that Eq. (16.6.5) is the solution of Eq. (16.6.4).

16.5 Consider an adaptive filter with two taps:

$$\hat{x}_n = h_0(n)y_n + h_1(n)y_{n-1} = [h_0(n), h_1(n)]\begin{bmatrix} y_n \\ y_{n-1} \end{bmatrix} = \mathbf{h}(n)^T\mathbf{y}(n)$$

The optimal filter weights are found adaptively by the gradient descent algorithm

$$\mathbf{h}(n+1) = \mathbf{h}(n) - \mu\frac{\partial\mathcal{E}}{\partial\mathbf{h}(n)}$$

where $\mathcal{E} = E[e_n^2]$ and e_n is the estimation error.

(a) Show that the above difference equation may be written as

$$\mathbf{h}(n+1) = \mathbf{h}(n) + 2\mu\big(\mathbf{r} - R\mathbf{h}(n)\big)$$

where

$$\mathbf{r} = \begin{bmatrix} R_{xy}(0) \\ R_{xy}(1) \end{bmatrix}, \quad R = \begin{bmatrix} R_{yy}(0) & R_{yy}(1) \\ R_{yy}(1) & R_{yy}(0) \end{bmatrix}$$

(b) Suppose $R_{xy}(0) = 10$, $R_{xy}(1) = 5$, $R_{yy}(0) = 3$, $R_{yy}(1) = 2$. Find the optimal weights $\mathbf{h} = \lim \mathbf{h}(n)$ as $n \to \infty$.

(c) Select $\mu = 1/6$. Explain why such a value is sufficiently small to guarantee convergence of the difference equation of part (a). What other values of μ also guarantee convergence?

(d) With $\mu = 1/6$, solve the difference equation of part (a) in closed form for $n \ge 0$. Discuss the rate of convergence.

16.6 Consider a single CCL as shown in Fig. 16.3.1.

(a) Suppose the reference signal is set equal to a unit step signal; that is, $y(n) = u(n)$. Show that the CCL will behave as a time-invariant linear filter with input x_n and output e_n. Determine the transfer function $H(z)$ from x_n to e_n.

(b) Find and interpret the poles and zeros of $H(z)$.

(c) Determine the range of μ-values for which $H(z)$ is stable.

16.7 Repeat Problem 16.6 when the reference signal is the alternating unit step; that is, $y(n) = (-1)^n u(n)$.

16.8 Let $\mathbf{h}_R$ and $\mathbf{h}_I$ be the real and imaginary parts of the complex weight vector $\mathbf{h} = \mathbf{h}_R + j\,\mathbf{h}_I$. Show that

$$\frac{\partial\mathcal{E}}{\partial\mathbf{h}^*} = \frac{1}{2}\left[\frac{\partial\mathcal{E}}{\partial\mathbf{h}_R} + j\,\frac{\partial\mathcal{E}}{\partial\mathbf{h}_I}\right]$$

Consider the simultaneous gradient descent with respect to $\mathbf{h}_R$ and $\mathbf{h}_I$, that is, $\mathbf{h}_R \to \mathbf{h}_R + \Delta\mathbf{h}_R$ and $\mathbf{h}_I \to \mathbf{h}_I + \Delta\mathbf{h}_I$, with

$$\Delta\mathbf{h}_R = -\mu\frac{\partial\mathcal{E}}{\partial\mathbf{h}_R}\,, \quad \Delta\mathbf{h}_I = -\mu\frac{\partial\mathcal{E}}{\partial\mathbf{h}_I}$$

Show that it is equivalent to the gradient descent $\mathbf{h} \to \mathbf{h} + \Delta\mathbf{h}$, where

$$\Delta\mathbf{h} = -2\mu\frac{\partial\mathcal{E}}{\partial\mathbf{h}^*}$$

Note the conjugation and the factor of two.

16.9 Using the transfer function of Eq. (16.9.1), derive an approximate expression for the 3-dB width of the notch. You may work to lowest order in μ.

16.10 *Computer Experiment.* Consider the noise canceling example discussed in Sec. 12.11 and in Problems 12.25–12.27 and defined by the following choice of parameters:

$$\omega_0 = 0.075\pi \text{ [rads/sample]}\,, \quad \phi = 0\,, \quad a_1 = -0.5\,, \quad a_2 = 0.8\,, \quad M = 4$$

(a) Generate a realization of the signals $x(n)$ and $y(n)$ and process them through the adaptive noise canceler of Sec. 16.9, using an Mth order adaptive filter and adaptation parameter μ. By trial and error select a value for μ that makes the LMS algorithm convergent, but not too small as to make the convergence too slow. Plot one of the filter weights $h_m(n)$ versus iteration number n, and compare the asymptotic value with the theoretical value obtained in Problem 12.26.

(b) After the weights have converged, plot 100 output samples of the error signal $e(n)$, and observe the noise cancellation property.

(c) Repeat (a) and (b) using an adaptive filter of order $M = 6$.

16.11 *Computer Experiment.* (a) Plot the magnitude of the frequency response of the adaptive noise canceler notch filter of Eq. (16.9.1) versus frequency ω $(z = e^{j\omega})$. Generate several such plots for various values of μ and observe the effect of μ on the width of the notch.

(b) Let $x(n) = e^{j\omega_0 n}$ and $y(n) = Ae^{j\omega_0 n}$, and select the parameters as

$$\omega_0 = 0.075\pi\,, \quad M = 2\,, \quad A = 0.01\,, \quad \mu = 0.1$$

Process $x(n)$ and $y(n)$ through the adaptive noise canceler of Sec. 16.9, and plot the output $e(n)$ versus n and observe the cancellation of the signal $x(n)$ due to the notch filter created by the presence of the weak sinusoidal reference signal $y(n)$.

16.12 *Computer Experiment.* Let $x(n) = x_1(n) + x_2(n)$, where $x_1(n)$ is a narrowband component defined by

$$x_1(n) = \sin(\omega_0 n + \phi)\,, \quad \omega_0 = 0.075\pi \text{ [rads/sample]}$$

where ϕ is a random phase uniformly distributed over $[0, 2\pi]$, and $x_2(n)$ is a fairly broadband component generated by sending zero-mean, unit-variance, white noise $\epsilon(n)$ through the filter

$$x_2(n) = \epsilon(n) + 2\epsilon(n-1) + \epsilon(n-2)$$

(a) Compute the autocorrelation functions of $x_1(n)$ and $x_2(n)$ and sketch them versus lag k. Based on this computation, select a value for the delay Δ to be used in the adaptive line enhancer discussed in Sec. 16.10.

(b) Generate a realization of $x(n)$ and process it through the ALE with an appropriately chosen adaptation parameter μ. Plot the output signals $\hat{x}(n)$ and $e(n)$, and compare them with the components $x_1(n)$ and $x_2(n)$, respectively.

16.13 The response of the ALE to an input sinusoid in noise can be studied as follows: Let the input be

$$x_n = A_1 e^{j\omega_1 n + j\phi} + v_n$$

where ϕ is a random phase independent of the zero-mean white noise v_n. The optimum Wiener filter weights of the ALE are given by

$$\mathbf{h} = R^{-1}\mathbf{r}$$

where $R_{ij} = R_{xx}(i-j)$ and $r_i = R_x(i+\Delta)$, as discussed in Sec. 16.10.

(a) Using the methods of Sec. 14.2, show that the optimum filter $\mathbf{h}$ is given by

$$\mathbf{h} = \frac{e^{j\omega_1\Delta}}{\dfrac{\sigma_v^2}{P_1} + M + 1}\mathbf{s}_{\omega_1}$$

where the phasing vector $\mathbf{s}_{\omega_1}$ was defined in Sec. 14.2, and $P_1 = |A_1|^2$ is the power of the sinusoid.

(b) Show that the mean output power of the ALE is given by

$$E\big[|\hat{x}_n|^2\big] = \mathbf{h}^\dagger R\mathbf{h} = \sigma_v^2\,\mathbf{h}^\dagger\mathbf{h} + P_1|\mathbf{h}^\dagger\mathbf{s}_{\omega_1}|^2$$

(c) Show that the SNR at the output is enhanced by a factor $M+1$ over the SNR at the input; that is, show that

$$(SNR)_{\text{out}} = \frac{P_1|\mathbf{h}^\dagger\mathbf{s}_{\omega_1}|^2}{\sigma_v^2\,\mathbf{h}^\dagger\mathbf{h}} = \frac{P_1}{\sigma_v^2}(M+1) = (M+1)\,(SNR)_{\text{in}}$$

(d) Derive an expression for the eigenvalue spread $\lambda_{\max}/\lambda_{\min}$ in terms of the parameters σ_v^2, P_1, and M.

(e) Show that if the delay Δ is removed; that is, $\Delta = 0$, then the optimal weight vector becomes equal to the unit vector

$$\mathbf{h} = [1, 0, 0, \dots, 0]^T$$

and that this choice corresponds to complete cancellation of the input signal $x(n)$ from the output $e(n)$.

16.14 *Computer Experiment.* Consider the autoregressive process y_n generated by the difference equation

$$y_n = -a_1 y_{n-1} - a_2 y_{n-2} + \epsilon_n$$

where $a_1 = -1.6$, $a_2 = 0.8$, and ϵ_n is zero-mean, unit-variance, white noise. Generate a realization of y_n and process it through the LMS adaptive predictor of order 2, as discussed in Sec. 16.11. Use a value for the adaptation parameter μ of your own choice. Plot the adaptive prediction coefficients $a_1(n)$ and $a_2(n)$ versus n, and compare their converged values with the theoretical values given above.

16.15 The adaptive predictor may be considered as the linearly constrained minimization problem $\mathcal{E} = E[e_n^2] = \min$, subject to the constraint that the first element of $\mathbf{a} = [1, a_1, \dots, a_M]^T$ be unity. This constraint may be written compactly as $\mathbf{u}^T\mathbf{a} = 1$, where $\mathbf{u} = [1, 0, \dots, 0]^T$. Rederive the adaptation equations of Sec. 16.11 using the formalism and results of Problem 16.2.

16.16 *Computer Experiment.* A complex-valued version of the LMS adaptive predictor of Sec. 16.11 is defined by

$$e_n = y_n + a_1(n)y_{n-1} + a_2(n)y_{n-2} + \cdots + a_M(n)y_{n-M}$$

$$a_m(n+1) = a_m(n) - 2\mu e_n y_{n-m}^*\,, \quad m = 1, 2, \dots, M$$

Let y_n consist of two complex sinusoids in zero-mean white noise

$$y_n = A_1 e^{j\omega_1 n} + A_2 e^{j\omega_2 n} + v_n$$

where the frequencies and the SNRs are

$$\omega_1 = 0.3\pi\,, \quad \omega_2 = 0.7\pi \text{ [radians/sample]}$$

$$10\log_{10}\big[|A_1|^2/\sigma_v^2\big] = 10\log_{10}\big[|A_2|^2/\sigma_v^2\big] = 20 \text{ dB}$$

(a) Generate a realization of y_n (using a complex-valued v_n) and process it through an Mth order LMS adaptive predictor using an adaptation constant μ. Experiment with several choices of M and μ. In each case, stop the algorithm after convergence has taken place and plot the AR spectrum $S(\omega) = 1/|A(\omega)|^2$ versus frequency ω. Discuss your results.

(b) Using the same realization of y_n, iterate the adaptive Pisarenko algorithm defined by Eqs. (16.12.5) and (16.12.6). After convergence of the Pisarenko weights, plot the Pisarenko spectrum estimate $S(\omega) = 1/|A(\omega)|^2$ versus frequency ω.

(c) Repeat (a) and (b) when the SNR of the sinewaves is lowered to 0 dB. Compare the adaptive AR and Pisarenko methods.

16.17 *Computer Experiment.* Reproduce the results of Figs. 7.19 and 7.20.

16.18 Derive Eqs. (16.14.8) and (16.14.9) that describe the operation of the adaptive linear combiner in the decorrelated basis provided by the Gram-Schmidt preprocessor.

16.19 *Computer Experiment.* Reproduce the results of Fig. 16.14.2.

16.20 What is the exact operational count of the conventional RLS algorithm listed in Sec. 16.15? Note that the inverse matrices P_0 and P_1 are symmetric and thus only their lower-triangular parts need be updated.

16.21 Verify the solution (16.15.56) for the rank-one updating of the LU factors L_0 and L_1. Also verify that Eq. (16.15.58) is equivalent to (16.15.54).

16.22 *Computer Experiment.* Reproduce the results of Fig. 16.17.1. Carry out the same experiment (with the same input data) using the conventional RLS algorithm and compare with FAEST. Carry out both experiments with various values of λ and comment on the results.

16.23 *Computer Experiment.* Reproduce the results of Fig. 16.18.1.

17

Appendices

A Matrix Inversion Lemma

The matrix inversion lemma, also known as Woodbury's identity, is useful in Kalman filtering and recursive least-squares problems. Consider the matrix relationship,

$$\boxed{R = A + UBV} \tag{A.1}$$

where

$$A \in \mathbb{C}^{N\times N}\,, \quad U \in \mathbb{C}^{N\times M}\,, \quad B \in \mathbb{C}^{M\times M}\,, \quad V \in \mathbb{C}^{M\times N}$$

and assume that A, B are both *invertible* and that $M \le N$. Then, the term UBV has rank M, while R, A have rank N. The matrix inversion lemma states that the inverse of R can be obtained from the inverses of A, B via the formula,

$$\boxed{R^{-1} = (A + UBV)^{-1} = A^{-1} - A^{-1}U\big[B^{-1} + VA^{-1}U\big]^{-1}VA^{-1}} \tag{A.2}$$

Proof: Multiply both sides of (A.1) by R^{-1} from the right, and then by A^{-1} from the left to obtain,

$$A^{-1} = R^{-1} + A^{-1}UBVR^{-1} \tag{A.3}$$

then, multiply both sides from the left by V,

$$VA^{-1} = VR^{-1} + VA^{-1}UBVR^{-1} \quad \Rightarrow \quad VA^{-1} = \big[I_M + VA^{-1}UB\big]VR^{-1}$$

where I_M is the $M \times M$ identity matrix, and solve for BVR^{-1},

$$VA^{-1} = \big[B^{-1} + VA^{-1}U\big]BVR^{-1} \quad \Rightarrow \quad BVR^{-1} = \big[B^{-1} + VA^{-1}U\big]^{-1}VA^{-1}$$

and substitute back into (A.3), after solving for R^{-1},

$$R^{-1} = A^{-1} - A^{-1}UBVR^{-1} = A^{-1} - A^{-1}U\big[B^{-1} + VA^{-1}U\big]^{-1}VA^{-1}$$

Thus given A^{-1} and B^{-1}, the inverse of the $N \times N$ matrix R requires only the inverse of the smaller $M \times M$ matrix, $B^{-1} + VA^{-1}U$.

B MATLAB Functions

```
% OSP Toolbox
% S. J. Orfanidis - 2018
%
% -----------------------------------------------------------------------------
% Local Polynomial Smoothing Filters
% -----------------------------------------------------------------------------
% binom     - vector of binomial coefficients
% bkfilt    - Baxter-King bandpass filter
% cldec     - classical decomposition method
% combfd    - comb fractional-delay filter design
% compl     - complement of an odd-length symmetric filter
% diffb     - backward difference operator
% diffmat   - difference convolution matrix
% diffpol   - differentiate polynomial
% diffs     - seasonal backward difference operator
% ecg       - ECG generator.
% ecgsim    - ECG simulation
% filtdbl   - filtering with double-sided FIR filter
% hahnbasis - Hahn orthogonal polynomials
% hahncoeff - coefficients of Hahn orthogonal polynomials
% hahnpol   - Hahn orthogonal polynomial evaluation
% hahnrec   - Hahn orthogonal polynomials
% hend      - Henderson weighting function
% kmat      - difference convolution matrix
% kraw      - Krawtchouk binomial weighting function
% kwindow   - Kaiser window for spectral analysis
% lagrfd    - Lagrange-interpolation fractional-delay filter
% lpbasis   - local polynomial basis
% lpdiff    - weighted local polynomial differentiation filters
% lpfilt    - local polynomial filtering - fast version
% lpfilt2   - local polynomial filtering - slower version
% lpinterp  - local polynomial interpolation and differentiation filters
% lpmat     - local polynomial smoothing matrix
% lpmissing - weighted local polynomial filters for missing data
% lprs      - local polynomial minimum-Rs smoothing filters
% lprs2     - local polynomial minimum-Rs smoothing filters (closed-form)
% lpsm      - weighted local polynomial smoothing and differentiation filters
% minrev    - minimum revision asymmetric filters
% polval    - polynomial evaluation in factorial power series
% rlpfilt   - robust local polynomial filtering
% sigav     - signal averaging
% smadec    - decomposition using seasonal moving-average filters
% smafilt   - impulse responses of seasonal decomposition moving average filters
% smat      - seasonal moving-average filtering matrix
% smav      - seasonal moving average filter
% stirling  - Stirling numbers of first or second kind, signed or unsigned
% swhdec    - seasonal Whittaker-Henderson decomposition
% trendma   - trend moving-average filter, 2xD if D is even, 1xD if D is odd
% upmat     - upsample matrix of smoothing filters
% whkdec    - Whittaker-Henderson-Kaiser seasonal decomposition
% x11dec    - US Census X-11 decomposition method for seasonal adjustment
% x11filt   - impulse responses of the US Census X-11 seasonal adjustment filters

% -----------------------------------------------------------------------------
% Local Linear Regression
```

```
% -----------------------------------------------------------------------------
% avobs   - average repeated observations
% locband - bandwidth for local polynomial regression
% locgcv  - local polynomial GCV and CV evaluation
% locgrid - uniform grid for local polynomial evaluation
% locpol  - local polynomial regression
% locval  - evaluation/interpolation of local polynomial regression
% locw    - local weighting functions for local polynomial regression
% loess   - Cleveland's robust locally weighted scatterplot smoothing (loess)
% loess2  - Cleveland's robust locally weighted scatterplot smoothing (loess)

% -----------------------------------------------------------------------------
% Spline and Whittaker-Henderson Smoothing
% -----------------------------------------------------------------------------
% splambda - find optimum lambda for spline smoothing using GCV
% splav    - averaged repeated observations at spline knots
% splcoeff - spline coefficients
% splgcv   - evaluate GCV(lambda)
% splmat   - spline smoothing matrices Q,T
% splsm    - spline smoothing using Reinsch's algorithm
% splsm2   - spline smoothing using Reinsch's algorithm - robust version
% splval   - evaluate spline smoothing polynomials
% whgcv    - Whittaker-Henderson smoothing method
% whgen    - generalized Whittaker-Henderson
% whimp    - Whittaker-Henderson filter impulse response
% whsm     - Whittaker-Henderson smoothing method
% whsm1    - Whittaker-Henderson smoothing method - L1 version

% -----------------------------------------------------------------------------
% Exponentially Weighted Averages
% -----------------------------------------------------------------------------
% binmat  - binomial boost matrices for exponential smoothers
% ema     - exponential moving average - exact version
% emaerr  - calculate MAE, MSE, and MAPE for a range of lambda's
% emap    - map equivalent lambdas between d=0 EMA and d=1 EMA
% emat    - polynomial to cascaded transformation matrix
% holt    - Holt's exponential smoothing
% holterr - calculate MAE, MSE, and MAPE for a range of lambda's
% mema    - multiple exponential moving average
% stema   - steady-state exponential moving average

% -----------------------------------------------------------------------------
% Linear Prediction & Wiener and Kalman Filtering Functions
% -----------------------------------------------------------------------------
% acext     - autocorrelation sequence extension using Levinson recursion
% acf       - sample auto-correlation function
% acmat     - construct autocorrelation Toeplitz matrix from autocorrelation lags
% acsing    - sinusoidal representation of singular autocorrelation matrices
% aicmdl    - estimates dimension of signal subspace from AIC and MDL criteria
% argen     - generate a zero-mean segment of an AR process
% bkwlev    - backward Levinson recursion
% burg      - Burg's method of linear prediction
% dir2nl    - direct form to normalized lattice
% dpd       - dynamic predictive deconvolution
% dwf       - sample processing algorithm of direct-form Wiener filter
% dwf2      - direct-form Wiener filter using circular delay-line buffer
% dwfilt    - direct-form Wiener filtering of data
% dwfilt2   - circular-buffer direct-form Wiener filtering of data
% faest     - sample processing algorithm of adaptive lattice Wiener filter
% firw      - FIR Wiener filter design
% flipv     - flip a vector, column, row, or both for a matrix
% frwlev    - forward Levinson recursion
% glwf      - sample processing algorithm of lattice Wiener filter
% kfilt     - Kalman filtering
% ksmooth   - Kalman smoothing
% latt      - sample processing algorithm of analysis lattice filter
% lattfilt  - lattice filtering of a data vector
% lattsect  - sample processing algorithm of a single lattice section
% lattsynth - sample processing algorithm of synthesis lattice filter
% lev       - Levinson-Durbin recursion
% lms       - sample processing LMS algorithm of direct-form Wiener filter
% lpf       - extract linear prediction filter from matrix L
% lpg       - extract reflection coefficients from matrix L
% lpspec    - compute LP spectrum of a prediction-error filter
% lwf       - sample processing algorithm of lattice Wiener filter
% lwfilt    - lattice Wiener filtering of data
% mgs       - adaptive modified Gram-Schmidt
% mgslms    - adaptive Gram-Schmidt using LMS
% minorm    - minimum-norm noise subspace eigenvector
% music     - MUSIC spectrum computation
% nlfilt    - filtering in the normalized lattice form
% obmat     - observability matrix for canonical or transposed realizations
% obmatc    - observability matrix for continuous-time
% rlev      - reverse of Levinson's algorithm
% rls       - RLS algorithm for adaptive linear combiner
% rlsl      - sample processing algorithm of lattice Wiener filter
% rmusic    - minimum-norm noise subspace eigenvector
% scatt     - direct scattering problem
% schur1    - Schur algorithm for linear prediction
% schur2    - Schur algorithm for Cholesky factorization
% spike     - spiking filter design
% yw        - Yule-Walker method of linear prediction


% ----------------------------------------------------------------------------
% SVD, Subspace, and ARMA Modeling Functions
% ----------------------------------------------------------------------------
% arma2imp  - ARMA impulse response
% armaacf   - ARMA autocorrelation function
% armachol  - ARMA covariance matrix Cholesky factorization
% armafit   - fitting an ARMA(p,q) model to covariance lags
% armainf   - ARMA asymptotic Fisher information matrix
% armainnov - ARMA modeling using the innovations method
% armamf    - Mayne-Firoozan ARMA modeling method
% armamyw   - ARMA modeling by the modified Yule-Walker method
% armasim   - simulate a zero-mean segment of a gaussian ARMA process
% armasim2  - simulate a zero-mean segment of a gaussian ARMA process
% bwidth    - beamwidth mapping from psi-space to phi-space
% cca       - Canonical Correlation Analysis
% ccacov    - CCA applied to a covariance matrix
% cholgs    - Cholesky factorization by Gram-Schmidt orthogonalization
% cholinnov - Cholesky factorization by innovations representation
% crb       - calculate Cramer-Rao bounds for sinusoids in noise
% crb2      - calculate Cramer-Rao bounds for sinusoids in noise
% datamat   - convolution data matrix of a signal vector
% datasig   - extract data signal from a Toeplitz or Toeplitz/Hankel data matrix
```

```
% dolph     - Dolph-Chebyshev array weights
% fisher    - calculate Fisher information matrix for sinusoids in noise
% imp2arma  - impulse response to ARMA coefficients
% irls      - Lp_regularized iteratively reweighted least squares
% irls_wh   - Lp_regularized IRLS Whittaker-Henderson
% lpls      - construct least-squares linear prediction filter from data matrix
% madurbin  - MA modeling by Durbin's method
% mafit     - Wilson's method of fitting an MA(q) model to covariance lags
% mainnov   - MA modeling by the innovations method
% mpencil   - matrix-pencil method of extracting sinusoids in noise
% poly2     - specialized version of poly
% scan      - scan array with given scanning phase
% setrank   - reduce the rank of a diagonal matrix of singular values
% sigsub    - construct reduced-rank signal subspace of a data matrix
% sines     - generate sum of real or complex decaying sinusoids in noise
% snap      - generate snapshot matrix for array problems
% snapshot  - generate data matrix of snapshots for array problems
% snr       - magnitude to SNR in dB, and conversely
% steer     - steer array towards given angle
% steering  - construct steering matrix of multiple sinusoids/plane-waves
% steermat  - construct steering matrix of multiple sinusoids/plane-waves
% svdenh    - SVD signal enhancement
% toepl     - Toeplitz, Hankel, or Toeplitz/Hankel approximation of data matrix
% varper    - percentage variances

% ----------------------------------------------------------------------------
% Wavelet Functions
% ----------------------------------------------------------------------------
% advance  - circular time-advance (left-shift) of a vector
% casc     - cascade algorithm for phi and psi wavelet functions
% circonv  - circular convolution
% cmf      - conjugate mirror of a filter
% convat   - convolution a trous
% convmat  - sparse convolution matrix
% convmat2 - sparse convolution matrix (simplified version)
% daub     - Daubechies scaling filters (daublets, symmlets, coiflets)
% dn2      - downsample by a factor of 2
% dwtcell  - cell array of sparse discrete wavelet transform matrices
% dwtdec   - DWT decomposition into orthogonal multiresolution components
% dwtmat   - discrete wavelet transform matrices
% dwtmat2  - discrete wavelet transform matrices
% dwtwrap  - wrap a DWT matrix into a lower DWT matrix
% flipv    - flip a vector, column, row, or both for a matrix
% fwt      - fast wavelet transform using convolution and downsampling
% fwtm     - fast wavelet transform in matrix form
% fwtmat   - overall DWT orthogonal matrix
% ifwt     - inverse fast wavelet transform using upsampling and convolution
% ifwtm    - inverse fast wavelet transform in matrix form
% iuwt     - inverse undecimated wavelet transform
% iuwtm    - inverse undecimated wavelet transform
% modwrap  - wrap matrix column-wise mod-N
% phinit   - eigenvector initialization of phi
% plotdec  - plot DWT/UWT decomposition or DWT/UWT coefficients
% up2      - upsample a vector by factor of two
% upr      - upsample a vector by factor of 2^r
% uwt      - undecimated wavelet transform
% uwtdec   - UWT multiresolution decomposition
% uwtm     - undecimated wavelet transform
% uwtmat   - undecimated wavelet transform matrices
% uwtmat2  - undecimated wavelet transform matrices
% w2V      - wavelet vector to wavelet matrix
% wcoeff   - extract wavelet coefficients from DWT at given level
% wdenoise - Donoho & Johnstone's VisuShrink denoising procedure
% wduwt    - wavelet denoising with UWT
% wthr     - soft/hard level-dependent wavelet thresholding

% -----------------------------------------------------------------------------
% Technical Analysis Functions
% -----------------------------------------------------------------------------
% accdist -  accumulation/distribution line
% atr     -  true range & average true range
% bbands  - Bollinger bands
% bma     - Butterworth moving average
% cci     -  commodity channel index
% chosc   -  Chaikin oscillator
% chvol   -  Chaikin volatility
% cmflow  -  Chaikin money flow
% cmo     - Chande momentum oscillator
% delay   - lag or delay or advance by d samples
% dema    - steady-state double exponential moving average
% dirmov  -  directional movement system
% dmi     - dynamic momentum index (DMI)
% donch   - Donchian channels
% dpo     - detrended price oscillator
% ehma    - exponential Hull moving average
% fbands  - fixed-envelope bands
% forosc  - forecast oscillator
% gdema   - generalized dema
% hma     - Hull moving average
% ilrs    - integrated linear regression slope indicator
% kbands  - Keltner bands or channels
% lreg    - linear regression, slope, and R-squared indicators
% mom     - momentum and price rate of change
% ohlc    - make Open-High-Low-Close bar chart
% ohlcyy  - OHLC plot with other indicators on the same graph
% pbands  - Projection Bands and Projection Oscillator
% pma     - predictive moving average, linear fit
% pma2    - predictive moving average, polynomial order d=1,2
% pmaimp  - predictive moving average impulse response
% pmaimp2 - predictive moving average impulse response, d=1,2
% pnvi    - positive and negative volume indices (PVI & NVI)
% prosc   - price oscillator & MACD
% psar    - Wilder's parabolic SAR
% r2crit  - R-squared critical values
% rsi     - relative strength index (RSI)
% sebands - standard-error bands
% sema    - single exponential moving average
% shma    - SMA-based Hull moving average
% sma     - simple moving average
% stbands - STARC bands
% stdev   - standard deviation index
% stoch   - stochastic oscillator
% t3      - Tillson's T3 indicator, triple gdema
% tcrit   - critical values of Student's t-distribution
% tdistr  - cumulative t-distribution
% tema    - triple exponential moving average
```

```
% tma     - triangular moving average
% trix    - TRIX oscillator
% vema    - variable-length exponential moving average
% vhfilt  - Vertical Horizontal Filter
% wema    - Wilder's exponential moving average
% wma     - weighted or linear moving average
% yylim   - adjust left/right ylim & ticks

% -----------------------------------------------------------------------------
% Miscellaneous Functions
% -----------------------------------------------------------------------------
% canfilt  - IIR filtering in canonical form using linear delay-line buffer
% ccan     - IIR filter in canonical form using circular delay-line buffer
% ccanfilt - IIR filtering in canonical form using circular delay-line buffer
% frespc   - frequency response of a cascaded IIR filter at a frequency vector w
% loadfile - load data file ignoring any text lines
% taxis    - define time axis
% up       - upsample by a factor of L
% ustep    - unit-step or rising unit-step function
% xaxis    - set x-axis limits and tick marks
% yaxis    - set y-axis limits and tick marks
% zmean    - zero mean of each column of a data matrix (or row vector)
```

References

References for Chap. 1

[1] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*, (2nd ed.), New York, McGraw-Hill, 1984; and 4th ed., with S. U. Pillai, 2002.

[2] M. G. Kendall and A. Stuart, *The Advanced Theory of Statistics*, vol. 2, (4th ed.), London, Griffin, 1979.

[3] H. W. Sorenson, *Parameter Estimation*, New York, Marcel Dekker, 1980.

[4] T. W. Anderson, *An Introduction to Multivariate Statistical Analysis*, (2nd ed.), New York, Wiley, 1984.

[5] M. G. Kendall and A. Stuart, *The Advanced Theory of Statistics*, vol. 3, (3d ed.), New York, Hafner Press, 1976.

[6] J. Cryer, *Times Series Analysis*, Boston, Duxbury Press, 1986.

[7] J. L. Doob, *Stochastic Processes*, New York, Wiley, 1953.

[8] P. R. Halmos, *Finite-Dimensional Vector Spaces*, New York, Van Nostrand, 1958.

[9] R. B. Blackman and J. W. Tukey, *The Measurement of Power Spectra*, New York, Dover, 1958.

[10] C. Bingham, M. D. Godfrey, and J. W. Tukey, Modern Techniques of Power Spectrum Estimation, *IEEE Trans. Audio Electroacoust.*, **AU-15**, 56–66 (1967).

[11] G. M. Jenkins and D. G. Watts, *Spectral Analysis and Its Applications*, San Francisco, Holden-Day, 1968.

[12] A. V. Oppenheim and R. W. Schafer, *Digital Signal Processing*, Englewood Cliffs, NJ, Prentice Hall, 1975.

[13] J. S. Lim and A. V. Oppenheim, eds., *Advanced Topics in Signal Processing*, Prentice Hall, Upper Saddle River, NJ, 1988.

[14] R. K. Otnes and L. Enochson, *Digital Time Series Analysis*, New York, Wiley, 1972.

[15] W. Davenport and W. Root, *Introduction to the Theory of Random Signals and Noise*, New York, McGraw-Hill, 1958.

[16] D. Childers, Ed., *Modern Spectrum Analysis*, New York, Wiley, 1978.

[17] F. J. Harris, On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform, *Proc. IEEE*, **66**, 51–83 (1978).

[18] A. H. Nuttal and G. C. Carter, A Generalized Framework for Power Spectral Estimation, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-28**, 334–335 (1980).

[19] S. M. Kay, *Modern Spectral Estimation*, Englewood Cliffs, NJ, Prentice Hall, 1988.

[20] S. L. Marple, *Digital Spectral Analysis with Applications*, Englewood Cliffs, NJ, Prentice Hall, 1987.

[21] P. D. Welch, The Use of Fast Fourier Transform for the Estimation of Power Spectra: A Method Based on Time Averaging over Short, Modified Periodograms, *IEEE Trans. Audio Electroacoust.*, **AU-15**, 70–73 (1967).

[22] G. P. Box, G. M. Jenkins, and G. C. Reinsel, *Time Series Analysis Forecasting and Control*, 4/e, Wiley, New York, 2008.

[23] H. Wold, *A Study in the Analysis of Time Series*, Uppsala, Sweden, Almqvist and Wiksell, 1931 and 1954.

[24] A. Papoulis, Predictable Processes and Wold's Decomposition: A Review, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-33**, 933 (1985).

[25] A. N. Kolmogorov, Sur l'Interpolation et Extrapolation des Suites Stationnaires, *C. R. Acad. Sci.*, **208**, 2043–2045 (1939). See also "Interpolation and Extrapolation of Stationary Random Sequences, and Stationary Sequences in Hilbert Space," reprinted in T. Kailath, Ed., *Linear Least-Squares Estimation*, Stroudsburg, PA, Dowden, Hutchinson, and Ross, 1977.

[26] E. A. Robinson, *Time Series Analysis and Applications*, Houston, TX, Goose Pond Press, 1981.

[27] C. R. Rao, *Linear Statistical Inference and Its Applications*, (2nd ed.), New York, Wiley, 1973.

[28] D. S. G. Pollock, *Handbook of Time Series Analysis, Signal Processing, and Dynamics*, Academic, New York, 1999.

[29] A. V. Oppenheim, R. W. Schafer, and J. R. Buck, *Discrete-Time Signal Processing*, 2nd ed., Prentice Hall, Upper Saddle River, NJ, 1999.

[30] S. J. Orfanidis, *Introduction to Signal Processing*, Prentice Hall, Upper Saddle River, NJ, 1996. Available online from: `http://www.ece.rutgers.edu/~orfanidi/intro2sp/`.

[31] S. J. Orfanidis, *Optimum Signal Processing*, 2nd ed., online book, 2007, available from: `http://www.ece.rutgers.edu/~orfanidi/osp2e/`.

[32] S. Lang and J. McClellan, A Simple Proof of Stability for All-Pole Linear Prediction Models, *Proc. IEEE*, **67**, 860-861 (1979).

[33] S. Kay and L. Pakula, Simple Proofs of the Minimum Phase Property of the Prediction Error Filter, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-31**, 501 (1983).

[34] P. Stoica and A. Nehorai, On Stability and Root Location of Linear Prediction Models, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-35**, 582 (1987).

[35] S. J. Orfanidis, A Proof of the Minimal Phase Property of the Prediction Error Filter, *Proc. IEEE*, **71**, 905 (1983).

Local Polynomial Smoothing Filters

[36] G. V. Schiaparelli, "Sul Modo Di Ricavare La Vera Espressione Delle Leggi Della Natura Dalle Curve Empiriche," *Effemeridi Astronomiche di Milano per l'anno 1866*, p.3–56, reprinted in *Le Opere di G. V. Schiaparelli*, vol.8, Ulrico Hoepli Publisher, Milano, 1930, and Johnson Reprint Corp., New York.

[37] A. Lees, "Interpolation and Extrapolation of Sampled Data," *IEEE Trans. Inform. Th.*, **2**, 12 (1956).

[38] K. R. Johnson, "Optimum, Linear, Discrete Filtering of Signals Containing a Nonrandom Component," *IEEE Trans. Inform. Th.*, **2**, 49 (1956).

[39] M. Blum, "An Extension of the Minimum Mean Square Prediction Theory for Sampled Input Signals," *IEEE Trans. Inform. Th.*, **IT-2**, 176 (1956).

[40] M. Blum, "On the Mean Square Noise Power of an Optimum Linear Discrete Filter Operating on Polynomial plus White Noise Input," *IEEE Trans. Inform. Th.*, **IT-3**, 225 (1957).

[41] J. D. Musa, "Discrete Smoothing Filters for Correlated Noise," *Bell Syst. Tech. J.*, **42**, 2121 (1963).

[42] A. Savitzky and M Golay, "Smoothing and Differentiation of Data by Simplified Least Squares Procedures," *Anal. Chem.*. **36**, 1627 (1964).

[43] M. U. A. Bromba and H. Ziegler, "Efficient Computation of Polynomial Smoothing Digital Filters," *Anal. Chem.*, **51**, 1760 (1979).

[44] M. U. A. Bromba and H. Ziegler, "Application Hints for Savitzky-Golay Digital Smoothing Filters," *Anal. Chem.*, **53**, 1583 (1981).

[45] T. H. Edwards and P. D. Wilson, "Digital Least Squares Smoothing of Spectra," *Appl. Spectrosc.*, **28**, 541 (1974).

[46] T. H. Edwards and P. D. Wilson, "Sampling and Smoothing of Spectra," *Appl. Spectrosc. Rev.*, **12**, 1 (1976).

[47] C. G. Enke and T. A. Nieman, "Signal-to-Noise Ratio Enhancement by Least-Squares Polynomial Smoothing," *Anal. Chem.*, **48**, 705A (1976).

[48] H. H. Madden, "Comments on the Savitzky-Golay Convolution Method for Least-Squares Fit Smoothing and Differentiation of Digital Data," *Anal. Chem.*, **50**, 1383 (1978).

[49] R. A. Leach, C. A. Carter, and J. M. Harris, "Least-Squares Polynomial Filters for Initial Point and Slope Estimation," *Anal. Chem.*, **56**, 2304 (1984).

[50] P. A. Baedecker, "Comments on Least-Squares Polynomial Filters for Initial Point and Slope Estimation," *Anal. Chem.*, **57**, 1477 (1985).

[51] J. Steinier, Y. Termonia, and J. Deltour, "Comments on Smoothing and Differentiation of Data by Simplified Least Squares Procedures," *Anal. Chem.*. **44**, 1627 (1972).

[52] H. Ziegler, "Properties of Digital Smoothing Polynomial (DISPO) Filters," *Appl. Spectrosc.*, **35**, 88 (1981).

[53] G. R. Phillips and J. M. Harris, "Polynomial Filters for Data Sets with Outlying or Missing Observations: Application to Charged-Coupled-Device- Detected Raman Spectra Contaminated by Cosmic Rays," *Anal. Chem.*, **62**, 2351 (1990).

[54] M. Kendall, *Time-Series*, 2nd ed., Hafner Press, Macmillan, New York, 1976.

[55] M. Kendall and A. Stuart, *Advanced Theory of Statistics*, vol. 3, 2nd ed., Charles Griffin & Co., London, 1968.

[56] R. W. Hamming, *Digital Filters*, 2nd ed., Prentice Hall, Upper Saddle River, NJ, 1983.

[57] C. S. Williams, *Designing Digital Filters*, Prentice Hall, Upper Saddle River, NJ, 1986.

[58] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes in C*, 2nd ed., Cambridge Univ. Press, New York, 1992.

[59] J. F. Kaiser and W. A. Reed, "Data Smoothing Using Lowpass Digital Filters," *Rev. Sci. Instrum.*, **48**, 1447 (1977).

[60] J. F. Kaiser and R. W. Hamming, "Sharpening the Response of a Symmetric Nonrecursive Filter by Multiple Use of the Same Filter," *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-25**, 415 (1975).

[61] J. Luo, et al., "Properties of Savitzky-Golay Digital Differentiators," *Dig. Sig. Process.*, **15**, 122 (2005).

[62] J. Luo, "Savitzky-Golay Smoothing and Differentiation Filter for Even Number Data," *Signal Process.*, **85**, 1429 (2005).

[63] S. Hargittai, "Savitzky-Golay Least-Squares Polynomial Filters in ECG Signal Processing," *Computers Cardiol.*, **32**, 763 (2005).

[64] T. C. Mills, "A Note on Trend Decomposition: The 'Classical' Approach Revisited with an Application to Surface Temperature Trends," *J. Appl. Statist.*, **34**, 963 (2007).

Henderson Filters

[65] E. L. De Forest, "On Some Methods of Interpolation Applicable to the Graduation of Irregular Series, such as Tables of Mortality," *Ann. Rep. Board of Regents of Smithsonian Institution*, 1871, p.275. Also, *ibid.*, 1873, p.319.

[66] E. L. De Forest, "On Adjustment Formulas," *The Analyst* (De Moines, Iowa), **4**, 79 (1877), and *ibid.*, p. 107.

[67] E. L. De Forest, "On the Limit of Repeated Adjustments," *The Analyst* (De Moines, Iowa), **5**, 129 (1878), and *ibid.*, p. 65.

[68] H. H. Wolfenden, "Development of Formulae for Graduation by Linear Compounding, With Special Reference to the Work of Erastus L. De Forest," *Trans. Actuarial Soc. Am.*, **26**, 81 (1925).

[69] F. R. Macauley, *The Smoothing of Time Series*, Nat. Bureau Econ. Res., NY, 1931.

[70] M. D. Miller, *Elements of Graduation*, Actuarial Soc. Am. and Am. Inst. Actuaries, 1946.

[71] C. A. Spoerl, "Actuarial Science—A Survey of Theoretical Development," *J. Amer. Statist. Assoc.*, **46**, 334 (1951).

[72] S. M. Stigler, "Mathematical Statistics in the Early States," *Ann. Statist.*, **6**, 239 (1978).

[73] H. L. Seal, "The Fitting of a Mathematical Graduation Formula: A Historical Review with Illustrations," *Blätter. Deutsche Gesellschaft für Versicherungsmathematik*, **14**, 237 (1980).

[74] H. L. Seal, "Graduation by Piecewise Cubic Polynomials: A Historical Review," *Blätter. Deutsche Gesellschaft für Versicherungsmathematik*, **15**, 89 (1981).

[75] J. M. Hoem, "The Reticent Trio: Some Little-Known Early Discoveries in Life Insurance Mathematics by L. H. Opperman, T. N. Thiele, and J. P. Gram," *Int. Statist. Rev.*, **51**, 213 (1983).

[76] W. F. Sheppard, "Reduction of Errors by Means of Negligible Differences," *Proc. Fifth Int. Congress of Mathematicians*, **2**, 348 (1912), Cambridge.

[77] W. F. Sheppard, "Fitting Polynomials by Method of Least Squares," *Proc. London Math. Soc., Ser. 2*, **13**, 97 (1913).

[78] W. F. Sheppard, "Graduation by Reduction of Mean Square Error," *J. Inst. Actuaries*, **48**, 171 (1914), see also, *ibid.*, **48**, 412 (1914), and **49**, 148 (1915).

[79] R. Henderson, "Note on Graduation by Adjusted Average," *Trans. Actuarial Soc. Am.*, **18**, 43 (1916).

[80] H. Vaughan, "Further Enquiries into the Summation Method of Graduation," *J. Inst. Actuaries*, **66**, 463 (1935).

[81] K. Weichselberger, "Über eine Theorie der gleitenden Durchschnitte und verschiedene Anwendungen dieser Theorie," *Metrica*, **8**, 185 (1964).

[82] I. J. Schoenberg, "Some Analytical Aspects of the Problem of Smoothing," in *Studies and Essays Presented to R. Courant on his 60th Birthday*, Interscience, NY, 1948.

[83] I. J. Schoenberg, "On Smoothing Operations and Their Generating Functions," *Bull. Am. Math. Soc.*, **59**, 199 (1953).

[84] T. N. E. Greville, "On Stability of Linear Smoothing Formulas," *SIAM J. Numer. Anal.*, **3**, 157 (1966).

[85] W. F. Trench, "Stability of a Class of Discrete Minimum Variance Smoothing Formulas," *SIAM J. Numer. Anal.*, **9**, 307 (1972).

[86] T. N. E. Greville, "On a Problem of E. L. De Forest in Iterated Smoothing," *SIAM J. Math. Anal.*, **5**, 376 (1974).

[87] O. Borgan, "On the Theory of Moving Average Graduation," *Scand. Actuarial J.*, p. 83, (1979).

[88] P. B. Kenny and J. Durbin, "Local Trend Estimation and Seasonal Adjustment of Economic and Social Time Series," *J. Roy. Statist. Soc., Ser. A*, **145**, 1 (1982).

[89] D. London, *Graduation: The Revision of Estimates*, ACTEX publications, Winsted, CT, 1985.

[90] E. S. W. Shiu, "Minimum-R_Z Moving-Average Formulas," *Trans. Soc. Actuaries*, **36**, 489 (1984).

[91] E. S. W. Shiu, "A Survey of Graduation Theory," in H. H. Panjer, ed., *Actuarial Mathematics*, Proc. Symp. Appl. Math, vol.35, 1986.

[92] E. S. W. Shiu, "Algorithms for MWA Graduation Formulas," *Actuarial Res. Clearing House*, **2**, 107 (1988).

[93] W. D. Hoskins and P. J. Ponzo, "Some Properties of a Class of Band Matrices," *Math. Comp.*, **26**, 393 (1972).

[94] A. Eisinberg, P. Pugliese, and N. Salerno, "Vandermonde Matrices on Integer Nodes: The Rectangular Case," *Numer. Math.*, **87**, 663 (2001).

[95] M. Dow, "Explicit Inverse of Toeplitz and Associated Matrices," *ANZIAM J.*, **44** (E), 185 (2003).

[96] A. Grey and P. Thomson, "Design of Moving-Average Trend Filters Using Fidelity, Smoothness and Minimum Revisions Criteria," Res. Rep. CENSUS/SRD/RR-96/1, Statistical Research Division, Bureau of the Census, Washington, DC.

[97] T. Proietti and A. Luati, "Least Squares Regression: Graduation and Filters," in M. Boumans, ed., *Measurement in Economics: A Handbook*, Academic, London, 2007.

[98] T. Proietti and A. Luati, "Real Time Estimation in Local Polynomial Regression, with Application to Trend-Cycle Analysis," *Ann. Appl. Statist.*, **2**, 1523 (2008).

[99] A. Luati and T. Proietti, "On the Equivalence of the Weighted Least Squares and the Generalised Least Squares Estimators," *Compstat 2008—Proc. Comput. Statist.*, P. Brito, ed., Physica-Verlag, Heidelberg, 2008. Available online from `http://mpra.ub.uni-muenchen.de/8910/`

Asymmetric End-Point Filters

[100] T. N. E, Greville, "On Smoothing a Finite Table," *J. SIAM*, **5**, 137 (1957).

[101] T. N. E, Greville, "Band Matrices and Toeplitz Inverses," *Lin. Alg. Appl.*, **27**, 199 (1979).

[102] T. N. E, Greville, "Moving-Weighted-Average Smoothing Extended to the Extremities of the Data. I. Theory," *Scand. Actuarial J.*, p. 39, (1981), and "part II. Methods,", *ibid.* p.65. See also "Part III. Stability and Optimal Properties,", *J. Approx. Th.*, **33** 43 (1981).

[103] J. M. Hoem and P. Linnemann, "The Tails in Moving Average Graduation," *Scand. Actuarial J.*, p. 193, (1988).

Discrete Chebyshev and Hahn Polynomials

[104] P. L. Chebyshev, “Sur l’Interpolation,” reprinted in A. Markoff and N. Sonin, *Oeuvres de P. L. Chebyshev*, vol.1, p. 541, Commissionaires de l’Acádemie Impériale des Sciences, St. Petersbourg, 1899, also Chelsea Publishing Co. , NY, 1961. See also p. 203, 381, 473, 701, and vol.2, p. 219. Available online from `http://www.archive.org/details/uvresdepltcheby00chebgoog`

[105] P. Butzer and F. Jongmans, “P. L. Chebyshev (1821-1894), A Guide to His Life and Work,” *J. Approx. Th.*, **96**, 111 (1999).

[106] C. Jordan, “Sur une Série de Polynomes Dont Chaque Somme Partielle Représente la Meilleure Approximation d’un Degré Donné Suivant la Méthode des Moindres Carrés,” *Proc. London Math. Soc.,* 2nd series, **20**, 297 (1922).

[107] L. Isserlis and V. Romanovsky, “Notes on Certain Expansions in Orthogonal and Semi-Orthogonal Functions,” *Biometrika*, **19**, 87 (1927).

[108] C. Jordan, *Calculus of Finite Differences*, Chelsea Publishing Co. NY, 1939.

[109] G. Szegö, *Orthogonal Polynomials*, Am Math. Soc., Providence, RI, 1939.

[110] P. T. Birge and J. W. Weinberg, “Least Squares Fitting of Data by Means of Polynomials,” *Rev. Mod. Phys.*, **19**, 298 (1947).

[111] M. Weber and A. Erdélyi, “On the Finite Difference Analogue of Rodrigues’ Formula,” *Am. Math. Monthly*, **59**, 163 (1952).

[112] G. E. Forsythe, “Generation and Use of Orthogonal Polynomials for Data-Fitting with a Digital Computer,” *J. Soc. Indust. Appl. Math.*, **5**, 74 (1957).

[113] S. Karlin and J. L. McGregor, “The Hahn Polynomials, Formulas and an Application,” *Scripta Math.*, **26**, 33 (1961).

[114] P. G. Guest, *Numerical Methods of Curve Fitting*, Cambridge Univ. Press, London, 1961.

[115] N. Morrison, *Introduction to Sequential Smoothing and Prediction*, McGraw-Hill, NY, 1969.

[116] B. A. Finlayson, *The Method of Weighted Residuals and Variational Principles*, Academic Press, NY, 1972.

[117] D. E. Clapp, “Adaptive Forecasting with Orthogonal Polynomial Filters,” *AIIE Trans.*, **6**, 359 (1974).

[118] F. B. Hildebrand, *Introduction to Numerical Analysis*, 2/e, McGraw-Hill, New York, 1974, reprinted by Dover Publications, Mineola, NY, 1987.

[119] R. R. Ernst, “Sensitivity Enhancement in Magnetic Resonance,” in *Advances in Magnetic Resonance*, vol. 2, J. S. Waugh, ed., Academic Press, 1966.

[120] C. P. Neuman and D. I. Schonbach, “Discrete (Legendre) Orthogonal Polynomials—A Survey,” *Int. J. Numer. Meth. Eng.*, **8**, 743 (1974).

[121] A. Proctor and P. M. A. Sherwood, “Smoothing of Digital X-ray Photoelectron Spectra by and Extended Sliding Least-Squares Approach,” *Anal. Chem.*, **52** 2315 (1980).

[122] P. D. Willson and S. R. Polo, “Polynomial Filters of any Degree,” *J. Opt. Soc. Am.*, **71**, 599 (1981).

[123] M. U. A. Bromba and H. Ziegler, “On Hilbert Space Design of Least-Weighted- Squares Digital Filters,” *Int. J. Circuit Th. Appl.*, **11**, 7 (1983).

[124] P. Steffen, “On Digital Smoothing Filters: A Brief Review of Closed Form Solutions and Two New Filter Approaches,” *Circ., Syst., and Signal Process.*, fb5, 187 (1986).

[125] H. W. Schüssler and P. Steffen, “Some Advanced Topics in Filter Design,” in Ref. [13].

[126] S. E. Bialkowski, “Generalized Digital Smoothing Filters Made Easy by Matrix Calculations,” *Anal. Chem.*, **61**, 1308 (1989).

[127] P. A. Gorry, “General Least-Squares Smoothing and Differentiation of by the Convolution (Savitzky-Golay) Method,” *Anal. Chem.*, **62**, 570 (1990).

[128] P. A. Gorry, “General Least-Squares Smoothing and Differentiation of Nonuniformly Spaced Data by the Convolution Method,” *Anal. Chem.*, **63**, 534 (1991).

[129] J. E. Kuo and H. Wang, “Multidimensional Least-Squares Smoothing Using Orthogonal Polynomials,” *Anal. Chem.*, **63**, 630 (1991).

[130] G. Y. Pryzva, “Kravchuk Orthogonal Polynomials,” *Ukranian Math. J.*, **44**, 792 (1992).

[131] P. Persson and G. Strang, “Smoothing by Savitzky-Golay and Legendre Filters,” in J. Rosenthal and D. S. Gilliam, eds., *Mathematical Systems Theory in Biology, Communications, Computation, and Finance*, Springer-Verlag, NY, 2003.

[132] W. Gautschi, *Orthogonal Polynomials: Computation and Approximation*, Clarendon Press, Oxford, 2004.

[133] M. E. H. Ismail, *Classical and Quantum Orthogonal Polynomials in One Variable*, Cambridge University Press, Cambridge, (2005).

[134] S. Samadi and A. Nishihara, “Explicit Formula for Predictive FIR Filters and Differentiators Using Hahn Orthogonal Polynomials,” *IEICE Trans. Fundamentals*, **E90**, 1511 (2007).

[135] M. J. Gottlieb, “Concerning Some Polynomials Orthogonal on a Finite or Enumerable Set of Points,” *A. J. Math*, **60**, 453 (1938).

[136] R. E. King and P. N. Paraskevopoulos, “Digital Laguerre Filters,” *Circ. Th. Appl.*, **5**, 81 (1977).

[137] M. R. Teague, “Image Analysis via the General Theory of Moments,” *J. Opt. Soc. Am.*, **70**, 920 (1980).

[138] R. M. Haralick, “Digital Step Edges from Zero Crossing of Second Directional Derivatives,” *IEEE Trans. Patt. Anal. Mach. Intell.*, **PAMI-6**, 58 (1984).

[139] C-S. Liu and H-C. Wang, “A Segmental Probabilistic Model of Speech Using an Orthogonal Polynomial Representation,” *Speech Commun.*, **18** 291 (1996).

[140] P. Meer and I. Weiss, “Smoothed Differentiation Filters for Images,” *J. Vis. Commun. Imag. Process.*, **3**, 58 (1992).

[141] G. Carballo, R. Álvarez-Nodarse, and J. S. Dehesa, “Chebyshev Polynomials in a Speech Recognition Model,” *Appl. Math. Lett.*, **14**, 581 (2001).

[142] R. Mukundan, S. H. Ong, and P. A. Lee, “Image Analysis by Tchebichef Moments,” *IEEE Trans. Image Process.*, **10**, 1357 (2001).

[143] J. Arvesú, J. Coussement, and W. Van Asscheb, “Some Discrete Multiple Orthogonal Polynomials,” *J. Comp. Appl. Math.*, **153**, 19 (2003).

[144] R. Mukundan, “Some Computational Aspects of Discrete Orthonormal Moments,” *IEEE Trans. Image Process.*, **13**, 1055 (2004).

[145] L. Kotoulas and I. Andreadis, “Image Analysis Using Moments,” *Proc. IEEE Int. Conf. Technol. Autom.* (ICTA-05), p.360, (2005).

[146] L. Kotoulas and I. Andreadis, “Fast Computation of Chebyshev Moments,” *IEEE Trans. Circuits Syst. Video Technol.*, **16**, 884 (2006).

[147] K. W. Lee, et al., “Image reconstruction Using Various Discrete Orthogonal Polynomials in Comparison with DCT,” *Appl. Math. Comp.*, **193**, 346 (2007).

[148] H. Zhu, et al., “Image Analysis by Discrete Orthogonal Dual Hahn Moments,” *Patt. Recogn. Lett.* **28**, 1688 (2007).

[149] H. Shu, L. Luo, and J-L Coatrieux, “Moment-Based Approaches in Imaging. Part 1, Basic Features,” *IEEE Eng. Med. Biol. Mag.*, **26**, no.5, 70 (2007).

[150] H. Shu, L. Luo, and J-L Coatrieux, “Moment-Based Approaches in Imaging. Part 2, Invariance,” *IEEE Eng Med Biol Mag.*, **27**, no.1, 81 (2008).

[151] E. Diekema and T. H. Koornwinder, “Differentiation by integration using orthogonal polynomials, a survey,” , *J. Approx.*, **164**, 637 (2012).

Predictive and Fractional-Delay Filters

[152] R. W. Schafer and L. R. Rabiner, “A Digital Signal Processing Approach to Interpolation,” *Proc. IEEE*, **61**, 692 (1973).

[153] H. W. Strube, “Sampled-Data Representation of a Nonuniform Lossless Tube of Continuously Variable Length,” *J. Acoust. Soc. Amer.*, **57**, 256 (1975).

[154] P. Heinonen and Y. Neuvo, “FIR-Median Hybrid Filters with Predictive FIR Substructures,” *IEEE Trans. Acoust., Speech, Signal Process.*, **36**, 892 (1988).

[155] C. W. Farrow, “A Continuously Variable Digital Delay Element,” *Proc. IEEE Int. Symp. Circuits and Systems, ISCAS-88*, p. 2641, (1988).

[156] G-S Liu and C-H Wei, “Programmable Fractional Sample Delay Filter with Lagrange Interpolation,” *Electronics Lett.*, **26**, 1608 (1990).

[157] T. G. Campbell and Y. Neuvo, “Predictive FIR Filters with Low Computational Complexity,” *IEEE Trans. Circ. Syst.*, **38** 1067 (1991).

[158] S. J. Ovaska, “Improving the Velocity Sensing Resolution of Pulse Encoders by FIR Prediction,” *IEEE Trans. Instr. Meas.*, **40**, 657 (1991).

[159] S. J. Ovaska, “Newton-Type Predictors—A Signal Processing Perspective,” *Signal Process.*, **25**, 251 (1991).

[160] G-S Liu and C-H Wei, “A New Variable Fractional Sample Delay Filter with Nonlinear Interpolation,” *IEEE Trans. Circ. Syst.–II*, **39**, 123 (1992).

[161] L. Erup., F. M. Gardner, and R. A. Harris, “Interpolation in Digital Modems—Part II: Implementation and Performance,” *IEEE Trans. Commun.*, **41**, 998 (1993).

[162] T. I. Laakso, et al., “Splitting the Unit Delay—Tools for Fractional Delay Filter Design,” *IEEE Signal Process. Mag.*, **13**, 30, Jan. 1996.

[163] P. J. Kootsookos and R. C. Williamson, “FIR Approximation of Fractional Sample Delay Systems,” *IEEE Trans. Circ. Syst.–II*, **43**, 269 (1996).

[164] O. Vainio, M. Renfors, and T. Saramäki, “Recursive Implementation of FIR Differentiators with Optimum Noise Attenuation,” *IEEE Trans. Instrum. Meas.*, **46**, 1202 (1997).

[165] P. T, Harju, “Polynomial Prediction Using Incomplete Data,” *IEEE Trans. Signal Process.*, **45**, 768 (1997).

[166] S. Tassart and P. Depalle, “Analytical Approximations of Fractional Delays: Lagrange Interpolators and Allpass Filters,” *IEEE Int. Conf. Acoust., Speech, Sig. Process.*, (ICASSP-97), **1** 455 (1997).

[167] S. Väliviita and S. J. Ovaska, “Delayless Recursive Differentiator with Efficient Noise Attenuation for Control Instrumentation,” *Signal Process.*, **69**, 267 (1998).

[168] S-C Pei and C-C Tseng, “A Comb Filter Design Using Fractional-Sample Delay,” *IEEE Trans. Circ. Syst.–II*, **45**, 649 (1998).

[169] S. Väliviita, S. J. Ovaska, and O. Vainio, “Polynomial Predictive Filtering in Control and Instrumentation: A Review,” *IEEE Trans. Industr. Electr.*, **46**, 876 (1999).

[170] E. Meijering, “A Chronology of Interpolation: From Ancient Astronomy to Modern Signal and Image Processing,” *Proc. IEEE*, **90**, 319 (2002).

[171] V. Välimäki, et al. “Discrete-Time Modeling of Musical Instruments,” *Rep. Progr. Phys.*, **69**, 1 (2006).

[172] C. Candan, “An Efficient Filtering Structure for Lagrange Interpolation,” *IEEE Signal Proc. Lett.*, **14**, 17 (2007).

[173] J. Vesma and T. Saramäki, “Polynomial-Based Interpolation Filters—Part I: Filter Synthesis,” *Circ. Syst, Signal Process.*, **26**, 115 (2007).

Maximally Flat Filters

[174] O. Herrmann, “On the Approximation Problem in Nonrecursive Digital Filter Design,” *IEEE Trans. Circ. Th.*, **CT-18**, 411 (1971).

[175] J. A. Miller, “Maximally Flat Nonrecursive Digital Filters,” *Electron. Lett.*, **8**, 157 (1972).

[176] M. F. Fahmy, “Maximally Flat Nonrecursive Digital Filters,” *Int. J. Circ. Th. Appl*, **4**, 311 (1976).

[177] J-P. Thiran, “Recursive Digital Filters with Maximally Flat Group Delay,” *IEEE Trans. Circ. Th.*, **CT-18**, 659 (1971).

[178] M. U. A. Bromba and H. Ziegler, “Explicit Formula for Filter Function of Maximally Flat Nonrecursive Digital Filters,” *Electron. Lett.*, **16**, 905 (1980), and *ibid.*, **18**, 1014 (1982).

[179] H. Baher, “FIR Digital Filters with Simultaneous Conditions on Amplitude and Group Delay,” *Electron. Lett.*, **18**, 296 (1982).

[180] L. R. Rajagopal and S. C. D. Roy, “Design of Maximally-Flat FIR Filters Using the Bernstein Polynomial,,, *IEEE Trans. Circ. Syst.*, **CAS-34**, 1587 (1987).

[181] E. Hermanowicz, “Explicit Formulas for Weighting Coefficients of Maximally Flat Tunable FIR delayers,” *Electr. Lett.*, **28**, 1936 (1992).

[182] I. W. Selesnick and C. S. Burrus, “Maximally Flat Low-Pass FIR Filters with Reduced Delay,” *IEEE Trans. Circ. Syst. II*, **45**, 53 (1998).

[183] I. W. Selesnick and C. S. Burrus, “Generalized Digital Butterworth Filter Design,” *IEEE Trans. Signal Process.*, **46**, 1688 (1998).

[184] S. Samadi, A. Nishihara, and H. Iwakura, “Universal Maximally Flat Lowpass FIR Systems,” *IEEE Trans. Signal Process.*, **48**, 1956 (2000).

[185] R. A. Gopinath, “Lowpass Delay Filters With Flat Magnitude and Group Delay Constraints,” *IEEE Trans. Signal Process.*, **51**, 182 (2003).

[186] S. Samadi, O. Ahmad, and M, N, S. Swami, “Results on Maximally Flat Fractional-Delay Systems,” *IEEE Trans. Circ. Syst.–I*, **51**, 2271 (2004).

[187] S. Samadi and A. Nishihara, “The World of Flatness,” *IEEE Circ. Syst. Mag.*, p.38, third quarter 2007.

Local Polynomial Modeling and Loess

[188] E. A. Nadaraya, “On Estimating Regression,” *Th. Prob. Appl.*, **10**, 186 (1964).

[189] G. S. Watson, “Smooth Regression Analysis,” *Sankya, Ser. A*, **26**, 359 (1964).

[190] M. B. Priestley and M. T. Chao, “Non-Parametric Function Fitting,” *J. Roy. Statist. Soc., Ser. B,* **34**, 385 (1972).

[191] C. J. Stone, “Consistent Nonparametric Regression (with discussion),” *Ann. Statist.*, **5**, 595 (1977).

[192] W. S. Cleveland, “Robust Locally Weighted Regression and Smoothing of Scatterplots,” *J. Amer. Statist. Assoc.*, **74**, 829 (1979).

[193] W. S. Cleveland and R. McGill “The Many Faces of a Scatterplot,” *J. Amer. Statist. Assoc.*, **79**, 807 (1984).

[194] . H. Friedman, “A Variable Span Smoother,” Tech. Rep. No. 5, Lab. Comput. Statist., Dept. Statist., Stanford Univ., (1984); see also, J. H. Friedman and W. Stueltze, “Smoothing of Scatterplots,” Dept. Statist., Tech. Rep. Orion 3, (1982).

[195] H-G. Müller, “Smooth Optimum Kernel Estimators of Densities, Regression Curves and Modes,” *Ann. Statist.*, **12**, 766 (1984).

[196] T. Gasser, H-G. Müller, and V. Mammitzsch, “Kernels for Nonparametric Curve Estimation,” *J. Roy. Statist. Soc., Ser. B*, **47**, 238 (1985).

[197] J. A. McDonald and A. B. Owen, “Smoothing with Split Linear Fits,” *Technometrics*, **28**, 195 (1986).

[198] A. B. Tsybakov, “Robust Reconstruction of Functions by the Local-Approximation Method,” *Prob. Inf. Transm.*, **22**, 69 (1986).

[199] W. S. Cleveland and S. J. Devlin, “Locally Weighted Regression: An Approach to Regression Analysis by Local Fitting,” *J. Amer. Statist. Assoc.*, **83**, 596 (1988).

[200] A. Buja, A. Hastie, and R. Tibshirani, “Linear Smoothers and Additive Models (with discussion),” *Ann. Statist.*, **17**, 453 (1989).

[201] B. L. Granovsky and H-G. Müller, “The Optimality of a Class of Polynomial Kernel Functions,” *Stat. Decis.*, **7**, 301 (1989).

[202] W. Härdle, *Applied Nonparametric Regression*, Cambridge Univ. Press, Cambridge, 1990.

[203] A. Hastie and R. Tibshirani, *Generalized Additive Models*, Chapman & Hall, London, 1990.

[204] B. L. Granovsky, H-G. Müller, “Optimizing Kernel Methods: A Unifying Variational Principle,” *Int. Stat. Rev.*, **59**, 373 (1991).

[205] N. S. Altman, “An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression,” *Amer. Statist.*, **46**, 175 (1992).

[206] I. Fan and I. Gijbels, “Variable Bandwidth and Local Linear Regression Smoothers,” *Ann. Statist.*, **20**, 2008 (1992).

[207] W. S. Cleveland and Grosse, “A Package of C and Fortran Routines for Fitting Local Regression Models,” 1992. Available from: `http://www.netlib.org/a/dloess`.

[208] W. S. Cleveland, *Visualizing Data*, Hobart Press, Summit, NJ, 1993.

[209] I. Fan, “Local Linear Regression Smoothers and Their Minimax Efficiencies,” *Ann. Statist.*, **21**, 196 (1993).

[210] A. Hastie and C. Loader, "Local Regression: Automatic Kernel Carpentry," *Statist. Sci.*, **8**, 120 (1993).

[211] M. C. Jones, S. J. Davies, and B. U. Park, "Versions of Kernel-Type Regression Estimators," *J. Amer. Statist. Assoc.*, **89**, 825 (1994).

[212] I. Fan and I. Gijbels, "Data-Driven Bandwidth Selection in Local Polynomial Fitting: Variable Bandwidth and Spatial Adaptation," *J. Roy. Statist. Soc., Ser. B*, **57**, 371 (1995).

[213] D. Ruppert, S. J. Sheather, and M. P. Wand, "An Effective Bandwidth Selector for Local Least Squares Regression," *J. A. Statist. Assoc.*, **90**, 125 (1995).

[214] M. P. Wand and M. C. Jones, *Kernel Smoothing*, Chapman & Hall, London, 1995.

[215] W. S. Cleveland and C. Loader, "Smoothing by Local Regression: Principles and Methods," in W. Härdle and M. G. Schimek, eds., *Statistical Theory and Computational Aspects of Smoothing*, Physica-Verlag, Heidelberg, May 1996.

[216] M. C. Jones, J. S. Marron, and S. J. Sheaver, "A Brief Survey of Bandwidth Selection for Density Estimation," *J. Amer. Statist. Assoc.*, **91**, 401 (1996).

[217] B. Seifert and T. Gasser, "Finite Sample Variance of Local Polynomials: Analysis and Solutions," *J. Amer. Statist. Assoc.*, **91**, 267 (1996).

[218] J. S. Simonoff, *Smoothing Methods in Statistics*, Springer-Verlag, New York, 1996.

[219] I. Fan and I. Gijbels, *Local Polynomial Modelling and Its Applications*, Chapman & Hall, London, 1996.

[220] A. Goldenshluger and A. Nemirovski, "On Spatial Adaptive Estimation of Nonparametric Regression," *Math. Meth. Stat.*, **6**, 135 (1997).

[221] A. W. Bowman and A. Azzalini, *Applied Smoothing Techniques for Data Analysis*, Oxford Univ. Press, New York, 1997.

[222] C. M. Hurvich and J. S. Simonoff, "Smoothing Parameter Selection in Nonparametric Regression Using an Improved AIC Criterion," *J. Roy. Statist. Soc., Ser. B*, **60**, 271 (1998).

[223] C. R. Loader, "Bandwidth Selection: Classical or Plug-In?," *Ann. Statist.*, **27**, 415 (1999).

[224] C. Loader, *Local Regression and Likelihood*, Springer-Verlag, New York, 1999.

[225] V. Katkovnik, "A New method for Varying Adaptive Bandwidth Selection," *IEEE Trans. Signal Process.*, **47**, 2567 (1999).

[226] I. Horová, "Some Remarks on Kernels," *J. Comp. Anal. Appl.*, **2**, 253 (2000).

[227] W. R. Schucany, "An Overview of Curve Estimators for the First Graduate Course in Nonparametric Statistics," *Statist. Sci.*, **19**, 663 (2004).

[228] C. Loader, "Smoothing: Local Regression Techniques," in J. Gentle, W. Härdle, and Y. Mori, eds., *Handbook of Computational Statistics*, Springer-Verlag, Heidelberg, 2004.

[229] V. Katkovnik, K. Egiazarian, and J. Astola, *Local Approximation Techniques in Signal and Image Processing*, SPIE Publications, Bellingham, WA, 2006.

[230] Data available from: `http://www.netlib.org/a/dloess`. Original source: N. D. Brinkman, "Ethanol - A Single-Cylinder Engine Study of Efficiency and Exhaust Emissions," *SAE Transactions*, **90**, 1410 (1981).

[231] Data available from `http://fedc.wiwi.hu-berlin.de/databases.php`, (MD*Base collection). Original source: Ref. [202] and G. Schmidt, R. Mattern, and F. Schüller, EEC Res. Program on Biomechanics of Impacts, Final report, Phase III, Project 65, Inst. für Rechtsmedizin, Univ. Heidelberg, Germany.

Exponential Smoothing

[232] R. G. Brown, *Smoothing, Forecasting and Prediction of Discrete-Time Series*, Prentice Hall, Englewood-Cliffs, NJ, 1962.

[233] D. C. Montgomery and L. A. Johnson, *Forecasting and Time Series Analysis*, McGraw-Hill, New York, 1976.

[234] C. D. Lewis, *Industrial and Business Forecasting Methods*, Butterworth Scientific, London, 1982.

[235] B. Abraham and J. Ledolter, *Statistical Methods for Forecasting*, Wiley, New York, 1983.

[236] S. Makridakis, et al., *The Forecasting Accuracy of Major Time Series Models*, Wiley, New York, 1983.

[237] S. Makridakis, S. C. Wheelwright, and R. J. Hyndman, *Forecasting, Methods and Applications*, 3/e, Wiley, New York, 1998.

[238] C. Chatfield, *Time Series Forecasting*, Chapman & Hall/CRC Press, Boca Raton, FL, 2001.

[239] R. J. Hyndman, A. B. Koehler, J. K. Ord, and R. D. Snyder, *Forecasting with Exponential Smoothing*, Springer-Verlag, Berlin, 2008.

[240] C. C. Holt, "Forecasting Seasonals and Trends by Exponentially Weighted Moving Averages," Office of Naval Research memorandum (ONR 52), 1957, reprinted in *Int. J. Forecast.*, **20**, 5 (2004); see also, *ibid.*, **20**, 11 (2004).

[241] P. R. Winters, "Forecasting Sales by Exponentially Weighted Moving Averages," *Manag. Sci.*, **6**, 324 (1960).

[242] J. F. Muth, "Optimal Properties of Exponentially Weighted Forecasts," *J. Amer. Statist. Assoc.*, **55**, 299 (1960).

[243] R. G. Brown and R. F. Meyer, "The Fundamental Theorem of Exponential Smoothing," *Oper. Res.*, **9**, 673 (1961).

[244] D. A. D'Esopo, "A Note on Forecasting by the Exponential Smoothing Operator," *Oper. Res.*, **9**, 686 (1961).

[245] D. R. Cox, "Prediction by Exponentially Weighted Moving Averages and Related Methods," *J. Roy. Statist. Soc., Ser. B*, **23**, 414 (1961).

[246] R. H. Morris and C. R. Glassey, "The Dynamics and Statistics of Exponential Smoothing Operators," *Oper. Res.*, **11**, 561 (1963).

[247] H. Theil and S. Wage, "Some Observations on Adaptive Forecasting," *Manag. Sci.*, **10**, 198 (1964).

[248] P. J. Harrison, "Short-Term Sales Forecasting," *Appl. Statist.*, **14**, 102 (1965).

[249] P. J. Harrison, "Exponential Smoothing and Short-Term Sales Forecasting," *Manag. Sci.*, **13**, 821 (1967).

[250] W. G. Gilchrist, "Methods of Estimation Involving Discounting," *J. Roy. Satist. Soc., Ser. B*, **29**, 355 (1967).

[251] C. C. Pegels, "Exponential Forecasting: Some New Variations," *Manag. Sci.*, **15**, 311 (1969).

[252] A. C. Watts, "On Exponential Smoothing of Discrete Time Series," *IEEE TRans. Inform. Th.*, **16**, 630 (1970).

[253] K. O. Cogger, "The Optimality of General-Order Exponential Smoothing," *Oper. Res.*, **22**, 858 (1974).

[254] S. D. Roberts and D. C. Whybark, "Adaptive Forecasting Techniques," *Int. J. Prod. Res.*, **12**, 635 (1974).

[255] M. L. Goodman, "A New Look at Higher-Order Exponential Smoothing for Forecasting," *Oper. Res.*, **22**, 880 (1974).

[256] D. E. Clapp, "Adaptive Forecasting with Orthogonal Polynomial Models," *AIIE Trans.*, **6**, 359 (1974).

[257] J. W. Tukey, *Exploratory Data Analysis*, Addison-Wesley, Reading, MA, 1977.

[258] J. F. Kaiser and R. W. Hamming, "Sharpening the Response of a Symmetric Nonrecursive Filter by the Multiple Use of the same Filter," *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-25**, 415 (1977).

[259] E. Mckenzie, "The Monitoring of Exponentially Weighted Forecasts," *J. Oper. Res. Soc.*, **29**, 449 (1978).

[260] C. Chatfield, "The Holt-Winters Forecasting Procedure," *Appl. Statist.*, **27**, 264 (1978).

[261] R. Fildes, "Quantitative Forecasting—The State of the Art: Extrapolative Methods," *J. Oper. Res. Soc.*, **30**, 691 (1979).

[262] S. Ekern, "Adaptive Exponential Smoothing Revisited," *J. Oper. Res. Soc.*, **32**, 775 (1981).

[263] S. A. Roberts, "A General Class of Holt-Winters Type Forecasting Models," *Manag. Sci.*, **28**, 808 (1982).

[264] E. J. Muth, "The Discrete Laguerre Polynomials and their Use in Exponential Smoothing," *IIE Trans.*, **15**, 166 (1983).

[265] E. S. Gardner, Jr., "Exponential Smoothing: The State of the Art," *J. Forecast.*, **4**, 1 (1985).

[266] B. Abraham and J. Ledolter, "Forecast Functions Implied by Autoregressive Integrated Moving Average Models and Other Related Forecast Procedures," *Int. Statist. Rev.*, **54**, 51 (1986).

[267] D. J. Dalrymple, "Sales Forecasting Practices: Results from a United States Survey," *Int. J. Forecast.*, **3**, 379 (1987).

[268] C. Chatfield and M. Yar, "Holt-Winters Forecasting: Some Practical Issues," *Statistician*, **37**, 129 (1988).

[269] E. Yashchin, "Estimating the Current Mean of a Process Subject to Abrupt Changes," *Technometrics*, **37**, 311 (1995).

[270] S. Satchell and A. Timmermann, "On the Optimality of Adaptive Expectations: Muth Revisited," *Int. J. Forecast.*, **11**, 407 (1995).

[271] H. Winklhofer, A. Diamantopoulos, and S. F. Witt, "Forecasting practice: A Review of the Empirical Literature and an Agenda for Future Research," *Int. J. Forecast.*, **12**, 193 (1996).

[272] S. Makridakis and M. Hibon, "The M3-Competition: Results, Conclusions and Implications," *Int. J. Forecast.*, **16**, 451 (2000).

[273] C. Chatfield, et al., "A New Look at Models for Exponential Smoothing," *Statistician*, **50**, 147 (2001).

[274] A. Chen and E. A. Elsayed, "Design and Performance Analysis of the Exponentially Weighted Moving Average Mean Estimate for Processes Subject to Random Step Changes," *Technometrics*, **44**, 379 (2002).

[275] D. J. Robb and E. A. Silver, "Using Composite Moving Averages to Forecast Sales," *J. Oper. Res. Soc.*, **53**, 1281 (2002).

[276] J. W. Taylor, "Smooth Transition Exponential Smoothing," *J. Forecast.*, **23**, 385 (2004).

[277] E. S. Gardner, Jr., "Exponential Smoothing: The State of the Art—Part II," *Int. J. Forecast.*, **22**, 239 (2006).

[278] B. Billah, et al., "Exponential Smoothing Model Selection for Forecasting," *Int. J. Forecast.*, **22**, 239 (2006).

[279] J. G. De Gooijer and R. J. Hyndman, "25 Years of Time Series Forecasting," *Int. J. Forecast.*, **22**, 443 (2006).

Technical Analysis in Financial Market Trading

[280] S. B. Achelis, *Technical Analysis from A to Z*, 2nd ed., McGraw-Hill, NY, 2001.

[281] J. W. Wilder, *New Concepts in Technical Trading Systems*, Trend Research, Greensboro, NC, 1978.

[282] "Surviving The Test of Time With J. Welles Wilder," interview by B. Twomey, *Tech. Anal. Stocks & Commod.*, **27**, no.3, 58 (2009).

[283] T. S. Chande and S. Kroll, *The New Technical Trader*, Wiley, NY, 1994.

[284] J. F. Ehlers, *Rocket Science for Traders*, Wiley, NY, 2001.

[285] J. F. Ehlers, *Cybernetic Analysis for Stocks and Futures*, Wiley, NY, 2004.

[286] P. J. Kaufman, *New Trading Systems and Methods*, 4/e, Wiley, 2005.

[287] D. K. Mak, *Mathematical Techniques in Financial Market Trading*, World Scientific, Singapore, 2006.

[288] *Technical Analysis*, PDF book, 2011, Creative Commons Attribution-Share, available from: `https://www.mrao.cam.ac.uk/~mph/Technical_Analysis.pdf`

[289] International Federation of Technical Analysts, `www.ifta.org`

[290] V. Zakamulin, *Market Timing with Moving Averages*, Palgrave Macmillan, 2017. See also by same author, "Moving Averages for Market Timing,", Oct. 2016. Available at SSRN: `https://ssrn.com/abstract=2854180`

[291] D. Penn, "The Titans Of Technical Analysis," *Tech. Anal. Stocks & Commod.*, **20**, no.10, 32 (2002).

[292] A. W. Lo and J. Hasanhodzic, *The Heretics of Finance*, Bloomberg Press, NY, 2009.

[293] M. Carr and A. Hestla, "Technical Analysis Adapts and Thrives," *Tech. Anal. Stocks & Commod.*, **29**, no.4, 46 (2011).

[294] J. K. Hutson, "Good Trix", *Tech. Anal. Stocks & Commod.*, **1**, no.5, 105, (1983); *ibid.*, **2**, no.2, 91, (1984). See also, D. Penn, "TRIX", *Tech. Anal. Stocks & Commod.*, **29**, no.9, 197, (2003).

[295] R. Barrons Roosevelt, "Metaphors For Trading," *Tech. Anal. Stocks & Commod.*, **16**, no.2, 67 (1998).

[296] T. S. Chande, "Adapting Moving Averages to Market Volatility," *Tech. Anal. Stocks & Commod.*, **10**, no.3, 108 (1992).

[297] P. G. Mulloy, "Smoothing Data with Faster Moving Averages," *Tech. Anal. Stocks & Commod.*, **12**, no.1, 11 (1994).

[298] P. G. Mulloy, "Smoothing Data with Less Lag," *Tech. Anal. Stocks & Commod.*, **12**, no.2, 72 (1994).

[299] T. S. Chande, "Forecasting Tomorrow's Trading Day," *Tech. Anal. Stocks & Commod.*, **10**, no.5, 220 (1992).

[300] P. E. Lafferty, "The End Point Moving Average," *Tech. Anal. Stocks & Commod.*, **13**, no.10, 413 (1995).

[301] D. Kraska, "The End Point Moving Average,", Letters to *Tech. Anal. Stocks & Commod.*, **14**, Feb. (1996).

[302] J. F. Ehlers, "Zero-Lag Data Smoothers," *Tech. Anal. Stocks & Commod.*, **20**, no.7, 26 (2002). See also, J. F. Ehlers and R. Way, "Zero Lag (Well, Almost)," *ibid.*, **28**, 30, Nov. (2010).

[303] W. Rafter, "The Moving Trend," *Tech. Anal. Stocks & Commod.*, **21**, no.1, 38 (2003).

[304] D. Meyers, "Surfing the Linear Regression Curve with Bond Futures," *Tech. Anal. Stocks & Commod.*, **16**, no.5, 209 (1998).

[305] B. Star, "Confirming Price Trend," *Tech. Anal. Stocks & Commod.*, **25**, no.13, 72 (2007).

[306] P. E. Lafferty, "How Smooth is Your Data Smoother?," *Tech. Anal. Stocks & Commod.*, **17**, no.6, 251 (1999).

[307] T. Tillson, "Smoothing Techniques For More Accurate Signals," *Tech. Anal. Stocks & Commod.*, **16**, no.1, 33 (1998).

[308] J. Sharp, "More Responsive Moving Averages," *Tech. Anal. Stocks & Commod.*, **18**, no.1, 56 (2000).

[309] A. Hull, "How to reduce lag in a moving average," `https://alanhull.com/hull-moving-average`.

[310] B. Star, "Detecting Trend Direction and Strength," *Tech. Anal. Stocks & Commod.*, **20**, no.1, 22 (2007).

[311] S. Evens, "Momentum And Relative Strength Index," *Tech. Anal. Stocks & Commod.*, **17**, no.8, 367 (1999).

[312] S. Evens, "Stochastics," *Tech. Anal. Stocks & Commod.*, **17**, no.9, 392 (1999).

[313] P. Roberts, "Moving Averages: The Heart of Trend Analysis," *Alchemist*, **33**, 12 (2003), Lond. Bullion Market Assoc., available online from: `www.lbma.org.uk`.

[314] K. Edgeley "Oscillators Go with the Flow," *Alchemist*, **37**, 17 (2005), Lond. Bullion Market Assoc., available online from: `www.lbma.org.uk`.

[315] D. Penn, "Moving Average Trios," *Tech. Anal. Stocks & Commod.*, **25**, no.9, 54 (2007).

[316] B. Star, "Trade the Price Swings," *Tech. Anal. Stocks & Commod.*, **21**, no.12, 68 (2003).

[317] A. Sabodin, "An MACD Trading System," *Tech. Anal. Stocks & Commod.*, **26**, no.3, 12 (2008).

[318] C. K. Langford, "Three Common Tools, One Protocol," *Tech. Anal. Stocks & Commod.*, **26**, no.10, 48 (2008).

[319] H. Seyedinajad, "The RSI Miracle," *Tech. Anal. Stocks & Commod.*, **27**, no.1, 12 (2009).

[320] M. Alves, "Join the Band: Applying Hysteresis to Moving Averages," *Tech. Anal. Stocks & Commod.*, **27**, no.1, 36 (2009).

[321] E. Donie, "An MACD Parallax View," *Tech. Anal. Stocks & Commod.*, **27**, no.4, 12 (2009).

[322] R. Singh and A. Kumar, "Intelligent Stock Trading Technique using Technical Analysis," *Int. J. Mgt. Bus. Studies*, **1**, 46 (2011).

[323] J. Bollinger, "Using Bollinger Bands," *Tech. Anal. Stocks & Commod.*, **10**, no.2, 47 (1992).

[324] S. Evens, "Bollinger Bands," *Tech. Anal. Stocks & Commod.*, **17**, no.3, 116 (1999).

[325] S. Vervoort, "Smoothing the Bollinger %b," *Tech. Anal. Stocks & Commod.*, **28**, no.5, 40 (2010); and Part 2, *ibid.*, **28**, no.6, 48 (2010).

[326] J. Gopalakrishnan and B. Faber, "Interview: System Trading Made Easy With John Bollinger," *Tech. Anal. Stocks & Commod.*, **30**, no.3, 36 (2012).

[327] A. Mustapha, "Bollinger Bands & RSI: A Magical Combo," *Tech. Anal. Stocks & Commod.*, **34**, no.6, 18 (2016).

[328] M. Widner, "Signaling Change with Projection Bands," *Tech. Anal. Stocks & Commod.*, **13**, no.7, 275 (1995).

[329] J. Andersen, "Standard Error Bands," *Tech. Anal. Stocks & Commod.*, **14**, no.9, 375 (1996).

[330] S. Evens, "Keltner Channels," *Tech. Anal. Stocks & Commod.*, **17**, no.12, 533 (1999).

[331] D. Penn, "Donchian Breakouts," *Tech. Anal. Stocks & Commod.*, **20**, no.2, 34 (2002); and, "Building a Better Breakout,", *ibid.*, **21**, no.10, 74 (2003).

[332] B. Star, "Trade Breakouts And Retracements With TMV," *Tech. Anal. Stocks & Commod.*, **30**, no.2, 13 (2012).

[333] F. Bertrand, “RSI Bands,” *Tech. Anal. Stocks & Commod.*, **26**, no.4, 44 (2008).

[334] S. Lim, T. T.. Hisarli, and N, S. He, “Profitability of a Combined Signal Approach: Bollinger Bands and the ADX,” *IFTA J.*, p.23, 2014 edition, `https://ifta.org/publications/journal/`.

[335] P, Aan, “Parabolic Stop/Reversal,” *Tech. Anal. Stocks & Commod.*, **7**, no.11, 411 (1989).

[336] T. Hartle, “The Parabolic Trading System,”, *Tech. Anal. Stocks & Commod.*, **11**, no.11, 477 (1993).

[337] D. Meyers, “Modifying the Parabolic Stop And Reversal,” *Tech. Anal. Stocks & Commod.*, **14**, no.4, 152 (1995).

[338] J. Sweeney, “Parabolics,” *Tech. Anal. Stocks & Commod.*, **15**, no.7, 329 (1997).

[339] R. Teseo, “Stay in the Market with Stop-And-Reverse,” *Tech. Anal. Stocks & Commod.*, **20**, no.4, 76 (2002).

[340] K. Agostino and B. Dolan, “Make the Trend Your Friend in Forex,” *Tech. Anal. Stocks & Commod.*, **22**, no.9, 14 (2004).

[341] D. Sepiashvili, “The Self-Adjusting RSI,” *Tech. Anal. Stocks & Commod.*, **24**, no.2, 20 (2006).

[342] G. Siligardos, “Leader Of The MACD,” *Tech. Anal. Stocks & Commod.*, **26**, no.7, 24 (2008).

[343] M. J. Pring, “The Special K, Part 1,” *Tech. Anal. Stocks & Commod.*, **26**, no.12, 44 (2008); and Part 2, *ibid.*, **27**, no.1, 28 (2009); see also, *ibid.*, “Identifying Trends With The KST Indicator,” **10**, no.10, 420 (1992).

[344] P. Konner, “Combining RSI with RSI,” *Tech. Anal. Stocks & Commod.*, **29**, no.1, 16 (2011).

[345] *Fidelity's Technical Indicator Guide:*
`https://www.fidelity.com/learning-center/trading-investing/technical-analysis/technical-indicator-guide/overview`

OANDA Technical Indicator Guide and Tutorials:
`https://www.oanda.com/forex-trading/learn/forex-indicators`
`https://www.oanda.com/forex-trading/learn/technical-analysis-for-traders`

[346] *TradingView Wiki:*
`https://www.tradingview.com/wiki`

[347] A. Raudys, V. Lenciauskas, and E. Malcius, “Moving Averages for Financial Data Smoothing,” in T. Skersys, R. Butleris, and R. Butkiene (Eds.), *Proceedings Information and Software Technologies*, 19th Int. Conf., ICIST 2013; paper available online from,
`https://pdfs.semanticscholar.org/257b/837649d8b50662b3fe2c21fce825a1c184e5.pdf`

[348] C. W. Gross and J. E. Sohl, “Improving Smoothing Models with an Enhanced Initialization scheme,” *J. Bus. Forecasting*, **8**, 13 (1989).

[349] J. R. Taylor, *Introduction to Error Analysis*, Oxford University Press, University Science Books, Mill Valley, CA.

Spline Smoothing

[350] `http://pages.cs.wisc.edu/~deboor/bib/`, extensive online spline bibliography.

[351] G. Wahba, *Spline Models for Observational Data*, SIAM Publications, Philadelphia, 1990.

[352] P. J. Green and B. W. Silverman, *Nonparametric Regression and Generalized Linear Mofdels: A Roughness Penalty Approach*, Chapman & Hall, London, 1994.

[353] R. L. Eubank, *Spline Smoothing and Nonparametric Regression*, Marcel Dekker, New York, 1988.

[354] I. M. Gelfand and S. V. Fomin, *Calculus of Variations*, Dover Publications, Mineola, NY, 2000; reprint of 1963 Prentice Hall edition.

[355] J. Stoer and R. Bulirsch, *Introduction to Numerical Analysis*, Springer-Verlag, New York, 1980.

[356] J. L. Walsh, J. H. Ahlberg, and E. N. Nilson, “Best Approximation Properties of the Spline Fit,” *J. Math. Mech.*, **11**, 225 (1962).

[357] I. J. Schoenberg, “Spline Functions and the Problem of Graduation,” *Proc. of the Nat. Acad. Sci.*, **52**, no.4, 947 (1964).

[358] C. H. Reinsch, “Smoothing by Spline Functions,” *Numer. Mathematik*, **10**, 177 (1967), and “Smoothing by Spline Functions. II,” *ibid.*, **16**, 451 (1971).

[359] P. M. Anselone and P. J. Laurent, “A General Method for the Construction of Interpolating or Smoothing Spline-Functions,” *Numer. Math.*, **12**, 66 (1968).

[360] D. Kershaw, “The Explicit Inverses of Two Commonly Occurring Matrices,” *Math. Comp.*, **23**. 189 (1969).

[361] A. M. Erisman and W. F. Tinney, “On Computing Certain Elements of the Inverse of a Sparse Matrix,” *Commun. ACM*, **18**, 177 (1975).

[362] S. Wold, “Spline Functions in Data Analysis,” *Technometrics*, **16**, 1 (1974).

[363] L. L. Horowitz, “The Effects of Spline Interpolation on Power Spectral Density,” *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-22**, 22 (1974).

[364] L. D'Hooge, J. De Kerf, and M. J. Goovaerts, “Adjustment of Mortality Tables by Means of Smoothing Splines,” *Bulletin de l'Association Royale des Actuaires Belge*, **71**, 78 (1976).

[365] D. L. Jupp, “B-Splines for Smoothing and Differentiating Data Sequences,” *Math. Geol.*, **8**, 243 (1976).

[366] C.S. Duris, “Discrete Interpolating and Smoothing Spline Functions,” *SIAM J. Numer. Anal.*, **14**, 686 (1977), and “Fortran Routines for Discrete Cubic Spline Interpolation and Smoothing,” *ACM Trans. Math. Softw.*, **6**, 92 (1980).

[367] H. S. Hou and H. C. Andrews, “Cubic Splines for Image Interpolation and Digital Filtering,” *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-26**, 508 (1978).

[368] G. H. Golub, M. Heath, and G. Wahba, “Generalized Cross-Validation as a Method for Choosing a Good Ridge Parameter,” *Technometrics*, **21**, 215 (1979).

[369] P. Craven and G. Wahba, “Smoothing by Spline Functions, Estimating the Correct Degree of Smoothing by the Method of Generalized Cross-Validation,” *Numer. Math.*, **31**, 377 (1979).

[370] P. L. Smith, “Splines as a Useful and Convenient Statistical Tool,” *Amer. Statist.*, **33**, 57 (1979).

[371] R. G. Keys, “Cubic Convolution Interpolation for Digital Image Processing,” *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-29**, 1153 (1981).

[372] J. McCutcheon, “Some Remarks on Splines,”, *Trans. Fac. Actuaries*, **37**, 421 (1981).

[373] C. L. Vaughan, “Smoothing and Differentiation of Displacement-Time Data: Application of Splines and Digital Filtering,” *Int. J. Bio-Med. Comput.*, **13**, 375 (1982).

[374] E. J. Wegman and I. W. Wright, “Splines in Statistics,” *J. Amer. Statist. Assoc.*, **78**, 351 (1983).

[375] B. K. P. Horn, “The Curve of Least Energy,” *ACM Trans. Math. Softw.*, **9**, 441 (1983).

[376] B. W. Silverman, “A Fast and Efficient Cross-Validation Method for Smoothing Parameter Choice in Spline Regression,” *J. Amer. Statist. Assoc.*, **79**, 584 (1984).

[377] M. F. Hutchison and F. R. de Hoog, “Smoothing Noisy Data with Spline Functions,” *Numer. Math.*, **47**, 99 (1985).

[378] B. W. Silverman, “Some Aspects of the Spline Smoothing Approach to Non-Parametric Regression Curve Fitting,” *J. Roy. Statist. Soc., Ser. B*, **47**, 1 (1985).

[379] P. H. C. Eilers and B. D. Marx, “Flexible Smoothing with B-Splines and Penalties,” *Statist. Sci.*, **11**, 89 (1989).

[380] K. F. Üstüner and L. A. Ferrari, “Discrete Splines and Spline Filters,” *IEEE Trans. Circ. Syst.—II*, **39**, 417 (1992).

[381] M. A. A. Moussa and M. Y. Cheema, “Non-Parametric Regression in Curve Fitting,” *Statistician*, **41** 209 (1992).

[382] M. Unser, A. Aldroubi, and M. Eden, “B-Spline Signal Processing: Part I—Theory,” *IEEE Trans. Signal Process.*, **41**, 821 (1993), and “Part II—Efficient Design and Applications,” *ibid.*, p. 834.

[383] R. L. Eubank, “A Simple Smoothing Spline,” *Amer. Statist.*, **48**, 103 (1994).

[384] D. Nychka, “Splines as Local Smoothers,” *Ann. Statist.*, **23**, 1175 (1995).

[385] M. Unser, “Splines, A Perfect Fit for Signal and Image Processing,” *IEEE Sig. Process. Mag.*, **16**, no.6, 22, (1999).

[386] R. Champion, C. T. Lenard, and T. M. Mills, “A Variational Approach to Splines,” *ANZIAM J.*, **42**, 119 (2000).

[387] V. Solo, “A Simple Derivation of the Smoothing Spline,” *Amer. Statist.*, **54**, 40 (2000).

[388] S. Sun, M. B. Egerstedt, and C. F. Martin, “Control Theoretic Smoothing Splines,” *IEEE Trans. Autom. Contr.*, **45**, 2271 (2000).

[389] H. Bachau, et al., "Applications of B-Splines in Atomic and Molecular Physics," *Rep. Prog. Phys.*, **64**, 1815 (2001).

[390] S. A. Dyer and J. S. Dyer, "Cubic-Spline Interpolation, Part 1,", *IEEE Instr. & Meas. Mag.*, March 2001, p. 44, and "Part 2,", *ibid.*, June 2001, p.34.

[391] J. D. Carew, et al., "Optimal Spline Smoothing of fMRI Time Series by Generalized Cross-Validation," *NeuroImage*, **18**, 950 (2003).

[392] A. K. Chaniotis and D. Poulikakos, "High Order Interpolation and Differentiation Using B-Splines," *J. Comput. Phys.*, **197**, 253 (2004).

[393] P. H. C. Eilers, "Fast Computation of Trends in Scatterplots," *Kwantitatieve Meth.*, **71**, 38 (2004).

[394] T. C. M. Lee, "Improved Smoothing Spline Regression by Combining Estimates of Different Smoothness," *Statist. Prob. Lett.*, **67**, 133 (2004).

[395] M. Unser and T. Blu, "Cardinal Exponential Splines: Part I—Theory and Filtering Algorithms," *IEEE Trans. Signal Process.*, **53**, 1425 (2005), and M. Unser, "Cardinal Exponential Splines: Part II—Think Analog, Act Digital," *ibid.*, p. 1439.

[396] H. L. Weinert, "A Fast Compact Algorithm for Cubic Spline Smoothing," *Comput. Statist. Data Anal.*, **53**, 932 (2009).

[397] G. Kimeldorf and G. Wahba, "A Correspondence Between Bayesian Estimation on Stochastic Processes and Smoothing by Splines," *Ann. Math. Statist.*, **41**, 495 (1970).

[398] G. Kimeldorf and G. Wahba, "Some Results on Tschebycheffian Spline Functions," *J. Math. Anal. Appl.*, **33**, 82 (1971).

[399] G. Wahba, "Improper Priors, Spline Smoothing and the Problem of Guarding Against Model Errors in Regression," *J. Roy. Statist. Soc., Ser. B*, **40**, 364 (1978).

[400] H. L. Weinert and G. S. Sidhu, "A Stochastic Framework for Recursive Computation of Spline Functions: Part II, Interpolating Splines," *IEEE Trans. Inform. Th.*, **24**, 45 (1978).

[401] H. L. Weinert, R. H. Byrd, and G. S. Sidhu, "A Stochastic Framework for Recursive Computation of Spline Functions: Part II, Smoothing Splines," *J. Optim. Th. Appl.*, **30**, 255 (1980).

[402] W. E. Wecker and C. F. Ansley, "The Signal Extraction Approach to Nonlinear Regression and Spline Smoothing," *J. Amer. Statist. Assoc.*, **78**, 81 (1983).

[403] R. Kohn and C. F. Ansley, "A New Algorithm for Spline Smoothing Based on Smoothing a Stochastic Process," *SIAM J. Stat. Comput.*, **8**, 33 (1987).

[404] R. Kohn and C. F. Ansley, "A Fast Algorithm for Signal Extraction, Influence and Cross-Validation in State Space Models, *Biometrika*, **76**, 65 (1989).

Whittaker-Henderson Smoothing

[405] A. Hald, "T. N. Thiele's Contributions to Statistics," *Int. Statist. Rev.*, **49**, 1 (1981), with references to Thiele's works therein.

[406] S. L. Lauritzen, "Time Series Analysis in 1880: A Discussion of Contributions Made by T. N. Thiele," *Int. Statist. Rev.*, **49**, 319 (1981). Reprinted in S. L. Lauritzen, ed., *Thiele: Pioneer in Statistics*, Oxford Univ. Press, Oxford, New York, 2002.

[407] G. Bohlmann, "Ein Ausgleichungsproblem," *Nachrichten Gesellschaft Wissenschaften zu Göttingen, Mathematische-Physikalische Klasse*, no.3, p.260, (1899).

[408] E. Whittaker, "On a New Method of Graduation," *Proc. Edinburgh Math. Soc.*, **41**, 63 (1923).

[409] E. Whittaker, "On the Theory of Graduation," *Proc. Roy. Soc. Edinburgh*, **44**, 77 (1924).

[410] E. Whittaker and G. Robinson, *The Calculus of Observations*, Blackie & Son, London, 1924.

[411] R. Henderson, "A New Method of Graduation," *Trans. Actuarial Soc. Am.*, **25**, 29 (1924).

[412] R. Henderson, "Further Remarks on Graduation," *Trans. Actuarial Soc. Am.*, **26**, 52 (1925).

[413] A. C. Aitken, "On the Theory of Graduation," *Proc. Roy. Soc. Edinburgh*, **46**, 36 (1925).

[414] A. W. Joseph, "The Whittaker-Henderson Method of Graduation," *J. Inst. Actuaries*, **78**, 99 (1952).

[415] C. E. V. Leser, "A Simple Method of Trend Construction," *J. Roy. Statist. Soc., Ser. B*, **23**, 91 (1961).

[416] A. W. Joseph, "Subsidiary Sequences for Solving Leser's Least-Squares Graduation Equations," *J. Roy. Statist. Soc., Ser. B*, **24**, 112 (1962).

[417] G. S. Kimeldorf and D. A. Jones, "Bayesian Graduation," *Trans. Soc. Actuaries*, **19**, Pt.1, 66 (1967).

[418] R. J. Shiller, "A Distributed Lag Estimator Derived from Smoothness Priors," *Econometrica*, **41**, 775 (1973).

[419] B. D. Cameron, et al., "Some Results of Graduation of Mortality Rates by the Whittaker-Henderson and Spline Fitting Methods," *Bulletin de l'Association Royale des Actuaires Belge*, **71**, 48 (1976).

[420] G. Taylor, "A Bayesian Interpretation of Whittaker-Henderson Graduation," *Insurance: Math. & Econ.*, **11**, 7 (1992).

[421] R. J. Verrall, "A State Space Formulation of Whittaker Graduation, with Extensions," *Insurance: Math. & Econ.*, **13**, 7 (1993).

[422] D. R. Schuette, "A Linear Programming Approach to Graduation", *Trans. Soc. Actuaries*, **30**, 407 (1978); with Discussions, *ibid.*, pp. 433, 436, 440, 442, 443.

[423] F. Y. Chan, et al., "Properties and modifications of Whittaker-Henderson graduation," *Scand. Actuarial J.*, **1982**, 57 (1982).

[424] F. Y. Chan, et al., "A generalization of Whittaker-Henderson graduation," *Trans. Actuarial Soc. Am.*, **36**, 183 (1984).

[425] F. Y. Chan, et al., "Applications of linear and quadratic programming to some cases of the Whittaker-Henderson graduation method," *Scand. Actuarial J.*, **1986**, 141 (1986).

[426] G. Mosheiov and A. Raveh, "On Trend Estimation of Time Series: A Simple Linear Programming Approach," *J. Oper. Res. Soc.*, **48**, 90 (1997).

[427] R. J. Brooks, et al., "Cross-validatory graduation," *Insurance: Math. Econ.*, **7**, 59 (1988).

[428] P. H. C. Eilers, "A Perfect Smoother," *Anal. Chem.*, **75**, 3631 (2003).

[429] W. E. Diewert and T. J. Wales, "A 'New' Approach to the Smoothing Problem," in M. T. Belongia and J. M. Binner, eds., *Money, Measurement and Computation*, Palgrave Macmillan, New York, 2006.

[430] H.L. Weinert, "Efficient Computation for Whittaker-Henderson Smoothing," *Comput. Statist. Data Anal.*, **52**, 959 (2007).

[431] T. Alexandrov, et al. "A Review of Some Modern Approaches to the Problem of Trend Extraction," US Census, Statistics Report No. 2008-3, available online from `http://www.census.gov/srd/papers/pdf/rrs2008-03.pdf`.

[432] A. S. Nocon and W. F. Scott, "An extension of the Whittaker-Henderson method of graduation," *Scand. Actuarial J.*, **2012**, 70 (2012).

[433] J. Vondrák, "A Contribution to the Problem of Smoothing Observational Data," *Bull. Astron. Inst. Czech.*, **20**, 349 (1969).

[434] J. Vondrák, "Problem of Smoothing Observational Data II," *Bull. Astron. Inst. Czech.*, **28**,84 (1977).

[435] J. Vondrák and A. Čepek, "Combined Smoothing Method and its Use in Combining Earth Orientation Parameters Measured by Space Techniques," *Astron. Astrophys. Suppl. Ser.*, **147**, 347 (2000).

[436] D. W. Zheng, et al., "Filtering GPS Time-Series using a Vondrak Filter and Cross-Validation," *J. Geodesy*, **79**, 363 (2005).

[437] Z-W Li, et al., "Least Squares-Based Filter for Remote Sensing Image Noise Reduction," *IEEE Trans. Geosci. Rem. Sens.*, **46**, 2044 (2008).

[438] Z-W Li, et al., "Filtering Method for SAR Interferograms with Strong Noise," *Int. J. Remote Sens.*, **27**, 2991 (2006).

Hodrick-Prescott and Bandpass Filters

[439] R. J. Hodrick and E. C. Prescott, "Postwar U.S. Business Cycles: An Empirical Investigation," *J. Money, Credit & Banking*, **29**, 1 (1997); earlier version: Carnegie-Mellon Univ., Discussion Paper No. 451, (1980).

[440] M. Unser, A. Aldroubi, and M. Eden, "Recursive Regularization Filters: Design, Properties, and Applications," *IEEE Trans. Patt. Anal. Mach. Intell.*, **13**, 272 (1991).

[441] A. C. Harvey and A. Jaeger, "Detrending, Stylized Facts and the Business Cycle," *J. Appl. Econometr.*, **8**, 231 (1993).

[442] R. G. King and S. T. Rebelo, "Low Frequency Filtering and Real Business Cycles," *J. Econ. Dynam. Contr.*, **17**, 207 (1993), and appendix available online from `http://www.kellogg.northwestern.edu/faculty/rebelo/htm/LFF-Appendix.pdf`.

[443] T. Cogley and J. M. Nason, "Effects of the Hodrick-Prescott Filter on Trend and Difference Stationary Time Series. Implications for Business Cycle Research," *J. Econ. Dynam. Contr.*, **19**, 253 (1995).

[444] J. Ehlgen, "Distortionary Effects of the Optimal Hodrick-Prescott Filter," *Econ. Lett.*, **61**, 345 (1998).

[445] U. Woitech, "A Note on the Baxter-King Filter," Dept. Econ., Univ. Glasgow, Working Paper, No. 9813, 1998, `http://www.gla.ac.uk/media/media_22357_en.pdf`.

[446] M. Baxter and R. G. King, "Measuring Business Cycles: Approximate Band-Pass Filters for Economic Time Series," *Rev. Econ. Stat.*, **81**, 575 (1999).

[447] Y. Wen and B. Zeng, "A Simple Nonlinear Filter for Economic Time Series Analysis," *Econ. Lett.*, **64**, 151 (1999).

[448] M. Bianchi, M. Boyle, and D. Hollingsworth, "A Comparison of Methods for Trend Estimation," *Appl. Econ. Lett.*, **6**, 103 (1999).

[449] P. Young and D. Pedregal, "Recursive and En-Bloc Approaches to Signal Extraction," *J. Appl. Statist.*, **26**, 103 (1999).

[450] J. J. Reeves, et al., "The Hodrick-Prescott Filter, a Generalization, and a New Procedure for Extracting an Empirical Cycle from a Series," *Stud. Nonlin. Dynam. Econometr.*, **4**, 1 (2000).

[451] D. S. G. Pollock, "Trend Estimation and De-Trending via Rational Square-Wave Filters," *J. Econometr.*, **99**, 317 (2000).

[452] V. Gómez, "The Use of Butterworth Filters for Trend and Cycle Estimation in Economic Time Series," *J. Bus. Econ. Statist.*, **19**, 365 (2001).

[453] T. M. Pedersen, "The Hodrick-Prescott Filter, the Slutzky Effect, and the Distortionary Effect of Filters," *J. Econ. Dynam. Contr.*, **25**, 1081 (2001).

[454] E. Slutzky, "The Summation of Random Causes as the Source of Cyclic Processes," *Econometrica*, **37**, 105 (1937).

[455] V. M. Guerrero, R. Juarez, and P. Poncela, "Data Graduation Based on Statistical Time Series Methods," *Statist. Probab. Lett.*, **52**, 169 (2001).

[456] M. O. Ravn and H. Uhlig, "On Adjusting the Hodrick-Prescott Filter for the Frequency of Observations," *Rev. Econ. Statist.*, **84**, 371 (2002).

[457] C. J. Murray, "Cyclical Properties of Baxter-King Filtered Time Series," *Rev. Econ. Statist.*, **85**, 472 (2003).

[458] A. C. Harvey and T. M . Trimbur, "General Model-Based Filters for Extracting Cycles and Trends in Economic Time Series," *Rev. Econ. Statist.*, **85**, 244 (2003).

[459] L. J. Christiano + T. J. Fitzgerald, "The Band Pass Filter," *Int. Econ. Rev.*, **44**, 435 (2003).

[460] A. Iacobucci and A. Noullez, "A Frequency Selective Filter for Short-Length Time Series," *Comput. Econ.*, **25**, 75 (2005).

[461] A. Guay and P. St.-Amant, "Do the Hodrick-Prescott and Baxter-King Filters Provide a Good Approximation of Business Cycles?," *Ann. Économie Statist.*, No. 77, p. 133, Jan-Mar. 2005.

[462] T. M. Trimbur, "Detrending Economic Time Series: A Bayesian Generalization of the Hodrick-Prescott Filter," *J. Forecast.*, **25**, 247 (2006).

[463] A. Maravall, A. del Río, "Temporal Aggregation, Systematic Sampling, and the Hodrick-Prescott Filter," *Comput. Statist. Data Anal.*, **52**, 975 (2007).

[464] V. M. Guerrero, "Estimating Trends with Percentage of Smoothness Chosen by the User," *Int. Statist. Rev.*, **76**, 187 (2008).

[465] T. McElroy, "Exact Formulas for the Hodrick-Prescott Filter," *Econometr. J.*, **11**, 209 (2008).

[466] D. E. Giles, "Constructing confidence bands for the Hodrick-Prescott filter," *Appl. Econ. Letters*, **20**, 480 (2013).

[467] D. S. G. Pollock, "Econometric Filters," *Comput. Econ.*, **48**, 669 (2016).

L_1 Trend Filtering

[468] S-J. Kim, et al., "ℓ_1 Trend Filtering," *SIAM Rev.*, **51**, 339 (2009).

[469] A. Moghtaderi, P. Borgnat, and P. Flandrin, "Trend Filtering: Empirical Mode Decompositions Versus ℓ_1 and Hodrick-Prescott," *Adv. Adaptive Data Anal.*, **3**, 41 (2011).

[470] B. Wahlberg, C. R. Rojas, and M. Annergren, "On ℓ_1 Mean and Variance Filtering," *2011 Conf. Record 45th Asilomar Conf. Signals, Systems and Computers*, (ASILOMAR), IEEE, p. 1913, (2011).

[471] R. J. Tibshirani, "Adaptive piecewise polynomial estimation via trend filtering," *Ann. Stat.*. **42**, 285 (2014).

[472] Y-X Wang, et al., "Trend Filtering on Graphs," *Proc. 18th Int. Conf. Artif. Intell. Stat.* (AISTATS), p. 1042, May 2015.

[473] A. Ramdas and R. J. Tibshirani, "Fast and Flexible ADMM Algorithms for Trend Filtering," *J. Comput. Graph. Stat.*, **25**, 839 (2016).

[474] H. Yamada and L. Jin, "Japan's output gap estimation and ℓ_1 trend filtering," *Empir. Econ.*, **45**, 81 (2013).

[475] H. Yamada, "Estimating the trend in US real GDP using the ℓ_1 trend filtering," *Appl. Econ. Letters*, 2016, p. 1.

[476] H. Yamada and G. Yoon, "Selecting the tuning parameter of the ℓ_1 trend filter," *Studies Nonlin. Dynam. Econometr.*, **20**, 97 (2016).

[477] S. Selvin, et al., "ℓ_1 Trend Filter for Image Denoising," *Procedia Comp. Sci.*, **93**, 495 (2016).

[478] J. Ottersten, B. Wahlberg, and C. R. Rojas, "Accurate Changing Point Detection for ℓ_1 Mean Filtering," *IEEE Sig. Process. Lett.*, **23**, 297 (2016).

Regularization

[479] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge Univ. Press, Cambridge, 2004. Available online from: `http://sites.google.com/site/ingridteles02/Book-ConvexOptimization.pdf`.

[480] A. N. Tikhonov and V. Y. Arsenin, *Solution of Ill-Posed Problems*, Winston, Washington DC, 1977.

[481] A. N. Tikhonov, et al., *Numerical Methods for the Solution of Ill-Posed Problems*, Springer, New York, 1995.

[482] A. E. Hoerl and R. W. Kennard, "Ridge Regression: Biased Estimation for Nonorthogonal Problems," *Technometrics*, **12**, 55 (1970).

[483] V. V. Ivanov, *Theory of Approximate Methods and Their Application to the Numerical Solution of Singular Integral Equations*, Nordhoff International, 1976.

[484] V. A. Morozov, *Methods for Solving Incorrectly Posed Problems*, Springer-Verlag, New York, 1984.

[485] N. Aronszajn, "Theory of Reproducing Kernels," *Trans. Amer. Math. Soc.*, **68**, 337 (1950).

[486] M. Foster, "An Application of the Wiener-Kolmogorov Smoothing Theory to Matrix Inversion," *J. SIAM*, **9**, 387 (1961).

[487] D. L. Phillips, "A Technique for the Numerical Solution of Certain Integral Equations of the First Kind," *J. ACM*, **9**, 84 (1962).

[488] M. A. Aizerman, E. M. Braverman, and L. I. Rozoner, "Theoretical Foundations of the Potential Function Method in Pattern Recognition Learning," *Autom. Remote Contr.*, **25**, 821 (1964).

[489] J. Callum, "Numerical Differentiation and Regularization," *SIAM J. Numer. Anal.*, **8**, 254 (1971).

[490] L. Eld'en, "An Algorithm for the Regularization of Ill-Conditioned, Banded Least Squares Problems," *SIAM J. Statist. Comput.*, **5**, 237 (1984).

[491] A. Neumaier, "Solving Ill-Conditioned and Singular Linear Systems: A Tutorial on Regularization," *SIAM Rev.*, **40**, 636 (1988).

[492] M. Bertero, C. De Mol, and E. R. Pikes, "Linear Inverse Problems with Discrete Data: I: General Formulation and Singular System Analysis," *Inv. Prob.*, **1**, 301 (1985).; and "II. Stability and Regularisation," *ibid.*, **4**, 573 (1988).

[493] M. Bertero, T. Poggio, and V. Torre, "Ill-Posed Problems in Early Vision," *Proc. IEEE*, **76**, 869 (1988).

[494] T. Poggio and F. Girosi, "Networks for Approximation and Learning," *Proc. IEEE*, **78**, 1481 (1990).

[495] A. M. Thompson, J. W. Kay, and D. M. Titterington, "Noise Estimation in Signal Restoration Using Regularization," *Biometrika*, **78**, 475 (1991).

[496] C. Cortes and V. Vapnik, “Support Vector Networks,” *Mach. Learn.*, **20**, 1 (1995).

[497] F. Girosi, M. Jones, and T. Poggio, “Regularization Theory and Neural Networks Architectures,” *Neural Comput.*, **7**, 219 (1995).

[498] A. J. Smola, B. Schölkopf, and K-R. Müller, “The Connection Between Regularization Operators and Support Vector Kernels,” *Neural Net.*, **11**, 637 (1998).

[499] V. N. Vapnik, *Statistical Learning Theory*, Wiley, New York, 1998.

[500] W. Fu, “Penalized Regressions: The Bridge versus the Lasso,” *J. Comput. Graph. Statist.*, **7**, 397 (1998). 1998.

[501] V. Cherkassky and F. Mulier, *Learning from Data: Concepts, Theory, and Methods*, Wiley, New York, 1998.

[502] F. Girosi, “An Equivalence Between Sparse Approximation and Support Vector Machines,” *Neural Comput.*, **10**, 1455 (1998).

[503] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*, Cambridge Univ. Press, Cambridge, 2000.

[504] T. Evgeniou, M. Pontil, and T. Poggio, “Regularization Networks and Support Vector Machines,” *Adv. Comput. Math.*, **13** 1 (2000).

[505] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Springer-Verlag, New York, 2001.

[506] F. Cucker and S. Smale, “On the Mathematical Foundations of Learning,” *Bull. AMS*, **39**, 1 (2001).

[507] L. Tenorio, “Statistical Regularization of Inverse Problems,” *SIAM Rev.*, **43**, 347 (2001).

[508] K-R. Müller, et al., “An Introduction to Kernel-Based Learning Algorithms,” *IEEE TRans. Neural Net.*, **12**, 181 (2001).

[509] B. Schölkopf, R. Herbrich, and A. J. Smola, “A Generalized Representer Theorem,” *Proc. 14th Ann. Conf. Comput. Learn. Th.*, p.416, (2001).

[510] T. Evgeniou, et al., “Regularization and Statistical Learning Theory for Data Analysis,” *Comput. Statist. Data Anal.*, **38**, 421 (2002).

[511] F. Cucker and S. Smale, “Best Choices for Regularization Parameters in Learning Theory: On the Bias-Variance Problem,” *Found. Comput. Math.*, **2**, 413 (2002).

[512] B. Schölkopf and A. Smola. *Learning with Kernels*, MIT Press, Cambridge, MA, 2002.

[513] J. A. K. Suykens, et al., *Least Squares Support Vector Machines*, World Scientific, Singapore, 2002.

[514] Z. Chen and S. Haykin, “On Different Facets of Regularization Theory,” *Neural Comput.*, **14**, 2791 (2002).

[515] T. Poggio and S. Smale, “The Mathematics of Learning: Dealing with Data,” *Notices AMS*, **50**, no.5, 537 (2003).

[516] M. Martínez-Ramón and C. Christodoulou, *Support Vector machines for Antenna Array Processing and Electromagnetics*, Morgan & Claypool, 2006.

[517] M. Martínez-Ramón, et al., “Kernel Antenna Array Processing,”, *IEEE Trans. Antennas Propagat.*, **55**, 642 (2007).

[518] M. Filippone, et al. “A Survey of Kernel and Spectral Methods for Clustering,” *Patt. Recogn.*, **41**, 176 (2008).

[519] W. Liu, P. P. Pokharel, and J. C. Principe, “The Kernel Least-Mean-Square Algorithm,” *IEEE Trans. Signal Process.*, **56**, 543 (2008).

L_1 Regularization and Sparsity

[520] O. J. Karst, “Linear Curve Fitting Using Least Deviations,” *J. Amer. Statist. Assoc.*, **53**, 118 (1958).

[521] E. J. Schlossmacher, “An Iterative Technique for Absolute Deviations Curve Fitting,” *J. Amer. Statist. Assoc.*, **68**, 857 (1973).

[522] V. A. Sposito, W. J. Kennedy and, J. E. Gentle, “Algorithm AS 110: L_p Norm Fit of a Straight Line,” *J. Roy. Statist. Soc., Series C*, **26**, 114 (1977).

[523] R. H. Byrd, D. A. Pyne, “Convergence of the iteratively reweighted least squares algorithm for robust regression,” Tech. Report, 313, Dept. Math. Sci., Johns Hopkins University, Baltimore, MD, 1979

[524] C. S. Burrus, 2012, “Iterative Reweighted least-squares,” *OpenStax-CNX web site*, `http://cnx. org/content/m45285/1.12`.

[525] S. C. Narula and J. F. Wellington, “The Minimum Sum of Absolute Errors Regression: A State of the Art Survey,” *Int. Statist. Review*, **50**, 317 (1982).

[526] R. Yarlagadda, J. B. Bednar, and T. L. Watt, “Fast algorithms for l_p deconvolution,” *IEEE Trans. Signal Process.*, **33**, 174 (1985). See also, J. A. Scales and S. Treitel, “On the connection between IRLS and Gauss’ method for l_1 inversion: Comments on ‘Fast algorithms for l_p deconvolution’,” *ibid.*, **35**, 581 (1987).

[527] J. A. Scales, A. Gersztenkorn, and S. Treitel, “Fast l_p solution of large, sparse, linear systems: Application to seismic travel time tomography,” *J. Comput. Phys.*, **75**, 314 (1988).

[528] G. Darche, “Iterative L^1 deconvolution,” Stanford Exploration Project, Annual Report 61, Jan. 1989; available from: `http://sepwww.stanford.edu/public/docs/sep61`.

[529] L. I. Rudin, S. Osher, and E. Fatemi, “Nonlinear total variation based noise removal algorithms,” *Physica D*, **60**, 259 (1992).

[530] K. Natarajan, “Sparse approximate solutions to linear systems,” *SIAM J. Comput.*, **24**, 227 (1995).

[531] R. Tibshirani, “Regression shrinkage and selection via the Lasso,” *J. Roy. Statist. Soc., Ser. B*, **58**, 267 (1996).

[532] F. Gorodnitsky and B. Rao, “Sparse signal reconstruction from limited data using FOCUSS: A reweighted norm minimization algorithm,” *IEEE Trans. Signal Process.*, **45**, 600 (1997).

[533] M. R. Osborne, B. Presnell, and B. A. Turlach, “On the LASSO and Its Dual,” *J. Comput. Graph. Stat.*, **9**, 319 (2000).

[534] S. S. Chen, D. L. Donoho, and M. A. Saunders, “Atomic decomposition by basis pursuit,” *SIAM Rev.*, **43**, 129 (2001).

[535] B. Efron, et al., “Least Angle Regression,” *Ann. Statist.*, **32**, 407 (2004).

[536] I. Daubechies, M. Defrise, and C. D. Mol, “An iterative thresholding algorithm for linear inverse problems with a sparsity constraint,” *Comm. Pure Appl. Math.,* **57** 1413 (2004).

[537] R. Tibshirani, et al., “Sparsity and smoothness via the fused Lasso,” *J. Roy. Statist. Soc., Ser. B*, **67**, 91 (2005).

[538] J-J. Fuchs, “Recovery of exact sparse representations in the presence of bounded noise.” *IEEE Trans. Inform. Th.*, **51**, 3601 (2005); and, “On Sparse Representations in Arbitrary Redundant Bases,” *ibid.*, **50**, 1341 (2004).

[539] J. A. Tropp, “Just Relax: Convex Programming Methods for Identifying Sparse Signals in Noise,” *IEEE Trans. Inform. Th.*, **52**, 1030 (2006).

[540] H. Zou and T. Hastie, “Regularization and variable selection via the elastic net,” *J. Roy. Statist. Soc., Ser. B*, **67**, 301 (2005).

[541] D. L. Donoho, “For most large underdetermined systems of linear equations the minimal ℓ_1-norm solution is also the sparsest solution,” *Comm. Pure Appl. Math.*, **59**, 797 (2006).

[542] E. J. Candès and T. Tao, “Decoding by linear programming,” *IEEE Trans, Inform. Th.*, **51**, 4203 (2005).

[543] E. J. Candès, J. K. Romberg, and T. Tao, “Stable signal recovery from incomplete and inaccurate measurements,” *Comm. Pure Appl. Math.*, **59** 1207 (2006).

[544] E. J. Candès, J. K. Romberg, “ℓ_1-MAGIC: Recovery of Sparse Signals via Convex Programming,” User’s Guide, 2006, available online from:
`https://statweb.stanford.edu/~candes/l1magic/downloads/l1magic.pdf`

[545] D. L. Donoho, “Compressed Sensing,” *IEEE Trans, Inform. Th.*, **52**, 1289 (2006).

[546] H. Zou, T. Hastie, and R. Tibshirani, “Sparse Principal Component Analysis,” *J. Comput. Graph. Stat.*, **15**, 265 (2006).

[547] M. Aharon, M. Elad, and A. Bruckstein, “K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation,” *IEEE Trans. Signal Process.*, **54**, 4311 (2006).

[548] S-J Kim, et al., “An Interior-Point Method for Large-Scale ℓ_1-Regularized Least Squares,” *IEEE J. Selected Topics Sig. Process.*, **1**, 606 (2007).

[549] A. d’Aspremont, et al., “A direct formulation for sparse PCA using semidefinite programming,” *SIAM Rev.*, **49**, 434 (2007).

[550] M. A. T. Figueiredo, R. D. Nowak, and S. J. Wright, "Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems," *IEEE J. Selected Topics Sig. Process.*, **1**, 586 (2007).

[551] M. Lobo, M. Fazel, and S. Boyd, "Portfolio optimization with linear and fixed transaction costs," *Ann. Oper. Res.*, **152**, 341 (2007).

[552] E. J. Candès and T. Tao, "The Dantzig Selector: Statistical Estimation When p Is Much Larger than n," *Ann. Statist.*, **35**, 2313 (2007); with Discussions, *ibid.*, p. 2352, 2358, 2365, 2370, 2373, 2385, 2392.

[553] E. J. Candès, M. Wakin, and S. Boyd, "Enhancing sparsity by reweighted ℓ_1 minimization," *J. Fourier Anal. Appl.*, **14**, 877 (2008).

[554] R. G. Baraniuk, et al., "A simple proof of the restricted isometry property for random matrices," *Constructive Approx.* **28**, 253 (2008).

[555] E. J. Candès, "The restricted isometry property and its implications for compressed sensing," *Comptes Rendus Mathematique*, **346**, 589 (2008).

[556] E. J. Candès and M. Wakin, "An introduction to compressive sampling," *IEEE Sig. Process. Mag.*, **25**(2), 21 (2008).

[557] A. M. Bruckstein, D. L. Donoho, and M. Elad, "From Sparse Solutions of Systems of Equations to Sparse Modeling of Signals and Images," *SIAM Rev.*, **51**, 34 (2009).

[558] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM J. Imaging Sci.*, **2**, 183 (2009).

[559] H. Mohimani, M. Babaie-Zadeh, and C. Jutten, "A fast approach for overcomplete sparse decomposition based on smoothed L_0 norm," *IEEE Trans. Signal Process.*, **57**, 289 (2009).

[560] R. E. Carrillo and K. E. Barner, "Iteratively re-weighted least squares for sparse signal reconstruction from noisy measurements," *43rd IEEE Conf. Inform. Sci. Syst.*, CISS 2009, p. 448.

[561] A. Cohen, W. Dahmen, and R. DeVore, "Compressed sensing and best k-term approximation," *J. Amer. Math. Soc.*, **22**, 211 (2009).

[562] E. J. Candès and Y. Plan, "Near-ideal model selection by ℓ_1 minimization," *Ann. Statist.*, **37**, 2145 (2009).

[563] M. J. Wainwright, "Sharp Thresholds for High-Dimensional and Noisy Sparsity Recovery Using ℓ_1-Constrained Quadratic Programming (Lasso)," *IEEE Trans. Inform. Th.*, **55**, 2183 (2009).

[564] I. Daubechies, M. Fornasier, and I. Loris, "Accelerated Projected Gradient Method for Linear Inverse Problems with Sparsity Constraints," *J. Fourier Anal. Appl.*, **14**, 764 (2008).

[565] I. Daubechies, et al., "Iteratively reweighted least squares minimization for sparse recovery," *Comm. Pure Appl. Math.*, **63**, 1 (2010).

[566] D. Wipf and S. Nagarajan, "Iterative reweighted ℓ_1 and ℓ_2 methods for finding sparse solutions," *IEEE J. Selected Topics Sig. Process.*, **4**, 317 (2010).

[567] E. Van Den Berg, et al., "Algorithm 890: Sparco: A testing framework for sparse reconstruction," *ACM Trans. Math. Softw.*, **35**, 29 (2009). Sparco web site: `http://www.cs.ubc.ca/labs/scl/sparco/`

[568] M. Elad, *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing*, Springer, 2010.

[569] P. Bühlmann and S. van de Geer, *Statistics for High-Dimensional Data*, Springer, 2011.

[570] J. Yang, and Y. Zhang, "Alternating direction algorithms for ℓ_1-problems in compressive sensing," *SIAM J. Sci. Comp.*, **33**, 250 (2011). YALL1 package: `http://yall1.blogs.rice.edu/`

[571] E. J. Candès, et al. "Robust Principal Component Analysis?," *J. Assoc. Comput. Mach.*, **58**, 11 (2011).

[572] D. Hardoon and J. Shawe-Taylor, "Sparse canonical correlation analysis," *Mach. Learn.* **83**, 331 (2011).

[573] Z. Ma, "Sparse principal component analysis and iterative thresholding," *Ann. Stat.*, **41**, 772 (2013).

[574] M. V. Afonso, J. M. Bioucas-Dias, and M. A. T. Figueiredo, "Fast Image Recovery Using Variable Splitting and Constrained Optimization," *IEEE Trans. Image Process.*, **19**, 2345 (2010); and "An Augmented Lagrangian Approach to the Constrained Optimization Formulation of Imaging Inverse Problems," *ibid.*, **20**, 68 (2011). SALSA software available from: `http://cascais.lx.it.pt/~mafonso/salsa.html`

[575] S. Boyd, et al., "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine Learning*, **3**(1), 3(1), 1 (2011); see also, `http://stanford.edu/~boyd/admm.html`.

[576] N. Parikh and S. Boyd, "Proximal Algorithms," *Foundations and Trends in Optimization*, **1**, 123 (2013).

[577] F. Bach, et al., "Optimization with Sparsity-Inducing Penalties," *Foundations and Trends in Machine Learning*, **4**(1), 1 (2012).

[578] Y-B Zhao and D. Li, "Reweighted ℓ_1-minimization for sparse solutions to underdetermined linear systems," *SIAM J. Optim.*, **22**, 1065 (2012).

[579] J. Mairal and B. Yu, "Complexity analysis of the lasso regularization path," *arXiv*, arXiv preprint: 1205.0079 (2012).

[580] I. Selesnick, 2012, "Introduction to Sparsity in Signal Processing," *OpenStax-CNX web site*, `https://cnx.org/content/m43545/latest`, including MATLAB examples using SALSA [574].

[581] S. Foucart and H. Rauhut, *A Mathematical Introduction to Compressive Sensing*, Birkhäuser, 2013.

[582] J. P. Brooks, J. H. Dulá, and E. L. Boone, "A Pure L_1-norm Principal Component Analysis," *Comput. Stat. Data Anal.*, **61**, 83 (2013).

[583] R. C. Aster, B. Borchers, and C. H. Thurber, *Parameter Estimation and Inverse Problems*, 2/e, Academic Press, 2013.

[584] R. J. Tibshirani, "The lasso problem and uniqueness," *Electr. J. Statist.*, **7**, 1456 (2013).

[585] D. Ba, et al., "Convergence and Stability of Iteratively Re-weighted Least Squares Algorithms," *IEEE Trans. Signal Process.*, **62**, 183 (2014).

[586] I. Rish and G. Grabarnik, *Sparse Modeling: Theory, Algorithms, and Applications,* Chapman and Hall/CRC, 2014.

[587] T. Hastie, R. Tibshirani, and M. Wainwright, *Statistical Learning with Sparsity: The Lasso and Generalizations*, CRC Press, 2015.

[588] C. F.Mecklenbräuker, P. Gerstoft, and E. Zöchmann, "c-LASSO and its dual for sparse signal estimation from array data," *Sig. Process.*, **130**, 204 (2017).

[589] `http://dsp.rice.edu/cs`, Compressive Sensing Resources.

[590] MATLAB packages for solving the L_1 regularization and related problems:

Mathworks	`https://www.mathworks.com/help/stats/lasso-and-elastic-net.html`
	`https://www.mathworks.com/help/stats/lasso.html`
ADMM	`http://stanford.edu/~boyd/admm.html`
CVX	`http://cvxr.com/cvx/`
FISTA	`http://ie.technion.ac.il/~becka/papers/rstls_package.zip`
Homotopy	`http://www.ece.ucr.edu/~sasif/homotopy/`
L1-MAGIC	`https://statweb.stanford.edu/~candes/l1magic/`
LARS	`https://publish.illinois.edu/xiaohuichen/code/lars/`
	`https://sourceforge.net/projects/sparsemodels/files/LARS/`
NESTA	`https://statweb.stanford.edu/~candes/nesta/`
REGTOOLS	`http://www.imm.dtu.dk/~pcha/Regutools/`
SALSA	`http://cascais.lx.it.pt/~mafonso/salsa.html`
SOL	`http://web.stanford.edu/group/SOL/software.html`
Sparco	`http://www.cs.ubc.ca/labs/scl/sparco/`
SpaRSA	`http://www.lx.it.pt/~mtf/SpaRSA/`
Sparselab	`https://sparselab.stanford.edu/`
SPGL1	`http://www.cs.ubc.ca/labs/scl/spgl1/`
TwIST	`http://www.lx.it.pt/~bioucas/TwIST/TwIST.htm`
YALL1	`http://yall1.blogs.rice.edu/`

Comb Filters and Signal Averaging

[591] S. F. George and A. S. Zamanakos, "Comb Filters for Pulsed Radar Use," *Proc. IRE*, **42**, 1159 (1954).

[592] G. Arndt, F. Stuber, and R. Panneton, "Video-Signal Improvement Using Comb Filtering Techniques," *IEEE TRans. Commun.*, **21**, 331 (1973).

[593] S-C Pei and C-C Tseng, “A Comb Filter design Using Fractional-Sample Delay,” *IEEE Trans. Circ. Syst.–II: Anal. Dig. Sig. Process.*, **45**, 649 (1998).

[594] A. G. Dempster, “Use of Comb Filters in GPS L1 Receivers,” *GPS Solut.*, **12**, 179 (2008).

[595] S. J. Orfanidis, “High-Order Digital Parametric Equalizer Design,” *J. Audio Eng. Soc.*, **53**, 1026 (2005). The MATLAB toolbox is available from `http://www.ece.rutgers.edu/~orfanidi/hpeq/`, or, `http://www.aes.org/journal/suppmat/`

[596] D. G. Childers, “Biomedical Signal Processing,” in *Selected Topics in Signal Processing*, S. Haykin, ed., Prentice Hall, Upper Saddle River, NJ, 1989.

[597] A. Cohen, *Biomedical Signal Processing*, vols. 1 and 2, CRC Press, Boca Raton, FL, 1986.

[598] H. G. Goovaerts and O. Rompelman, “Coherent Average Technique: A Tutorial Review,” *J. Biomed. Eng.*, **13**, 275 (1991).

[599] P. Horowitz and W. Hill, *The Art of Electronics*, 2nd ed., Cambridge University Press, Cambridge, 1989.

[600] O. Rompelman and H. H. Ros, “Coherent Averaging Technique: A Tutorial Review, Part 1: Noise Reduction and the Equivalent Filter,” *J. Biomed. Eng.*, **8**, 24 (1986); and “Part 2: Trigger Jitter, Overlapping Responses, and Non-Periodic Stimulation,” ibid., p. 30.

[601] V. Shvartsman, G. Barnes, L. Shvartsman, and N. Flowers, “Multichannel Signal Processing Based on Logic Averaging,” *IEEE Trans. Biomed. Eng.*, **BME-29**, 531 (1982).

[602] C. W. Thomas, M. S. Rzeszotarski, and B. S. Isenstein, “Signal Averaging by Parallel Digital Filters,” *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-30**, 338 (1982).

[603] T. H. Wilmshurst, *Signal Recovery from Noise in Electronic Instrumentation*, 2nd ed., Adam Hilger and IOP Publishing, Bristol, England, 1990.

[604] J. F. Kaiser and R. W. Schafer, “On the Use of the I_0-Sinh Window for Spectrum Analysis,” *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-28**, 105 (1980).

X-11 Seasonal Adjustment Method

[605] J. Shiskin, A. Young, and J. Musgrave, “The X-11 Variant of the Census Method II Seasonal Adjustment Program,” US Census Bureau, Technical Paper 15, (1967), available from [609].

[606] E. B, Dagum, “The X-11-ARIMA Seasonal Adjustment Method,” Statistics, Canada, (1980), available from [609].

[607] `http://www.census.gov/srd/www/x12a/`, US Census Bureau X-12-ARIMA Seasonal Adjustment Program.

[608] `http://www.census.gov/srd/www/sapaper/sapaper.html`, US Census Bureau Seasonal Adjustment Papers.

[609] `http://www.census.gov/srd/www/sapaper/historicpapers.html`, Historical Papers on X-11 and Seasonal Adjustment.

[610] K. F. Wallis, “Seasonal Adjustment and Relations Between Variables,” *J. Amer. Statist. Assoc.*, **69**, 18 (1974).

[611] K. F. Wallis, “Seasonal Adjustment and Revision of Current Data: Linear Filters for the X-11 Method,” *J. Roy. Statist. Soc., Ser. A*, **145**, 74 (1982).

[612] W. R. Bell and S. C. Hillmer, “Issues Involved with Seasonal Adjustment of Economic Time Series,” *J. Bus. Econ. Statist.*, **2**, 291 (1984). Available on line from `http://www.census.gov/srd/papers/pdf/rr84-09.pdf`.

[613] W. R. Bell and B. C. Monsell, “X-11 Symmetric Linear Filters and their Transfer Functions,” US Census Bureau, SRD Research Report, No. RR-92/15, (1992). Available online from the web site [608].

[614] E. B. Dagum, N. Chhab, and K. Chiu, “Derivation and Properties of the X11ARIMA and Census X11 Linear Filters,” *J. Official Statist.*, **12**, 329 (1996).

[615] J. C. Musgrave, “A Set of Weights to End all End Weights,” Working paper, US Dept. Commerce, (1964), available online from [609].

[616] M. Doherty, “The Surrogate Henderson Filters in X-11”, *Aust. N. Z. J. Stat.*, **43**, 385 (2001), originally circulated in 1996.

[617] D. F. Findley, et al., “New Capabilities and Methods of the X-12-ARIMA Seasonal-Adjustment Program,” *J. Bus. Econ. Statist.*, **16**, 127 (1998), with Comments, p.153.

[618] D. Ladiray and B. Quenneville, *Seasonal Adjustment with the X-11 Method*, Lecture Notes in Statistics No. 158, Springer-Verlag, New York, 2001. Available online from the web site [608] (in French and Spanish.)

[619] A. G. Gray and P. J. Thomson, “On a Family of Finite Moving-Average Trend Filters for the Ends of Series,” *J. Forecasting*, **21**, 125 (2002).

[620] B. Quenneville, D. Ladiray, and B. Lefrançois, “A Note on Musgrave Asymmetrical Trend-Cycle Filters,” *Int. J. Forecast.*, **19**, 727 (2003).

[621] D. F. Findley and D. E. K. Martin, “Frequency Domain Analysis of SEATS and X-11/X-12-ARIMA Seasonal Adjustment Filters for Short and Moderate-Length Time Series,” *J. Off, Statist.*, **22**, 1 (2006).

[622] C. E. V. Leser, “Estimation of Quasi-Linear Trend and Seasonal Variation,” *J. Amer. Statist. Assoc.*, **58**, 1033 (1963).

[623] H. Akaike, “Seasonal Adjustment by a Bayesian Modeling,” *J. Time Ser. Anal.*, **1**, 1 (1980).

[624] E. Schlicht, “A Seasonal Adjustment Principle and a Seasonal Adjustment Method Derived from this Principle,” *J. Amer. Statist. Assoc.*, **76**, 374 (1981).

[625] F. Eicker, “Trend-Seasonal Decomposition of Time Series as Whittaker-Henderson Graduation,” *Statistics*, **19**, 313 (1988).

Model-Based Seasonal Adjustment

[626] E. J. Hannan, “The Estimation of Seasonal Variation in Economic Time Series,” *J. Amer. Statist. Assoc.*, **58**, 31 (1963).

[627] E. J. Hannan, “The Estimation of Changing Seasonal Pattern,” *J. Amer. Statist. Assoc.*, **59**, 1063 (1964).

[628] M. Nerlove, “Spectral Analysis of Seasonal Adjustment Procedures,” *Econometrica*, **32**, 241 (1964).

[629] J. P. Burman, “Moving Seasonal Adjustment of Economic Time Series,” *J. Roy. Statist. Soc., Ser. A*, **128**, 534 (1965).

[630] D. M. Grether and M. Nerlove, “Some Properties of “Optimal” Seasonal Adjustment,” *Econometrica*, **38**, 682 (1970).

[631] G. E. P. Box, S. Hillmer, and G. C. Tiao, “Analysis and Modeling of Seasonal Time Series,” (1978), available online from [609].

[632] J. P. Burman, “Seasonal Adjustment by Signal Extraction,” *J. Roy. Statist. Soc., Ser. A*, **143**, 321 (1980).

[633] S. C. Hillmer and G. C. Tiao, “An ARIMA-Model-Based Approach to Seasonal Adjustment,” *J. Amer. Statist. Assoc.*, **77**, 63 (1982).

[634] W. S. Cleveland, A. E, Freeny, and T. E. Graedel, “The Seasonal Component of Atmospheric CO_2: Information from New Approaches to the Decomposition of Seasonal Time Series,” *J. Geoph. Res.*, **88**, 10934 (1983).

[635] P. Burridge and K. F. Wallis, “Unobserved-Components Models for Seasonal Ajustment Filters,” *J. Bus. Econ. Statist.*, **2**, 350 (1984).

[636] G. Kitagawa and W. Gersch, “A Smoothness Priors-State Space Modeling of Time Series with Trend and Seasonality,” *J. Amer. Statist. Assoc.*, **79**, 378 (1984).

[637] R. B. Cleveland, et al., “STL: A Seasonal-Trend Decomposition Procedure Based on Loess,” *J. Official Statist.*, **6**, 3 (1990).

[638] G. Kitagawa and W. Gersch, *Smoothness Priors Analysis of Time Series*, Springer, New York, 1996.

[639] V. Gómez and A. Maravall, “Programs TRAMO and SEATS. Instructions for the User (with some updates),” Working Paper 9628, (Servicio de Estudios, Banco de España, 1996).

[640] C. Planas, “The Analysis of Seasonality In Economic Statistics: A Survey of Recent Developments,” *Qüestió*, **22**, 157 (1998).

[641] V. Gómez and A. Maravall, “Seasonal Adjustment and Signal Extraction in Economic Time Series,” chapter 8, in *A Course in Time Series Analysis*, D. Peña, G. C. Tiao, and R. S. Tsay, eds., Wiley, New York, 2001. Available online from `http://bde.es/servicio/software/tramo/sasex.pdf`.

[642] J. A.D. Aston, et al., “New ARIMA Models for Seasonal Time Series and Their Application to Seasonal Adjustment and Forecasting,” US Census Bureau, (2007), available online from [608].

Unobserved Components Models

[643] E. J. Hannan, “Measurement of a Wandering Signal Amid Noise,” *J. Appl. Prob.*, **4**, 90 (1967).

[644] E. L. Sobel, “Prediction of a Noise-Distorted, Multivariate, Non-Stationary Signal,” *J. Appl. Prob.*, **4**, 330 (1967).

[645] W. P. Cleveland and G. C. Tiao, “Decomposition of Seasonal Time Series: A Model for the Census X-11 Program,” *J. Amer. Statist. Assoc.*, **71**, 581 (1976).

[646] D. A. Pierce, “Signal Extraction Error in Nonstationary Time Series,” *Ann. Statist.*, **7**, 1303 (1979).

[647] W. Bell, “Signal Extraction for Nonstationary Time Series,” *Ann. Statist.*, **12**, 646 (1984), with correction, *ibid.*, **19**, 2280 (1991).

[648] A. Maravall, “A Note on Minimum Mean Squared Error Estimation of Signals with Unit Roots,” *J. Econ. Dynam. & Contr.*, **12**, 589 (1988).

[649] W. R. Bell and E. K. Martin, “Computation of Asymmetric Signal Extraction Filters and Mean Squared Error for ARIMA Component Models,” *J. Time Ser. Anal.*, **25**, 603 (2004). Available online from [608].

[650] S. Beveridge and C. Nelson, “A New Approach to Decomposition of Economic Time Series into Permanent and Transitory Components with Particular Attention to Measurement of the Business Cycle,” *J. Monet. Econ.*, **7**, 151 (1981).

[651] V. Gómez and A. Maravall, “Estimation, Prediction, and Interpolation for Nonstationary Series with the Kalman Filter,” *J. Amer. Statist. Assoc.* **89**, 611 (1994).

[652] P. Young, “Data-Based Mechanistic Modelling of Environmental, Ecological, Economic, and Engineering Systems,” *Environ. Model. & Soft.*, **13**, 105 (1998).

[653] V. Gómez, “Three Equivalent Methods for Filtering Finite Nonstationary Time Series,” *J. Bus. Econ. Stat.*, **17**, 109 (1999).

[654] A. C. Harvey and S. J. Koopman, “Signal Extraction and the Formulation of Unobserved Components Modelsm” *Econometr. J.*, **3**, 84 (2000).

[655] R. Kaiser and A. Maravall, *Measuring Business Cycles in Economic Time Series*, Lecture Notes in Statistics, **154**, Springer-Verlag, New York, 2001. Available online from `http://www.bde.es/servicio/software/tramo/mhpfilter.pdf`.

[656] E. Ghysels and D. R. Osborn, *The Econometric Analysis of Seasonal Time Series*, Cambridge Univ. Press, Cambridge, 2001.

[657] D. S. G. Pollock, “Filters for Short Non-Stationary Sequences,” *J. Forecast.*, **20**, 341 (2001).

[658] R. Kaiser and A. Maravall, “Combining Filter Design with Model-Based Filtering (with an Application to Business Cycle Estimation),” *Int. J. Forecast.*, **21** 691 (2005).

[659] A. Harvey and G. De Rossi, “Signal Extraction,” in *Palgrave Handbook of Econometrics*, vol 1, K. Patterson and T. C. Mills, eds., 2006, Palgrave MacMillan, New York, 2006.

[660] A. Harvey, “Forecasting with Unobserved Components Time Series Models,” *Handbook of Economic Forecasting*, G. Elliot, C. Granger, and A. Timmermann, eds., North Holland, 2006.

[661] D. S. G. Pollock, “Econometric Methods of Signal Extraction,” *Comput. Statist. Data Anal.*, **50**, 2268 (2006).

[662] M. Bujosa, A. Garcia-Ferrer, and P. C. Young, “Linear Dynamic Harmonic Regression,” *Comput. Statist. Data Anal.*, **52**, 999 (2007).

[663] T. McElroy, “Matrix Formulas for Nonstationary ARIMA Signal Extraction,” *Econometr. Th.*, **24**, 988 (2008).

[664] M. Wildi, *Real-Time Signal Extraction*, Springer, New York, 2008. Available online from `http://www.idp.zhaw.ch/fileadmin/user_upload/engineering/_Institute_und_Zentren/IDP/sonderthemen/sef/signalextraction/papers/IDP-WP-08Sep-01.pdf`.

Wavelets and Applications

[665] I. Daubechies, *Ten Lectures on Wavelets*, SIAM, Philadelphia, PA, 1992.

[666] J. M. Combes, A. Grossmann, and P. Tchamitchian, eds., *Wavelets, Time-Frequency Methods and Phase Space*, Springer-Verlag, Berlin, 1989.

[667] C. K. Chui, *An Introduction to Wavelets*, Academic Press, New York, 1992.

[668] Y. Meyer, *Wavelets, Algorithms and Applications*, SIAM, Philadelphia, 1993.

[669] A. Akansu and R. Haddad, *Multiresolution Signal Decomposition*, Academic Press, New York, 1993.

[670] P. P. Vaidyanathan, *Multirate Systems and Filter Banks*, Prentice Hall, Englewood Cliffs, NJ, 1993.

[671] G. Kaiser, *A Friendly Guide to Wavelets* Birkhäuser, Boston, 1994.

[672] V. Wickerhauser, *Adapted Wavelet Analysis from Theory to Software*, AK Peters, Boston, 1994.

[673] M. Vetterli and J. Kovačević, *Wavelets and Subband Coding*, Prentice Hall, Englewood Cliffs, NJ, 1995.

[674] G. Strang and T. Nguyen, *Wavelets and Filter Banks*, Wellesley-Cambridge Press, Wellesley, MA, 1996.

[675] C. S. Burrus, R. A. Gopinath, and H. Guo, *Introduction to Wavelets and Wavelet Transforms: A Primer*, Prentice Hall, Upper Saddle River, NJ, 1998.

[676] S. Mallat, *A Wavelet Tour of Signal Processing*, Academic, New York, 1998.

[677] A. Antoniadis and G. Oppenheim, eds., *Wavelets and Statistics*, Lecture Notes in Statistics v. 103, Springer-Verlag, New York, 1995.

[678] B. Vidakovic, *Statistical Modeling with Wavelets*, Wiley, New York, 1999.

[679] R. Gençay, F. Selçuk, and B. Whitcher, *An Introduction to Wavelets and Other Filtering Methods in Finance and Economics*, Academic, New York, 2001.

[680] A. Jensen and A. la Cour-Harbo, *Ripples in Mathematics*, Springer, New York, 2001.

[681] S. Jaffard, Y. Meyer, and R. D. Ryan, *Wavelets: Tools for Science and Technology*, SIAM, Philadelphia, 2001.

[682] A. Cohen, *Numerical Analysis of Wavelet Methods*, Elsevier, Amsterdam, 2003.

[683] C. Heil, D. F. Walnut, and I. Daubechies, *Fundamental Papers in Wavelet Theory*, Princeton Univ. Press, Princeton, NJ, 2006.

[684] D. B. Percival and A. T. Walden, *Wavelet Methods for Time Series Analysis* Cambridge University Press, Cambridge, 2006.

[685] P. Van Fleet, *Discrete Wavelet Transformations*, Wiley, New York, 2008.

[686] G. P. Nason, *Wavelet Methods in Statistics with R*, Springer, New York, 2008.

[687] G. Strang, “Wavelets and Dilation Equations: A Brief Introduction,” *SIAM J. Math. Anal.*, **31**, 614 (1989).

[688] C. Heil and D. Walnut, “Continuous and Discrete Wavelet Transforms,” *SIAM Rev.*, **31**, 628 (1989).

[689] L. Cohen, “Time-Frequency Distributions: A Review,” *Proc. IEEE,* **77**, 941 (1989).

[690] O. Rioul and M. Vetterli, “Wavelets and Signal Processing,” *IEEE SP Mag.*, **8**, no.4, 14, October 1991.

[691] Special issue on Wavelets, *IEEE Trans. Inform. Th.*, **38**, Mar. 1992.

[692] *IEEE Trans. Signal Process.*, Special Issue on Wavelets and Signal Processing, **41**, Dec. 1993.

[693] A. H. Tewfik, M. Kim, and M. Deriche, “Multiscale Signal Processing Techniques: A Review,” in N. K. Bose and C. R. Rao, eds., *Handbook of Statistics*, vol. 10, Elsevier, Amsterdam, 1993.

[694] Special Issue on Wavelets, *Proc. IEEE*, **84**, Apr. 1996.

[695] G. Strang, “Wavelet Transforms versus Fourier Transforms,” *Bull. (New Series) Am. Math. Soc.*, **28**, 288 (1993).

[696] B. Jawerth and T. Swelden, “An Overview of Wavelet Based Multiresolution Analyses,” *SIAM Rev.*, **36**, 377 (1994).

[697] G. Strang, “Wavelets,” *Amer. Scientist*, **82**, 250, May-June 1994.

[698] P. M. Bentley and J. T. E. McDonnell, “Wavelet Transforms: An Introduction,” *Electr. Comm. Eng. J.*, p. 175, Aug. 1994.

[699] A. Graps, “An Introduction to Wavelets,” *IEEE Comput. Sci. Eng. Mag.*, **2**, no. 2, 50, Summer 1995.

[700] J. R. Williams and K. Amaratunga, “Introduction to Wavelets in Engineering,” *Int. J. Numer. Meth. Eng.*, **37**, 2365 (1994).

[701] I. Daubechies, “Where Do Wavelets Come From? A Personal Point of View,” *Proc. IEEE*, **84**, 510 (1996).

[702] W. Sweldens, “Wavelets: What next?,” *Proc. IEEE*, **84**, 680 (1996).

[703] C. Mulcahy, “Plotting and Scheming with Wavelets,” *Math. Mag.*, **69**, 323 (1996).

[704] C. Mulcahy, “Image Compression Using The Haar Wavelet Transform,” *Spelman College Sci. Math. J.*, **1**, 22 (1997).

[705] M . Vetterli, “Wavelets, Approximation, and Compression,” *IEEE SP Mag.*, Sept. 2001, p. 59.

[706] P. P. Vaidyanathan, “Quadrature Mirror Filter Banks, M-band Extensions and Perfect Reconstruction Techniques,” *IEEE ASSP Mag.*, **4**, no. 3, 4, July 1987.

[707] P. P. Vaidyanathan and Z. Doganata, ”The Role of Lossless Systems in Modern Digital Signal Processing: A Tutorial,” *IEEE Trans. Educ.*, **32**, 181 (1989).

[708] P. P. Vaidyanathan, “Multirate Digital Filters, Filter Banks, Polyphase Networks, and Applications: A Tutorial,” *Proc. IEEE*, **78**, 56 (1990).

[709] A. Haar, “Zur Theorie der Orthogonalen Funktionensysteme,” *Math. Annal.*, **69**, 331 (1910). Reprinted in [683].

[710] D. Gabor, “Theory of Communication,” *J. IEE*, **93**, 429 (1946).

[711] D. Esteban and C. Galand, “Application of Quadrature Mirror Filters to Split-Band Voice Coding Schemes,” *Proc. IEEE Int. Conf. Acoust. Speech, Signal Process.,* May 1977, p. 191. Reprinted in [683].

[712] P. J. Burt and E. H. Adelson, “The Laplacian Pyramid as a Compact Image Code,” *IEEE Trans. Commun.*, **31**, 532 (1983). Reprinted in [683].

[713] M. J. T. Smith and T. P. Barnwell III, “A Procedure for Designing Exact Reconstruction Filter Banks for Tree-Structured Sub-Band Coders,” *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Process.*, San Diego, CA, March 1984. Reprinted in [683].

[714] F. Mintzer, “Filters for Distortion-Free Two-Band Multirate Filter Banks,” *IEEE Trans. Acoust., Speech, Signal Process.*, **33**, 626 (1985). Reprinted in [683].

[715] A. Grossmann and J. Morlet, “Decomposition of Hardy Functions into Square Integrable Wavelets of Constant Shape,” *SIAM J. Math. Anal.*, **15**, 723 (1984). Reprinted in [683].

[716] A. Grossmann, J. Morlet, and T. Paul, “Transforms Associated to Square Integrable Group Representations I,” *J. Math. Phys.*, **26**, 2473 (1985). Reprinted in [683].

[717] I. Daubechies, “Orthonormal Bases of Compactly Supported Wavelets,” *Commun. Pure Appl. Math.*, **41** 909 (1988). Reprinted in [683].

[718] G. Battle, “A block spin construction of ondelettes. Part I: Lemarié functions” *Commun. Math. Phys.*, **110**, 601 (1987); and, “Part II: the QFT connection,” *ibid.*, **114**, 93 (1988). Reprinted in [683].

[719] P. G. Lemarié, “Ondelettes á localisation exponentielle,” *J. Math. Pures Appl.*, **67**, 227 (1988).

[720] Y. Meyer, “Wavelets with Compact Support,” Zygmund Lectures, U. Chicago (1987). Reprinted in [683].

[721] S. Mallat, “A Theory for Multiresolution Signal Decomposition: the Wavelet Representation,” *IEEE Trans. Patt. Recogn. Mach. Intell.*, **11**, 674 (1989). Reprinted in [683].

[722] S. Mallat, “Multiresolution Approximations and Wavelet Orthonormal Bases of $L^2(\mathbf{R})$,” *Trans. Amer. Math. Soc.*, **315**, 69 (1989). Reprinted in [683].

[723] A. Cohen, “Ondelettes, Analysis Multirésolutions et Filtres Mirroirs en Quadrature,” *Ann. Inst. H. Poincaré, Anal. Non Linéaire*, **7**, 439 (1990). Reprinted in [683].

[724] A. Grossmann, R. Kronland-Martinet, and J. Morlet, “Reading and Understanding Continuous Wavelet Transforms,” in [666].

[725] M. Holschneider, et al, “A Real Time Algorithm for Signal Analysis with the Help of the Wavelet Transform,” in [666].

[726] I. Daubechies, “The Wavelet Transform, Time-Frequency Localization and Signal Analysis,” *IEEE Trans. Inform. Th.*, **36**, 961 (1990). Reprinted in [683].

[727] M. Holsclmeider, “Wavelet Analysis on the Circle,” *J. Math. Phys.*, **31**, 39 (1990).

[728] G. Beylkin, R. Coifman, and V. Rokhlin, “Fast Wavelet Transforms and Numerical Algorithms I,,, *Commun. Pure Appl. Math.*, **44**, 141 (1991). Reprinted in [683].

[729] W. Lawton, “Tight Frames of Compactly Supported Affine Wavelets,, *J. Math. Phys.*, **31**, 1898 (1990). Reprinted in [683].

[730] W. Lawton, “Necessary and Sufficient Conditions for Constructing Orthonormal Wavelet Bases,” *J. Math. Phys.*, **32**, 57 (1991).

[731] W. Lawton, “Multiresolution Properties of the Wavelet Galerkin Operator,” *J. Math. Phys.*, **32**, 1440 (1991).

[732] I. Daubechies and J. Lagarias, “Two-Scale Difference Equations I. Existence and Global Regularity of Solutions,” *SIAM J. Math. Anal.*, **22**, 1388 (1991); and, “II. Local Regularity, Infinite Products of Matrices and Fractals,” *ibid.*, **24**, 1031 (1992).

[733] A. Cohen, I. Daubechies, and J.-C. Feauveau, “Biorthogonal Bases of Compactly Supported Wavelets,” *Commun. Pure Appl. Math.*, **45**, 485 (1992).

[734] O. Rioul and P. Duhamel, “Fast Algorithms for Discrete and Continuous Wavelet Transforms,” *IEEE Trans. Inform. Th.*, **38**, 569 (1992).

[735] M. Vetterli and C. Herley, “Wavelets and Filter Banks: Theory and Design,” *IEEE Trans. Signal Process.*, **40**, 2207 (1992).

[736] G. G. Walter, “A Sampling Theorem for Wavelet Subspaces,” *IEEE Trans. Inform. Th.*, **38**, 881 (1992).

[737] N. H. Getz, “A Perfectly Invertible, Fast, and Complete Wavelet Transform for Finite Length Sequences: The Discrete Periodic Wavelet Transform,” *SPIE Mathematical Imaging*, vol, 2034, p. 332, (1993).

[738] L. Cohen, “The Scale Representation,” *IEEE Trans. Signal Process.*, **41**, 3275 (1993).

[739] I. Daubechies, “Orthonormal Bases of Compactly Supported Wavelets II, Variations on a Theme,” *SIAM J. Math. Anal.*, **24**, 499 (1993).

[740] O. Rioul, “A Discrete-Time Multiresolution Theory,” *IEEE Trans. Signal Process.*, **41**, 2591 (1993).

[741] X. Xia and Z. Zhang, “On Sampling Theorem, Wavelets, and Wavelet Transforms, *IEEE Trans. Signal Process.*, **41**, 3524 (1993).

[742] W. Sweldens, “The Lifting Scheme: A Custom-Design Construction of Biorthogonal Wavelets,” *Appl. Comput. Harmon. Anal.*,**3**, 186 (1996).

[743] W. Sweldens, “The Lifting Scheme: A Construction of Second Generation Wavelets,” *SIAM J. Math. Anal.*, **29**, 511 (1996).

[744] G. Strang, “Eigenvalues of $(\downarrow 2)H$ and Convergence of the Cascade Algorithm,” *IEEE Trans. Signal Process.*, **44**, 233 (1996).

[745] S. H. Maes, “Fast Quasi-Continuous Wavelet Algorithms for Analysis and Synthesis of One-Dimensional Signals,” *SIAM J. Appl. Math.*, **57**, 1763 (1997).

[746] I. Daubechies and W. Sweldens, “Factoring Wavelet Transforms into Lifting Steps,” *J. Fourier Anal. Appl.*, **4**, 247 (1998).

[747] M. Unser and T. Blu, “Wavelet Theory Demystified,” *IEEE Trans. Signal Process.*, **51**, 470 (2003).

[748] P. Dutilleux, “An Implementation of the Algorithme à Trous to Compute the Wavelet Transform,” in [666].

[749] S. Mallat, “Zero-Crossings of a Wavelet Transform,” *IEEE Trans. Inform. Th.*, **37**, 1019 (1991).

[750] G. Beylkin, “On the Representation of Operators in Bases of Compactly Supported Wavelets,” *SIAM J. Numer. Anal.*, **29**, 1716 (1992).

[751] M. J. Shensa, “The Discrete Wavelet Transform: Wedding the á Trous and Mallat Algorithms,” *IEEE Trans. Signal Process.*, **40**, 2464 (1992).

[752] G. P. Nason and B. W. Silverman, “The Discrete Wavelet Transform in S,” *J. Comput. Graph. Statist.*, **3**, 163 (1994).

[753] G. P. Nason and B. W. Silverman, “The Stationary Wavelet Transform and Some Statistical Applications,” in [677].

[754] R. R. Coifman and D. L. Donoho, “Translation-Invariant Denoising,” in [677].

[755] J. C. Pesquet, H. Krim, and H. Carfantan, “Time-Invariant Orthonormal Wavelet Representations,” *IEEE Trans. Signal Process.*, **44**, 1964 (1996).

[756] J. Liang and T. W. Parks, “A Translation-Invariant Wavelet Representation Algorithm with Applications,” *IEEE Trans. Signal Process.*, **44**, 225 (1996).

[757] M. Lang, et al., “Noise Reduction Using An Undecimated Discrete Wavelet Transform,” *IEEE Signal Process. Lett.*, **3**, 10 (1996).

[758] H. Sari-Sarraf and D. Brzakovic, “A Shift-Invariant Discrete Wavelet Transform,” *IEEE Trans. Signal Process.*, **45**, 2621 (1997).

[759] J. E. Fowler, “The Redundant Discrete Wavelet Transform and Additive Noise, *IEEE Signal Process. Lett.*, **12**, 629 (2005).

[760] A. F. Abdelnour and I. W. Selesnick, “Symmetric Nearly Shift-Invariant Tight Frame Wavelets,” *IEEE Trans. Signal Process.*, **53**, 231 (2005).

[761] J.-L. Starck, J. Fadili, and F. Murtagh, “The Undecimated Wavelet Decomposition and its Reconstruction,” *IEEE Trans. Imag. Process.*, **16**, 297 (2007).

[762] J. D. Johnston, “Transform Coding of Audio Signals Using Perceptual Noise Criteria,” *IEEE J. Selected Areas Commun.*, **6**, 314 (1988).

[763] D. J. LeGall, H. Gaggioni, and C. T. Chen, “Transmission of HDTV Signals Under 140 Mbits/s Using a Subband Decomposition and Discrete Cosine Transform Coding,” in L. Chiariglione, ed., *Signal Processing of HDTV*, Elsevier, Amsterdam, 1988.

[764] JPEG Technical Specification: Revision (DRAFT), Joint Photographic Experts Group, ISO/IEC JTC1/SC2/WG8, CCITT SGVIII, August 1990.

[765] G. K. Wallace, “The JPEG Still Picture Compression Standard,” *Commun. ACM*, **34**, 30 (1991).

[766] D. LeGall, “MPEG: A Video Compression Standard for Multimedia Applications,” *Commun. ACM*, **34** 46 (1991).

[767] N. S. Jayant, “Signal Compression: Technology Targets and Research Directions,” *IEEE J. Sel. Areas Commun.*, **10**, 796 (1992).

[768] M. Antonini, et al., “Image Coding Using Wavelet Transform,” *IEEE Trans. Im. Process.*, **1**, 205 (1992).

[769] R. DeVore, B. Jawerth, and V. Popov, “Compression of Wavelet Decompositions,” *Amer. J. Math.*, **114**, 737 (1992). Reprinted in [683].

[770] R. DeVore, B. Jawerth, and B. Lucier, “Image Compression Through Wavelet Transform Coding,” *IEEE Trans. Inform. Th.*, **38**, 719 (1992).

[771] M. Farge, “Wavelet Transforms and their Applications to Turbulence,” *Ann. Rev. Fluid Mech.*, **24**, 395 (1992).

[772] J. N. Bradley, C. M. Brislawn, and T. Hopper, “The FBI Wavelet/Scalar Quantization Standard for Grey-Scale Fingerprint Image Compression,” *Proc. SPIE*, **1961**, 293 (1993).

[773] C. M. Brislawn, “Fingerprints Go Digital,” *Notices AMS*, **42** no. 11, 1278 (1995).

[774] C. M. Brislawn, et al., “FBI Compression Standard for Digitized Fingerprint Images,” *Proc. SPIE*, **2847**, 344 (1996).

[775] E. J. Stollnitz, T. D. DeRose, and D. H. Salesin, “Wavelets for Computer Graphics: A Primer, part 1,” *IEEE Comput. Graph. Appl.*, **15**, 76 (1995).

[776] G. Pan, “Orthogonal Wavelets with Applications in Electromagnetics,” *IEEE Trans. Magn.*, **32**, 975 (1996).

[777] R. Zaciu, et al., “Image Compression Using an Overcomplete Discrete Wavelet Transform,” *IEEE Trans. Consum. Electron.*, **42**, 500 (1996).

[778] N. Erdol and F. Basbug, “Wavelet Transform Based Adaptive Filters: Analvsis and New Results,” *IEEE Trans. Signal Process.*, **44**, 2163 (1996).

[779] A. Bijaoui, et al., “Wavelets and the Study of the Distant Universe,” *Proc. IEEE*, **84**, 670 (1996).

[780] M. Unser and A. Aldroubi, “A Review of Wavelets in Biomedical Applications,” *Proc. IEEE*, **84**, 626 (1996).

[781] B. K. Alsberg, A. M. Woodward, and D. B. Kell, “An Introduction to Wavelet Transforms for Chemometricians: A Time-Frequency Approach,” *Chemometr. Intell. Lab. Syst.*, **37**, 215 (1997).

[782] B. K. Alsberg, et al., “Wavelet Denoising of Infrared Spectra,” *Analyst*, **122**, 645 (1997).

[783] B. Walczak and D. L. Massart, “Wavelets – Something for Analytical Chemistry?,” *Trends Anal. Cem.*, **15**, 451 (1997).

[784] A. Chambolle, et al., “Nonlinear Wavelet Image Processing: Variational Problems, Compression and Noise Removal Through Wavelet Shrinkage,” *IEEE Trans. Imag. Process.*, **7**, 319 (1998).

[785] A. K-M. Leung, F-T. Chau, and J-B. Gao, “A Review on Applications of Wavelet Transform Techniques in Chemical Analysis: 1989–1997,” *Chemometr. Intell. Lab. Syst.*, **43**, 165 (1998).

[786] G. Strang, “The Discrete Cosine Transform,” *SIAM Rev.*, **41**, 135 (1999).

[787] C. Torrence and G. P. Compo, “A Practical Guide to Wavelet Analysis,” *Bull. Amer. Meteor. Soc.*, **79**, 621 (1998).

[788] J. B. Ramsey, “The Contribution of Wavelets to the Analysis of Economic and Financial Data,” *Phil. Trans. Roy. Soc. Lond. A*, **357**, 2593 (1999).

[789] M. W. Marcellin, et al., “An Overview of JPEG2000,” *Proc. Data Compression Conf.*, Snowbird, Utah, March 2000, p. 523.

[790] ISO/IEC JTC1/SC29/WG1/N1646R, JPEG 2000 Part I Final Committee Draft Version 1.0, Mar. 2000, available from `http://www.jpeg.org/public/fcd15444-1.pdf`.

[791] C. Christopoulos, A. Skodras, and T. Ebrahimi, “The JPEG2000 Still Image Coding System: An Overview,”, *IEEE Trans. Consum. Electron.*, **46**, 1103 (2000).

[792] C-H. Lee, Y-J Wang, and W-L Huang, “A Literature Survey of Wavelets in Power Engineering Applications,” *Proc. Natl. Sci. Counc. ROC(A)*, **24**, 249 (2000).

[793] C.H. Kim and R. Aggarwal, “Wavelet Transforms in Power Systems, Part 1: General Introduction to the Wavelet Transforms,” *Power Eng. J.*, **14**, 81 (2000); and “Part 2: Examples of Application to Actual Power System Transients,”, *ibid.*, **15**, 193 (2000).

[794] S. G. Chang, B. Yu, and M. Vetterli, “Adaptive Wavelet Thresholding for Image Denoising and Compression,” *IEEE Trans. Imag. Process.*, **9**, 1532 (2000).

[795] D. B. H. Tay, “Rationalizing the Coefficients of Popular Biorthogonal Wavelet Filters,” *IEEE Trans. Circ. Syst. Video Tech.*, **10**, 998 (2000).

[796] B. E. Usevitch, “A Tutorial on Modern Lossy Wavelet Image Compression: Foundations of JPEG 2000,” *IEEE SP Mag.*, Sept. 2001, p. 22.

[797] M. D. Adams, “The JPEG-2000 Still Image Compression Standard,” ISO/IEC JTC 1/SC 29/WG1 N 2412, Sept. 2001, Available from `http://www.ece.ubc.ca/~mdadams`.

[798] J-L, Starck and F. Murtagh, “Astronomical Image and Signal Processing Looking at Noise, Information, and Scale,” *IEEE SP Mag.*, p.30, Mar. 2001.

[799] J. B. Ramsey, “Wavelets in Economics and Finance: Past and Future,” *Stud. Nonlin. Dynam. Econometr.*, 2002.

[800] M. Unser and T. Blu, “Mathematical Properties of the JPEG2000 Wavelet Filters,” *IEEE Trans. Imag. Process.*, **12**, 1080 (2003).

[801] F. Truchetet and O. Laligant, “Wavelets in Industrial Applications: A Review,” *Proc. SPIE*, **5607**, 1 (2004).

[802] M. J. Fadili and E. T. Bullmore, “A Comparative Evaluation of Wavelet-Based Methods for Hypothesis Testing of Brain Activation Maps,” *NeuroImage*, **23**, 1112 (2004).

[803] M. N. O. Sadiku, C. M. Akujuobi, and R. C. Garcia, “An Introduction to Wavelets in Electromagnetics,” *IEEE Microwave Mag.*, **6**, no.5, p.63, June 2005.

[804] P. S. Addison, “Wavelet Transforms and the ECG: A Review,” *Physiol. Meas.*, **26**, R155 (2005).

[805] M. Kaboudan, “Computational Forecasting of Wavelet-converted Monthly Sunspot Numbers,” *J. Appl. Statist.*, **33**, 925 (2006).

[806] P. Liò, “Wavelets in Bioinformatics and Computational Biology: State of Art and Perspectives,” *Bionform. Rev.*, **21**, 207 (2007).

[807] P. M. Crowley, “A Guide to Wavelets for Economists,” *J. Econ. Surveys*, **21**, 207 (2007).

[808] J. E. Fowler and B. Pesquet-Popescu, ”An Overview on Wavelets in Source Coding, Communications, and Networks,” *EURASIP J. Imag. Vid. Process.*, vol. 2007, Article ID 60539, (2007).

[809] I. Balasingham and T. A. Ramstad, J. E. Fowler and B. Pesquet-Popescu, ”Are the Wavelet Transforms the Best Filter Banks for Image Compression?” *EURASIP J. Imag. Vid. Process.*, vol. 2008, Article ID 287197, (2008).

[810] F. Truchetet and O. Laligant, “Review of Industrial Applications of Wavelet and Multiresolution-Based Signal and Image Processing,” *J. Electron. Imag.*, **17**, 031102 (2008)

[811] H. Hashish, S. H. Behiry, and N.A. El-Shamy, “Numerical Integration Using Wavelets,” *Appl. Math. Comput.* **211**, 480 (2009).

[812] B. Mandelbrot and J. W. Van Ness, “Fractional Brownian Motions: Fractional Noises and Applications,” *SIAM Rev.*, **10**, 422 (1968).

[813] S. Granger and R. Joyeux, “An Introduction to Long-Memory Time Series Models and Fractional Differencing,” *J. Time Ser. Anal.*, **1**, 15 (1980).

[814] J. R. M. Hosking, “Fractional Differencing,” *Biometrika*, **68**, 165 (1981).

[815] G. Wornell, “A Karhunen-Loève Like Expansion for $1/f$ Processes via Wavelets,” *IEEE Trans. Inform. Th.*, **36**, 859 (1990).

[816] G. Wornell and A. V. Oppenheim, “Wavelet-Based Representations for a Class of Self-Similar Signals with Application to Fractal Modulation,” *IEEE Trans. Inform. Th.*, **38**, 785 (1992).

[817] P. Flandrin, “Wavelet Analysis and Synthesis of Fractional Brownian Motion,” *IEEE Trans. Inform. Th.*, **38**, 910 (1992).

[818] P. Abry, et al., “The Multiscale Nature of Network Traffic,” *IEEE SP Mag.*, **19**, no. 3, 28, May 2002.

[819] R. A. DeVore and B. J. Lucier, “Fast Wavelet Techniques for Near-Optimal Image Processing,” *MILCOM ’92, IEEE Mil. Commun. Conf.*, p.1129, (1992).

[820] D. Donoho, “Unconditional Bases are Optimal Bases for Data Compression and Statistical Estimation,” *Appl. Computat. Harmon. Anal.*, **1**, 100 (1993).

[821] D. L. Donoho and I. M. Johnstone, “Ideal Spatial Adaptation by Wavelet Shrinkage,” *Biometrika*, **81**, 425 (1994).

[822] , D. L. Donoho, “Denoising by Soft Thresholding,” *IEEE Trans. Inform. Th.*, **41**, 613 (1995).

[823] , D. L. Donoho, et al., “Wavelet Shrinkage: Asymptopia?,” *J. Roy. Statist. Soc., Ser. B*, **57**, 301 (1995).

[824] D. L. Donoho and I. M. Johnstone, “Adapting to Unknown Smoothness via Wavelet Shrinkage,” *J. Amer. Statist. Assoc.*, **90**, 1200 (1995). Reprinted in [683].

[825] A. Antoniadis, “Smoothing Noisy Data with Tapered Coiflets Series,” *Scand. J. Statist.*, **23**, 313 (1996).

[826] F. Abramovich and B. W. Silverman, “Wavelet Decomposition Approaches to Statistical Inverse Problems,” *Biometrika*, **85**, 115 (1998).

[827] D. L. Donoho, et al., “Data Compression and Harmonic Analysis,” *IEEE Trans. Inform. Th.*, **44**, 2435 (1998).

[828] F. Abramovich, T. Sapatinas, and B. W. Silverman, “Wavelet Thresholding via Bayesian Approach,” *J. Roy. Statist. Soc., Ser. B.*, **60**, 725 (1998).

[829] B. W. Silverman, “Wavelets in Statistics: Beyond the Standard Assumptions,” *Phil. Trans. Roy. Soc. Lond. A*, **357**, 2459 (1999).

[830] G. P. Nason and R. von Sachs, “Wavelets in Time-Series Analysis,” *Phil. Trans. Roy. Soc. Lond. A*, **357**, 2511 (1999).

[831] F. Abramovich, T. C. Baily, and T. Sapatinas, “Wavelet Analysis and Its Statistical Applications,” *Statistician*, **49**, 1 (2000).

[832] A. Antoniadis, J. Bigot, and T. Sapatinas, “Wavelet Estimators in Nonparametric Regression: A Comparative Simulation Study,” *J. Statist. Softw.*, **6**, 1 (2001).

[833] A. Antoniadis and J. Fan, “Regularization of Wavelet Approximations,” *J. Amer. Statist. Assoc.*, **96**, 939 (2001).

[834] `http://www.cmap.polytechnique.fr/~bacry/LastWave`, LastWave, Emmanuel Bacry.

[835] `http://www.cs.kuleuven.ac.be/~wavelets`, Uytterhoeven, et al., C++ implementation.

[836] `http://www-stat.stanford.edu/~wavelab/` Wavelab.

[837] `http://www.dsp.rice.edu/software/RWT` Rice Wavelet Toolbox.

[838] `http://paos.colorado.edu/research/wavelets`, Torrance and Compo.

[839] `http://www.curvelet.org/`, Curvelets.

[840] `http://www.stats.bris.ac.uk/~wavethresh`, Wavethresh in R.

[841] `http://taco.poly.edu/WaveletSoftware/`, S. Cai and K. Li.

[842] `http://www2.isye.gatech.edu/~brani/wavelet.html`, B. Vidakovic.

[843] `http://www-lmc.imag.fr/SMS/software/GaussianWaveDen/index.html`, A. Antoniadis, J. Bigot, and J. Sapatinas.

[844] `http://www.atmos.washington.edu/~wmtsa/`, Percival and Walden, WMTSA toolbox.

[845] `http://cas.ensmp.fr/~chaplais/UviWave/About_UviWave.html`, Uvi-Wave.

[846] `http://inversioninc.com/wavelet.html`, N. H. Getz, see Ref. [737].

[847] `http://www.math.rutgers.edu/~ojanen/wavekit/`, H. Ojanen, Wavekit.

[848] `http://cam.mathlab.stthomas.edu/wavelets/packages.php`, P. Van Fleet, see [685].

Wiener and Kalman Filtering

[849] N. Wiener, *Extrapolation, Interpolation and Smoothing of Stationary Time Series with Engineering Applications*, New York, Wiley, 1949.

[850] A. N. Kolmogorov, Sur l’Interpolation et Extrapolation des Suites Stationnaires, *C. R. Acad. Sci.*, **208**, 2043–2045 (1939). See also Interpolation and Extrapolation of Stationary Random Sequences, and Stationary Sequences in Hilbert Space, reprinted in T. Kailath, Ed., *Linear Least-Squares Estimation*, Stroudsburg, PA, Dowden, Hutchinson, and Ross, 1977.

[851] H. W. Bode and C. E. Shannon, A Simplified Derivation of Linear Least-Squares Smoothing and Prediction Theory, *Proc. IRE*, **38**, 417-425 (1950).

[852] R. E. Kalman, “A New Approach to Linear Filtering and Prediction Problems,” *Trans. ASME, Ser. D, J. Basic Eng.*, **82**, 34–45 (1960).

[853] R. E. Kalman and R. S. Bucy, “New Results in Linear Filtering and Prediction Theory,” *Trans. ASME, Ser. D, J. Basic Eng.*, **83**, 95–107 (1961).

[854] R. E. Kalman, “New Methods in Wiener Filtering Theory,” in *Proc. First Symp. Engineering Appl. of Random Function Theory and Probability*, J. L. Bogdanoff and F. Kozin, eds., Wiley, New York, 1963, pp. 270–388.

[855] H. W. Sorenson, “Least-Squares Estimation: From Gauss to Kalman,” *IEEE Spectrum*, **7**, 63 (1970).

[856] T. Kailath, “An Innovations Approach to Least-Squares Estimation. Part I: Linear Filtering in Additive White Noise,” *IEEE Trans. Autom. Control*, **AC-13**, 646–655 (1968).

[857] P. Whittle, *Prediction and Regulation*, New York: Van Nostrand Reinhold, 1963.

[858] A. M. Yaglom, *Theory of stationary Random Functions*, Englewood Cliffs, NJ, Prentice-Hall, 1962.

[859] T. Kailath, Some Topics in Linear Estimation, in M. Hazewinkel and J. C. Willems, Eds., *Stochastic Systems: The Mathematics of Filtering and Identification*, Boston, D. Reidel Publications, 1981, pp.307–350.

[860] A. H. Jazwinski, *Stochastic Processes and Filtering Theory*, Dover Publications, NY, 2007, reporting of the Academic Press, 1970 edition.

[861] A. P. Sage and J. L. Melsa, *Estimation Theory with Applications to Communication and Control*, New York, McGraw-Hill, 1971.

[862] A. Gelb, *Applied Optimal Estimation*, Cambridge, MA, MIT Press, 1974.

[863] B. Anderson and J. Moore, *Optimal Filtering*, Englewood Cliffs, NJ, Prentice-Hall, 1979. Available online from: `http://users.cecs.anu.edu.au/~john/papers/index.html`

[864] M. Srinath and P. Rajasekaran, *Introduction to Statistical Signal Processing*, New York, Wiley, 1979.

[865] T. Kailath, A. H. Sayed, and B. Hassibi, *Linear Estimation*, Prentice Hall, Englewood Cliffs, NJ, 2000.

[866] T. Kailath, “A View of Three Decades of Linear Filtering Theory,” *IEEE Trans. Info. Theory*, **IT-20**, 146 (1974).

[867] T. R. Kronhamn, “Geometric Illustration of the Kalman Filter Gain and Covariance Update Algorithms,” *IEEE Control Syst. Magazine*, May 1985, p. 41.

[868] B. Friedland, “Optimum Steady-State Position and Velocity Estimation Using Noisy Sampled Position Data,” *IEEE Trans. Aerosp. Elect. Syst.*, **AES-9**, 906 (1972).

[869] P. R. Kalata, “The Tracking Index: A Generalized Parameter for α–β and α–β–γ Target Trackers,” *IEEE Trans. Aerosp. Elect. Syst.*, **AES-20**, 174 (1984).

[870] R. T. Benedict and G. W. Bordner, “Synthesis of an Optimal Set of Radar Track-While-Scan Smoothing Equations,” *IRE Trans. Automat. Contr.*, **AC-7**, 27 (1962).

[871] S. J. Orfanidis, “An Exact Solution of the Time-Invariant Discrete Kalman Filter,” *IEEE Trans. Automat. Contr.*, **AC-27** ,240 (1982).

[872] S. J. Orfanidis, “A Group Theoretical Approach to Optimal Estimation and Control,” *J. Math. Anal. Appl.*, **97**, 393 (1983).

[873] J. E. Gray and G. J. Foster, “An Extension of the Tracking Index Concept to Non-Kalman Filter Selection Techniques,” *Proc. 13th Southeastern Symp. Systems Theory*, p.373, March 1998.

[874] E. Brookner, *Tracking and Kalman Filtering Made Easy*, Wiley, New York, 1998.

[875] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Estimation with Applications to Tracking and Navigation*, Wiley, New York, 2001.

[876] K. V. Ramachandra, “Optimum Steady-State Position, Velocity, and Acceleration Estimation Using Noisy Sampled Position Data,” *IEEE Trans. Aerosp. Elect. Syst.*, **AES-23**, 705 (1987).

[877] W. F. Arnold, III and A. J. Laub, “Generalized Eigenproblem Algorithms and Software for Algebraic Riccati Equations,,, *Proc. IEEE*, **72**, 1746 (1984).

[878] P. Benner, A. J. Laub, and V. Mehrmann, “A Collection of Benchmark Examples for the Numerical Solution of Algebraic Riccati Equations II: Discrete-Time Case,” Dec. 1995, available online from `http://www.tu-chemnitz.de/sfb393/Files/PS/spc95-23.ps.gz`

[879] L. A. McGee and S. F. Schmidt, “Discovery of the Kalman Filter as a Practical Tool for Aerospace and Industry,” NASA-TM-86847, 1985, available from `http://ntrs.nasa.gov/`, Document ID: 19860003843.

[880] M. W. A. Smith and A. P, Roberts, “An Exact Equivalence Between thr Discrete- and Continuous-Time Formulations of the Kalman Filter,” *Math and Comput. in Simulation*, **20**, 102 (1978).

[881] A. E. Bryson and Y-C Ho, *Applied Optimal Control*, Hemisphere Publishing, Washington, 1975.

[882] A. E. Bryson and M. Frazier, “Smoothing for Linear and Non-Linear Dynamic Systems,” *Proc. Opt. Syst, Synthesis Conf.*, 1962, p.354, reprinted in T. Kailath, ed., *Linear Least-Squares Estimation*, Dowden, Hutchinson, and Ross, Stroudsburg, PA, 1977.

[883] H. E. Rauch, “Solutions to the Linear Smoothing Problem,” *IEEE Trans. Automat. Contr.*, **AC-8**, 371 (1963).

[884] H. E. Rauch, F. Tung, and C. T. Striebel, “Maximum Likelihood Estimates of Linear Dynamic Systems,” *AIAA J.*, **3**, 1445 (1965).

[885] P. De Jong, “A Cross-Validation Filter for Time Series Models,” *Biometrika*, **75**, 594 (1988).

[886] P. De Jong, “Smoothing and Interpolation with the State-Space Model,” *J. Amer. Statist. Assoc.* **84**, 1085 (1989).

Kalman Filtering – Square Root Algorithms

[887] D. Q. Mayne, “A Solution of the Smoothing Problem for Linear Dynamic Systems,” *Automatica*, **4**, 73 (1966).

[888] P. Dyer and S. McReynolds, “Extension of square-root filtering to include process noise,” *J. Optim. Th. Appl.*, **3**, 444 (1969).

[889] P. G. Kaminski, A. E. Bryson, and S. F. Schmidt, “Discrete Square-Root Filtering—A Survey of Current Techniques,” *IEEE Trans. Automat. Contr.,* **AC-16**, 727 (1971).

[890] G. J. Bierman, “A Comparison of Discrete Linear Filtering Algorithms,” *IEEE TRans. Aerosp. Electron. Syst.* **AES-9**, 28 (1973).

[891] M. Morf and T. Kailath, “Square-Root Algorithms for Least-Squares Estimation,” *IEEE Trans. Automat. Contr.*, **AC-20**, 487 (1975).

[892] G. J. Bierman, *Factorization Methods for Discrete Sequential Estimation*, Academic, New York, 1977, and Dover Publications, 2006.

[893] G. J. Bierman, “A New Computationally Efficient Fixed-Interval, Discrete-Time Smoothers,” *Automatica*, **19**, 503 (1983).

[894] M. Verhaegen and P. Van Dooren, “Numerical Aspects of Different Kalman Filter Implementations,” *IEEE Trans. Automat. Contr.*, **AC-31**, 907 (1986).

[895] S. R. McReynolds, “Covariance factorization algorithms for fixed-interval smoothing of linear discrete dynamic systems,” *IEEE Trans. Automat. Contr.*, **AC-35**, 1181 (1990).

[896] P. Park and T. Kailath, "Square-root Bryson-Frazier smoothing algorithms", *IEEE Trans. Automat. Contr.*, **AC-40**, 761 (1995).

Kalman Filtering – ML and EM Algorithms

[897] F. Schweppe, “Evaluation of Likelihood Functions for Gaussian Signals,” *IEEE Trans. Inform. Th.*, **IT-11**, 61 (1965).

[898] R. L. Kashyap, “Maximum Likelihood Identification of Stochastic Linear Systems,” *IEEE Trans. Automat. Contr.*, **AC-15**, 25 (1970).

[899] R. K. Mehra, “On the Identification of Variances and Adaptive Kalman Filtering,” *IEEE Trans. Automat. Contr.*, **AC-15**, 175 (1970).

[900] R. K. Mehra, “On-Line Identification of Linear Dynamic Systems with Applications to Kalman Filtering,” *IEEE Trans. Automat. Contr.*, **AC-16**, 12 (1971).

[901] N. K. Gupta and R. K. Mehra, “Computational Aspects of Maximum Likelihood Estimation and Reduction in Sensitivity Function Calculations,” *IEEE Trans. Automat. Contr.*, **AC-19**, 774 (1974).

[902] A. C. Harvey, *Forecasting Structural Time Series Models and the Kalman Filter*, Cambridge Univ. Press, Cambridge, 1989.

[903] J. Durbin and S. J. Koopman, *Time Series Analysis by State Space Methods*, Oxford Univ. Press, Oxford, 2001.

[904] R. H. Shumway and D. S. Stoffer, *Time Series Analysis and Its Applications*, Springer, New York, 2006.

[905] A. P. Dempster, N. M. Laird, and D. B. Rubin, “Maximum Likelihood from Incomplete Data via the EM Algorithm,” *J. Roy. Stat. Soc., Ser. B*, **39**, 1 (1977).

[906] G. J. McLachlan, T. Krishnan, *The EM Algorithm and Extensions*, 2nd ed., Wiley, Hoboken, NJ, 2008.

[907] P. A. Ruud, “Extensions of Estimation Methods Using the EM Algorithm,” *J. Econometrics*, **49**, 305 (1991).

[908] T. K. Moon, “The Expectation-Maximization Algorithm,” *IEEE Sig. Proc. Mag.*, **13**, no.6, 47 (1996).

[909] R. H. Shumway and D. S. Stoffer, “An Approach to Time Series Smoothing and Forecasting Using the EM Algorithm,” *J. Time Ser. Anal.*, **3**, 253 (1982).

[910] M. W. Watson and R. F. Engle, “Alternative Algorithms for the Estimation of Dynamic Factor, MIMIC and Varying Coefficient Regression Models,” *J. Econometrics*, **23**, 385 (1983).

[911] Z. Ghahramani and G. Hinton, “Parameter Estimation for Linear Dynamic Systems,” Tech. Rep. CRG-TR-96-2, Dept. Computer Science, University of Toronto, 1996, available from: `http://www.cs.toronto.edu/~hinton/absps/tr-96-2.pdf`

[912] G. W. Cobb, “The Problem of the Nile: Conditional Solution to a Changepoint Problem,” *Biometrika*, **65**, 243 (1978).

[913] D. R. Hunter and K. Lange, “A tutorial on MM algorithms,” *Am. Statistician*, **58**, 30 (2004).

[914] Y. Sun, P. Babu, and D. P. Palomar, “Majorization-Minimization Algorithms in Signal Processing, Communications, and Machine Learning,” *IEEE Trans. Signal Process.*, **65**, 794 (2017).

Linear Prediction

[915] G. P. Box and G. M. Jenkins, *Time Series Analysis, Forecasting, and Control*, San Francisco, Holden-Day, 1970.

[916] P. Whittle, *Prediction and Regulation*, New York, Van Nostrand Reinhold, 1963.

[917] J. Makhoul, Linear Prediction: A Tutorial Review, *Proc. IEEE*, **63**, 56 (1975).

[918] N. Levinson, The Wiener RMS Error Criterion in Filter Design and Prediction, *J. Math. Physics*, **25**, 261 (1947).

[919] J. Durbin, The Fitting of Time Series Models, *Rev. Inst. Int. Stat.*, **28**, 344 (1973).

[920] J. D. Markel and A. H. Gray, Jr., *Linear Prediction of Speech*, New York, Springer-Verlag, 1976.

[921] E. A. Robinson, *Multichannel Time-Series Analysis with Digital Computer Programs*, (2nd ed.), Houston, TX, Goose Pond Press, 1983.

[922] E. A. Robinson, *Statistical Communication and Detection*, New York, Hafner, 1967.

[923] S. Tretter, *Introduction to Discrete-Time Signal Processing*, New York, Wiley, 1976.

[924] E. A. Robinson and S. Treitel, *Geophysical Signal Analysis*, Englewood Cliffs, NJ, Prentice-Hall, 1980.

[925] E. A. Robinson and S. Treitel, Maximum Entropy and the Relationship of the Partial Autocorrelation to the Reflection Coefficients of a Layered System, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-28**, 22 (1980).

[926] S. M. Kay and S. L. Marple, Spectrum Analysis–A Modern Perspective, *Proc. IEEE*, **69**, 1380 (1981).

[927] S. Haykin, Ed., *Nonlinear Methods of Spectral Analysis*, New York, Springer-Verlag, 1979.

[928] A. Papoulis, Predictable Processes and Wold’s Decomposition: A Review, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-33**, 933 (1985).

[929] O. Barndorff-Nielsen and G. Schou, On the Parametrization of Autoregressive Models by Partial Autocorrelations, *J. Multiv. Anal.*, **3**, 408 (1973).

[930] F. L. Ramsey, Characterization of the Partial Autocorrelation Function, *Ann. Stat.*, **2**, 1296 (1974).

[931] M. Morf, A. Vieira, and T. Kailath, Covariance Characterization by Partial Autocorrelation Matrices. *Ann. Stat.*, **6**, 643 (1978).

[932] R. E. Kalman, On Partial Realizations, Transfer Functions, and Canonical Forms, *Acta Polytech. Scandinav.*, Math. Comput. Sci. Series, **13**, 9 (1979).

[933] R. E. Kalman, Realization of Covariance Sequences, in I. Gohberg, Ed., *Toeplitz Centennial, Operator Theory: Advances and Applications*, vol. 4, Boston, Birkhäuser, 1982.

[934] W. Gragg and A. Lindquist, On the Partial Realization Problem, *Lin. Alg. Appl.*, **50**, 277 (1983).

[935] T. K. Citron, A. M. Bruckstein, and T. Kailath, An Inverse Scattering Approach to the Partial Realization Problem, *Proc. 1984 IEEE Int. Conf. Decision and Control*, Las Vegas, NV, p. 1503.

[936] T. T. Georgiou, Realization of Power Spectra from Partial Covariance Sequences, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-35**, 438 (1987).

[937] S. Saito and K. Nakata, *Fundamentals of Speech Processing*, New York, Academic, 1985.

[938] N. I. Aheizer and M. Krein, *Some Questions in the Theory of Moments*, Providence, RI, Am. Math Soc., 1962.

[939] R. R. Bitmead and B. D. O. Anderson, Asymptotically Fast Solution of Toeplitz and Related Systems of Linear Equations, *Lin. Alg. Appl.*, **34**, 103 (1980).

[940] R. P. Brent, F. G. Gustavson, and D. Y. Y. Yun, Fast Solution of Toeplitz Systems of Equations and Computation of Padé Approximants, *J. Algorithms*, **1**, 259 (1980).

[941] H. M. Ahmed. J. M. Delosme, and M. Morf, Highly Concurrent Computing Structures for Matrix Arithmetic and Signal Processing, *Computer Magazine*, **15**, 65 (Jan. 1982).

[942] H. T. Kung, Why Systolic Architectures?, *Computer Magazine*, **15**, 37 (Jan. 1982).

[943] R. P. Brent and F. T. Luk, A Systolic Array of the Linear-Time Solution of Toeplitz Systems of Equations, *J. VLSI Comput. Syst.*, **1**, 1 (1983).

[944] S. K. Rao and T. Kailath, Orthogonal Digital Filters for VLSI Implementation, *IEEE Trans. Circ. Syst.*, **CAS-31**, 933 (1984).

[945] D. R. Sweet, Fast Toeplitz Orthogonalization, *Numer. Math.*, **43**, 1 (1984).

[946] S. Y. Kung, On Super Computing with Systolic/Wavefront Array Processors, *Proc. IEEE*, **72**, 867 (1984).

[947] S. Y. Kung, VLSI Array Processors, *ASSP Magazine*, **2**, no.3, 4, (1985).

[948] S. Y. Kung, VLSI Signal Processing: From Transversal Filtering to Concurrent Array Processing, in S. Y. Kung, H. J. Whitehouse, and T. Kailath, Eds., *VLSI and Modern Signal Processing*, Englewood Cliffs, NJ, Prentice-Hall, 1985.

[949] G. R. Nudd and J. G. Nash, Application of Concurrent VLSI Systems to Two-Dimensional Signal Processing, *ibid.*

[950] R. Schreiber, Systolic Linear Algebra Machines in Digital Signal Processing, *ibid.*

[951] P. Dewilde, E. Deprettere, and R. Nouta, Parallel and Pipelined VLSI Implementation of Signal Processing Algorithms, *ibid.*

[952] R. Kumar, A Fast Algorithm for Solving a Toeplitz System of Equations, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-33**, 254 (1985).

[953] J. R. Bunch, Stability of Methods for Solving Toeplitz Systems of Equations, *SIAM J. Sci. Stat. Comput.*. **6**, 349 (1985).

[954] A. D. McAulay, Parallel AR Computation with a Reconfigurable Signal Processor, *Proc. 1986 IEEE Int. Conf. Acoust., Speech, Signal Process.*, Tokyo, p.1365.

[955] A. W. Bojanczyk, Systolic Implementation of the Lattice Algorithm for Least Squares Linear Prediction Problems, *Lin. Alg. Appl.*, **77**, 27 (1986).

[956] F. De Hoog, A New Algorithm for Solving Toeplitz Systems of Equations, *Lin. Alg. Appl.*, **88/89**, 123 (1987).

[957] H. Kimura and T. Osada, Canonical Pipelining of Lattice Filters, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-35**, 878 (1987).

[958] P. Dewilde and E. F. Deprettere, Modelling VLSI Interconnects as an Inverse Scattering Problem, *Proc. 1987 IEEE Int. Conf. Circuits and Systems*, Philadelphia, PA, p.147.

[959] Y. Bistritz, Zero Location with Respect to the Unit Circle of Discrete-Time Linear System Polynomials, *Proc. IEEE*, **72**, 1131 (1984).

[960] P. Delsarte and Y. Genin, The Split Levinson Algorithm, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-34**, 470, (1986).

[961] Y. Bistritz, H. Lev-Ari, and T. Kailath, Immitance-Domain Levinson Algorithms, *Proc. 1986 IEEE Int. Conf: Acoust., Speech, Signal Process.*, Tokyo, p.253.

[962] P. Delsarte and Y. Genin, On the Splitting of Classical Algorithms in Linear Prediction Theory, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-35**, 645 (1987).

[963] Y. Bistritz, H. Lev-Ari, and T. Kailath, Complexity Reduced Lattice Filters for Digital Speech Processing, *Proc. 1987 IEEE Int. Conf: Acoust., Speech, Signal Process.*, Dallas, TX, p.21.

[964] Y. Bistritz and T. Kailath, Fast Algorithms for Non-Hermitian Quasi-Toeplitz Matrices, *Proc. 1987 IEEE Int. Conf. Circuits and Systems*, Philadelphia, PA, p.1068.

[965] H. Krishna and S. D. Morgera, The Levinson Recurrence and Fast Algorithms for Solving Toeplitz Systems of Linear Equations, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-35**, 839 (1987).

[966] S. D. Morgera and H. Krishna, Generalized Levinson/Szegö Complex Recurrences for a Class of Second-Order Nonstationary Stochastic Processes, *Proc. 1987 IEEE Int. Conf. Circuits and Systems*, Philadelphia, PA, p.84.

[967] G. Martinelli, G. Orlandi, and P. Burrascano, Yule-Walker Equations and Bartlett's Bisection Theory, *IEEE Trans. Circ. Syst.*, **CAS-32**, 1074 (1985).

[968] A. J. Berkhout, Stability and Least-Squares Estimation, *Automatica*, **11**, 637 (1975).

[969] A. Vieira and T. Kailath, Another Approach to the Schur-Cohn Criterion, *IEEE Trans. Circuits and Systems*, **CAS-24**, 218-220 (April 1977).

[970] R. J. Duffin, Algorithms for Classical Stability Problems, *SIAM Rev.*, **11**, 196 (1969).

[971] P. P. Vaidyanathan and S. K. Mitra, A Unified Structural Interpretation of Some Well-Known Stability-Test Procedures for Linear Systems, *Proc. IEEE*, **75**, 478 (1987).

[972] N. I. Achiezer, *The Classical Moment Problem*, Edinburgh, Oliver and Boyd, 1965.

[973] G. Szegö, *Orthogonal Polynomials*, Providence, RI, American Mathematical Society, 1958.

[974] E. A. Robinson and S. Treitel, Digital Signal Processing in Geophysics, in A. Oppenheim, Ed., *Applications of Digital Signal Processing*, Englewood Cliffs, NJ, Prentice-Hall, 1978.

[975] S. Treitel and E. A. Robinson, The Design of High-Resolution Digital Filters, *IEEE Trans. Geosci. Electron.*, **GE-4**, 25 (1966).

[976] J. Claerbout, *Fundamentals of Geophysical Data Processing*, New York, McGraw-Hill, 1976.

[977] I. C. Gohberg and I. A. Fel'dman, *Convolution Equations and Projection Methods for their Solution*, Providence, RI, American Mathematical Society, 1974.

[978] W. F. Trench, An Algorithm for the Inversion of Finite Toeplitz Matrices, *J. Soc. Ind. Appl. Math.*, **12**, 515 (1964).

[979] S. Zohar, Toeplitz Matrix Inversion: The Algorithm of W. F. Trench, *J. Assoc. Comput. Mach.*, **16**, 592 (1969).

[980] S. Zohar, The Solution of a Toeplitz Set of Linear Equations, *J. Assoc. Comput. Mach.*, **21**, 272 (1974).

[981] T. Kailath, A. Vieira, and M. Morf, Inverses of Toeplitz Operators, Innovations and Orthogonal Polynomials, *SIAM Rev.*, **20**, 106 (1978).

[982] H. Lev-Ari and T. Kailath, Triangular Factorization of Structured Hermitian Matrices, in I. Gohberg, Ed., *I. Schur Methods in Operator Theory and Signal Processing, Operator Theory: Advances and Applications*, vol.18, Boston, Birkhäuser, 1986.

[983] I. Gohberg, T. Kailath, and I. Koltracht, Efficient Solution of Linear Systems of Equations with Recursive Structure, *Lin. Alg. Appl.*, **80**, 81 (1986).

[984] I. Gohberg, T. Kailath, I. Koltracht, and P. Lancaster, Linear Complexity Parallel Algorithms for Linear Systems of Equations with Recursive Structure, *Lin. Alg. Appl.*, **88/89**, 271 (1987).

[985] I. Schur, On Power Series Which Are Bounded in the Interior of the Unit Circle I and II, in I. Gohberg, Ed., *I. Schur Methods in Operator Theory and Signal Processing, Operator Theory: Advances and Applications*, vol.18, Boston, Birkhäuser, 1986.

[986] T. Kailath, A Theorem of I. Schur and Its Impact on Modern Signal Processing, *ibid.*

[987] E. H. Bareiss, Numerical Solution of Linear Equations with Toeplitz and Vector Toeplitz Matrices, *Numer. Math.*, **13**, 404 (1969).

[988] J. Rissanen, Algorithms for Triangular Decomposition of Block Hankel and Toeplitz Matrices with Application to Factoring Positive Matrix Polynomials, *Math. Comp.*, **27**, 147 (1973).

[989] J. Rissanen, Solution of Linear Equations with Hankel and Toeplitz Matrices, *Numer. Math.*, **22**, 361 (1974).

[990] J. Le Roux and C. J. Gueguen, A Fixed Point Computation of Partial Correlation Coefficients, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-25**, 257 (1977).

[991] P. Dewilde, A. Vieira, and T. Kailath, On the Generalized Szegö-Levinson Realization Algorithm for Optimal Linear Predictors Based on a Network Synthesis Approach, *IEEE Trans. Circuits Syst.*, **CAS-25**, 663 (1978).

[992] P. Delsarte, Y. Genin, and Y. Kamp, Schur Parametrization of Positive Definite Block-Toeplitz Systems, *SIAM J. Appl. Math.*, **36**, 34 (1979).

[993] T. Kailath, S. Y. Kung, and M. Morf, Displacement Rank of Matrices and Linear Equations, *J. Math. Anal. Appl.*, **68**, 395 (1979).

[994] P. Dewilde and H. Dym, Schur Recursions, Error Formulas, and Convergence of Rational Estimators for Stationary Stochastic Sequences, *IEEE Trans. Inform. Th.*, **IT-27**, 446 (1981).

[995] P. Dewilde, J. T. Fokkema, and I. Widya, Inverse Scattering and Linear Prediction: The Continuous Time Case, in M. Hazewinkel and J. C. Willems, Eds., *Stochastic Systems: The Mathematics of Filtering and Identification and Applications*, Boston, Reidel, 1981.

[996] E. Jonkheere and P. Delsarte, Inversion of Toeplitz Operators, Levinson Equations, and Gohberg-Krein Factorization–A Simple and Unified Approach for the Rational Case, *J. Math. Anal. Appl.*, **87**, 295 (1982).

[997] S. Y. Kung and Y. H. Hu, A Highly Concurrent Algorithm and Pipelined Architecture for Solving Toeplitz Systems, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-31**, 66 (1983).

[998] H. Lev-Ari and T. Kailath, Lattice Filter Parametrization and Modeling of Nonstationary Processes, *IEEE Trans. Inform. Th.*, **IT-30**, 2 (1984).

[999] T. Kailath, Ed. *Modern Signal Processing*, Washington, DC, Hemisphere Publishing, 1985.

[1000] T. Kailath, Signal Processing in the VLSI Era, in S. Y. Kung, H. J. Whitehouse, and T. Kailath, Eds., *VLSI and Modern Signal Processing*, Englewood Cliffs, NJ, Prentice-Hall, 1985.

[1001] A. Yagle and B. C. Levy, The Schur Algorithm and Its Applications, *Acta Applic. Math.*, **3**, 255 (1985).

[1002] T. Kailath, A. M. Bruckstein, and D. Morgan, Fast Matrix Factorization via Discrete Transmission Lines, *Lin. Alg. Appl.*, **75**, 1 (1985).

[1003] P. P. Vaidyanathan and S. K. Mitra, Discrete Version of Richard's Theorem and Applications to Cascaded Lattice Realization of Digital Filter Transfer Functions, *IEEE Trans. Circ. Syst.*, **CAS-33**, 26 (1986).

[1004] J. Le Roux, Some Properties of the Schur Recursion for the Direct Computation of the Matricial Spectral Factor, *Signal Processing*, **11**, 359 (1986).

[1005] A. M. Bruckstein and T. Kailath, An Inverse Scattering Framework for Several Problems in Signal Processing, *ASSP Magazine*, no.1, 6 (1987).

[1006] P. Delsarte and Y. Genin, The Tridiagonal Approach to Inverse Scattering Problems, *Proc. 1987 IEEE Int. Conf. Circuits and Systems*, Philadelphia, PA, p.140.

[1007] H. Lev-Ari and T. Kailath, Lossless Cascade Networks: The Crossroads of Stochastic Estimation, Inverse Scattering, and Filter Synthesis, *Proc. 1987 IEEE Int. Conf. Circuits and Systems*, Philadelphia, PA, p.1088.

[1008] J. P. Burg, Maximum Entropy Spectral Analysis, Presented at *37th Annual Int. SEG Meeting*, Oklahoma City, (1967).

[1009] D. Childers, Ed., *Modem Spectrum Analysis*, New York, IEEE Press, 1978.

[1010] E. R. Kanasewich, *Time Sequence Analysis in Geophysics*, Edmonton, University of Alberta Press, 1975.

[1011] D. E. Smylie, G. K. C. Clarice, and T. J. Ulrich, Analysis of Irregularities in the Earth's Rotation, in *Methods of Computational Physics*, Vol.13, New York, Academic, 1973, p.391.

[1012] T. J. Ulrich and R. W. Clayton, Time Series Modelling and Maximum Entropy, *Phys. Earth Planet. Inter.*, **12**, 188 (1976).

[1013] M. Morf, B. Dickinson, T. Kailath, and A. Vieira, Efficient Solution of Covariance Equations for Linear Prediction, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-25**, 429 (1977).

[1014] E. T. Jaynes, On the Rationale of Maximum-Entropy Methods, *Proc. IEEE*, **70**, 939 (1982).

[1015] B. R. Frieden, Dice, Entropy, and Likelihood, *Proc. IEEE*, **73**, 1764 (1985).

[1016] B. Helme and C. L. Nikias, Improved Spectrum Performance via a Data-Adaptive Weighted Burg Technique, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-33**, 903 (1985).

[1017] P. F. Fougere, Applications of Maximum Entropy Spectrum Estimation to Air Force Problems, *Proc. Third ASSP Workshop on Spectrum Estimation and Modeling*, Boston, 1986, p.77.

[1018] J. Makhoul, Maximum Confusion Spectral Analysis, *Proc. Third ASSP Workshop on Spectrum Estimation and Modeling*, Boston, 1986, p.6.

[1019] B. S. Atal and S. Hanauer, Speech Analysis and Synthesis by Linear Prediction of the Speech Wave, *J. Acoust. Soc. Amer.*, **50**, 637 (1971).

[1020] F. Itakura and S. Saito, A Statistical Method for Estimation of Speech Spectral Density and Formant Frequencies, *Electr. Commun.*, **53-A**, 36 (1970).

[1021] R. Schafer and L. Rabiner, Digital Representation of Speech Signals, *Proc. IEEE*, **63**, 66 (1975).

[1022] L. R. Rabiner and R. W. Schafer, *Digital Processing of Speech Signals*, Englewood Cliffs, NJ, Prentice-Hall, 1978.

[1023] J. D. Markel and A. H. Gray, Jr. Roundoff Noise Characteristics of a Class of Orthogonal Polynomial Structures, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-23**, 473 (1975).

[1024] R. Viswanathan and J. Makhoul, Quantization Properties of Transmission Parameters in Linear Predictive Systems, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-23**, 309 (1975).

[1025] N. Morgan, *Talking Chips*, New York, McGraw-Hill, 1984.

[1026] M. R. Schroeder, Predictive Coding of Speech: Historical Review and Directions for Future Research, *Proc. 1986 IEEE Int. Conf. Acoust., Speech, Signal Process.*, Tokyo, p.261.

[1027] P. E. Papamichalis, *Practical Approaches to Speech Coding,* Englewood Cliffs, NJ, Prentice-Hall, 1987.

[1028] A. Isaksson, A. Wennberg, and L. H. Zetterberg, Computer Analysis of EEG Signals with Parametric Models, *Proc. IEEE*, **69**, 451 (1981).

[1029] W. Gersch, Spectral Analysis of EEG's by Autoregressive Decomposition of Time Series, *Math. Biosci.*, **7**, 205 (1970).

[1030] C. D. McGillem, J. I. Aunon, and D. G. Childers, Signal Processing In Evoked Potential Research: Applications of Filtering and Pattern Recognition, *CRC Critical Reviews of Bioengineering*, **6**, 225 (October 1981).

[1031] A. Isaksson and A. Wennberg, Spectral Properties of Nonstationary EEG Signals, Evaluated by Means of Kalman Filtering: Application Examples from a Vigilance Test, in P. Kellaway and I. Petersen, Eds., *Quantitative Analysis Studies in Epilepsy*, New York, Raven Press, 1976.

[1032] G. Bodenstein and H. M. Praetorius, Feature Extraction from the Electroencephalogram by Adaptive Segmentation, *Proc. IEEE*, **65**, 642 (1977).

[1033] T. Bohlin, Analysis of EEG Signals with Changing Spectra using a Short-Word Kalman Estimator, *Math. Biosci.*, **35**, 221 (1977).

[1034] F. H. Lopes da Silva, Analysis of EEG Nonstationarities, in W. A. Cobb and H. Van Duijn, Eds., *Contemporary Clinical Neurophysiology* (EEG Suppl. No. 34), Amsterdam, Elsevier, 1978.

[1035] Z. Rogowski, I. Gath, and E. Bental, On the Prediction of Epileptic Seizures, *Biol. Cybernetics*, **42**, 9 (1981).

[1036] F. Itakura, Minimum Prediction Residual Principle Applied to Speech Recognition, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-23**, 67 (1975).

[1037] J. M. Tribolet, L. R. Rabiner, and M. M. Sondhi, Statistical Properties of an LPC Distance Measure, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-27**, 550 (1979).

[1038] P. de Souza and P. J. Thompson, LPC Distance Measures and Statistical Tests with Particular Reference to the Likelihood Ratio, *IEEE Trans. Acoust., Speech. Signal Process.*, **ASSP-30**, 304 (1982).

[1039] R. M. Gray, et al., Distortion Measures for Speech Processing, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-28**, 367 (1980).

[1040] J. L. Flanagan, Talking with Computers: Synthesis and Recognition of Speech by Machines, *IEEE Trans. Biomed. Eng.*, **BME-29**, 223 (1982).

[1041] L. Dusek. T. B. Schalk, and M. McMahan, Voice Recognition Joins Speech on Programmable Board, *Electronics* **56** (8), 128 (April 1983).

[1042] H. Wakita, Direct Estimation of the Vocal Tract Shape by Inverse Filtering of Acoustic Speech Waveforms, *IEEE Trans. Audio Electroacoust.*, **AU-21**, 417 (1973).

[1043] J. A. Ware and K. Aki, Continuous and Discrete Inverse Scattering Problems in a Stratified Elastic Medium. I. Plane Waves at Normal Incidence, *J. Acoust. Soc. Am.*, **45**, 91 (1969).

[1044] L. C. Wood and S. Treitel, Seismic Signal Processing, *Proc. IEEE*, **63**, 649 (1975).

[1045] P. L. Goupillaud, An Approach to Inverse Filtering of Near-Surface Layer Effects from Seismic Records, *Geophysics*, **26**, 754 (1961).

[1046] J. F. Claerbout, Synthesis of a Layered Medium from Its Acoustic Transmission Response, *Geophysics*, **33**, 264 (1968).

[1047] F. Koehler and M. T. Taner, Direct and Inverse Problems Relating Reflection Coefficients and Reflection Response for Horizontally Layered Media, *Geophysics*, **42**, 1199 (1977).

[1048] E. A. Robinson and S. Treitel, The Fine Structure of the Normal Incidence Synthetic Seismogram, *Geophys. J . R. Astron. Soc.*, **53**, 289 (1978).

[1049] S. Treitel and E. A. Robinson, Maximum Entropy Spectral Decomposition of a Seismogram into Its Minimum Entropy Component Plus Noise, *Geophysics*, **46**, 1108 (1981).

[1050] J. M. Mendel and F. Habibi-Ashrafi, A Survey of Approaches to Solving Inverse Problems for Lossless Layered Media Systems, *IEEE Trans. Geosci. Electron.*, **GE-18**, 320 (1980).

[1051] K. P. Bube and R. Burridge, The One-Dimensional Problem of Reflection Seismology, *SIAM Rev.*, **25**, 497 (1983).

[1052] S. H. Gray, The Relationship Between "Direct, Discrete" and "Iterative, Continuous" One-Dimensional Inverse Methods, *Geophysics*, **49**, 54 (1984).

[1053] A. M. Bruckstein, B. C. Levy, and T. Kailath, Differential Methods for Inverse Scattering, *SIAM J. Appl. Math.*, **45**, 312 (1985).

[1054] R. G. Newton, Inversion of Reflection Data for Layered Media: A Review of Exact Methods, *Geophys. J. R. Astron. Soc.*, **65**, 191 (1981).

[1055] E. A. Robinson, A Spectral Approach to Geophysical Inversion by Lorentz, Fourier, and Radon Transforms, *Proc. IEEE*, **70**, 1039 (1982).

[1056] J. G. Berryman and R. R. Greene, Discrete Inverse Methods for Elastic Waves in Layered Media, *Geophysics*, **45**, 213 (1980).

[1057] F. J. Dyson, Old and New Approaches to the Inverse Scattering Problem, in E. H. Lieb, B. Simon, and A. S. Wightman, Eds., *Studies in Mathematical Physics*, Princeton, Princeton University Press, 1976.

[1058] K. M. Case, Inverse Scattering, Orthogonal Polynomials, and Linear Estimation, in I. C. Gohberg and M. Kac, Eds., *Topics in Functional Analysis, Advances in Mathematics Supplementary Studies*, Vol.3, New York, Academic, 1978.

[1059] M. T. Silvia and E. A. Robinson, *Deconvolution of Geophysical Time Series in the Exploration for Oil and Natural Gas*, Amsterdam, Elsevier, 1979.

[1060] S. Twomey, *Introduction to the Mathematics of Inversion in Remote Sensing and Indirect Measurements*, Amsterdam, Elsevier, 1977.

[1061] B. R. Frieden, "Image Enhancement and Restoration," in T. S. Huang, Ed., *Picture Processing and Digital Filtering*, New York, Springer-Verlag, 1975.

[1062] S. Treitel and L. R. Lines, "Linear Inverse Theory and Deconvolution," *Geophysics*, **47**, 115 (1982).

[1063] J. F. Claerbout and F. Muir, "Robust Modeling with Erratic Data," *Geophysics*, **38**, 826 (1973).

[1064] H. L. Taylor, S. C. Banks, and J. F. McCoy, "Deconvolution with the L_1 Norm," *Geophysics*, **44**, 39 (1979).

[1065] D. W. Oldenburg, "A Comprehensive Solution to the Linear Deconvolution Problem," *Geophys. J. R. Astron. Soc.*, **65**, 331 (1981).

[1066] S. Levy and P. K. Fullagar, "Reconstruction of a sparse spike train from a portion of its spectrum and application to high-resolution deconvolution," *Geophysics*, **46**, 1235 (1981).

[1067] D. W. Oldenburg, S. Scheuer, and S. Levy "Recovery of the acoustic impedance from reflection seismograms," *Geophysics*, **48**, 1318 (1983).

[1068] F. Santosa and W. W. Symes, W. W. "Linear inversion of band-limited reflection seismograms," *SIAM J. Sci. Statist. Comput.*, **7**, 1307 (1986).

[1069] R. Mammone and G. Eichmann, "Superresolving Image Restoration Using Linear Programming," *Applied Optics*, **21**, 496 (1982).

[1070] R. Mammone and G. Eichmann, "Restoration of Discrete Fourier Spectra Using Linear Programming,"*J. Optical Soc. Am.*, **72**, 987 (1982).

[1071] I. Barrodale and F. D. K. Roberts, "An Improved Algorithm for the Discrete L_1 Linear Approximation," *SIAM J. Numer. Anal.*, **10**, 839 (1973).

[1072] I. Barrodale and F. D. K. Roberts, "Algorithm 478: Solution of an Overdetermined System of Equations in the L_1 Norm," *Commun. ACM*, **17**, 319 (1974).

[1073] B. Drachman, "Two Methods to Deconvolve: L_1-Method Using Simplex Algorithm and L_2-Method Using Least Squares and a Parameter," *IEEE Trans. Antenn. Propag.*, **AP-32**, 219 (1984).

[1074] R. W. Schafer, R. M. Mersereau. and M. A. Richards, "Constrained Iterative Restoration Algorithms," *Proc. IEEE*, **69**, 432 (1981).

Spectrum Estimation and Array Processing

[1075] O. L. Frost, Power Spectrum Estimation, in G. Tacconi, Ed., *Aspects of Signal Processing,* Boston. Reidel. 1977.

[1076] P. R. Gutowski, E. A. Robinson, and S. Treitel, Spectral Estimation: Fact or Fiction?, *IEEE Trans. Geosci. Electron.*, **GE-16**, 80 (1978).

[1077] *Proc. IEEE,* **70** (9) (September 1982), Special Issue on Spectral Estimation.

[1078] A. Papoulis, Maximum Entropy and Spectral Estimation: A Review, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-29**, 1176 (1981).

[1079] E. A. Robinson, A Historical Perspective of Spectrum Estimation, *Proc. IEEE,* **70**, 885 (1982).

[1080] S. B. Kesler, Ed., *Modern Spectrum Analysis II,* New York, IEEE Press, 1986.

[1081] J. Capon, High Resolution Frequency Wavenumber Spectrum Analysis, *Proc. IEEE,* **57**, 1408 (1969).

[1082] J. Capon, Maximum Likelihood Spectral Estimation, in S. Haykin. Ed., *Nonlinear Methods of Spectral Analysis,* New York, Springer-Verlag. 1979.

[1083] R. T. Lacoss, Data Adaptive Spectral Analysis Methods, *Geophysics,* **36**, 661 (1971).

[1084] V. F. Pisarenko, The Retrieval of Harmonics from a Covariance Function, *Geoph. J . R. Astron. Soc.*, **33**, 347 (1973).

[1085] E. H. Satorius and J. R. Zeidler, Maximum Entropy Spectral Analysis of Multiple Sinusoids in Noise, *Geophysics,* **43**, 1111 (1978).

[1086] D. W. Tufts and R. Kumaresan, Singular Value Decomposition and Improved Frequency Estimation Using Linear Prediction, *IEEE Trans. Acoust., Speech. Signal Process.*, **ASSP-30**, 671 (1982).

[1087] D. W. Tufts and R. Kumaresan, Estimation of Frequencies of Multiple Sinusoids: Making Linear Prediction Perform like Maximum Likelihood, *Proc. IEEE,* **70**, 975 (1982).

[1088] S. L. Marple, Frequency Resolution of Fourier and Maximum Entropy Spectral Estimates, *Geophysics,* **47**, 1303 (1982).

[1089] M. Quirk and B. Liu, On the Resolution of Autoregressive Spectral Estimation, *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 1095 (1983).

[1090] S. Y. Kung and Y. H. Hu, Improved Pisarenko's Sinusoidal Spectrum Estimate via SVD Subspace Approximation Methods, *Proc. 21st IEEE Int. Conf. Decision and Control,* Orlando, FL, (1982), p. 1312.

[1091] Y. H. Hu and S. Y. Kung, Toeplitz Eigensystem Solver, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-33**, 1264 (1985).

[1092] B. D. Steinberg, *Principles of Aperture and Array System Design,* New York, Wiley, 1976.

[1093] J. E. Hudson, *Adaptive Array Principles,* Stevenage, UK, Peter Peregrinus, 1981.

[1094] D. E. N. Davies, K. G. Corless, D. S. Hicks, and K. Milne, Array Signal Processing, in A. W. Rudge, K. Milne, A. D. Olver, and P. Knight, Eds., *The Handbook of Antenna Design,* vol. 2, London, Peter Peregrinus, 1983.

[1095] N. L. Owsley, Sonar Array Processing, in S. Haykin, Ed., *Array Signal Processing,* Englewood Cliffs, NJ, Prentice-Hall, 1985.

[1096] S. Haykin, Radar Signal Processing, *ASSP Magazine,* **2**, no.2, 2 (1985).

[1097] B. L. Lewis, F. F. Kretschmer, and W. W. Shelton, Eds., *Aspects of Radar Signal Processing,* Norwood, MA, Artech House, 1986.

[1098] W. C. Knight, R. G. Pridham, and S. M. Kay, Digital Signal Processing for Sonar, *Proc. IEEE,* **69**, 1451 (1981).

[1099] W. F. Gabriel, Spectral Analysis and Adaptive Array Superresolution Techniques, *Proc. IEEE,* **68**, 654 (1980).

[1100] R. N. McDonough, Application of the Maximum Likelihood Method and the Maximum Entropy Method to Array Processing, in S. Haykin, Ed., *Nonlinear Methods of Spectral Analysis,* New York, Springer-Verlag, 1979.

[1101] D. H. Johnson, The Application of Spectral Estimation Methods to Bearing Estimation Problems, *Proc. IEEE,* **70**, 1018 (1982).

[1102] A. J. Berni, Angle-of-Arrival Estimation Using an Adaptive Antenna Array, *IEEE Trans. Aerosp. Electron. Syst.*, **AES-11**, 278 (1975).

[1103] T. Thorvaldsen, Maximum Entropy Spectral Analysis in Antenna Spatial Filtering, *IEEE Trans. Antennas Propag.*, **AP-28**, 552 (1980).

[1104] T. E. Barnard, Two Maximum Entropy Beamforming Algorithms for Equally Spaced Line Arrays, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-30**, 175 (1980).

[1105] N. L. Owsley, Spectral Signal Set Extraction, in G. Tacconi, Ed., *Aspects of Signal Processing,* Boston, D. Reidel, 1977.

[1106] J. E. Evans, Aperture Sampling Techniques for Precision Direction Finding, *IEEE Trans. Aerosp. Electron. Syst.*, **AES-15**, 899 (1979).

[1107] W. D. White, Angular Spectra in Radar Applications, *IEEE Trans. Aerosp. Electron. Syst.*, **AES-15**, 895 (1979).

[1108] J. E. Evans, Comments on "Angular Spectra in Radar Applications" *IEEE Trans. Aerosp. Electron. Syst.*, **AES-15**, 891 (1979).

[1109] W. S. Ligget, Passive Sonar: Fitting Models to Multiple Time Series, in J. W. R. Griffiths, et al., Eds., *Signal Processing,* New York, Academic, 1973.

[1110] R. O. Schmidt, Multiple Emitter Location and Signal Parameter Estimation, *Proc. 1979 RADC Spectral Estimation Workshop,* Rome, NY, p. 243. Reprinted in the Special Issue on Adaptive Processing Antenna Systems, *IEEE Trans. Antennas Propag.*, **AP-34**, 276 (1986).

[1111] S. S. Reddi, Multiple Source Location–A Digital Approach, *IEEE Trans. Aerosp. Electron. Syst.*, **AES-15**, 95 (1979).

[1112] G. Bienvenu and L. Kopp, Adaptivity to Background Noise Spatial Coherence for High Resolution Passive Methods, *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 307–310 (1980).

[1113] A Cantoni and L. Godara, Resolving the Directions of Sources in a Correlated Field Incident on an Array, *J. Acoust. Soc. Am.*, **67**, 1247 (1980).

[1114] D. Bordelon, Complementarity of the Reddi Method of Source Direction Estimation with those of Pisarenko and Cantoni and Godara, I, *J. Acoust., Soc. Am.*, **69**, 1355 (1981).

[1115] T. S. Durrani and K. C. Sharman, Extraction of an Eigenvector-Oriented "Spectrum" for the MESA Coefficients, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-30**, 649 (1982).

[1116] T. P. Bronez and J. A. Cadzow, An Algebraic Approach to Superresolution Adaptive Array Processing, *IEEE Trans. Aerosp. Electron. Syst.*, **AES-19**, 123 (1983).

[1117] R. Kumaresan and D. W. Tufts, Estimating the Angles of Arrival of Multiple Plane Waves, *IEEE Trans. Aerosp. Electron. Syst.*, **AES-19**, 134 (1983).

[1118] D. H. Johnson and S. R. DeGraaf, Improving the Resolution of Bearing in Passive Sonar Arrays by Eigenvalue Analysis, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-30**, 638 (1982).

[1119] T. E. Evans, et al., High Resolution Angular Spectrum Estimation Techniques for Terrain Scattering Analysis and Angle of Arrival Estimation, *Proc. First ASSP Spectral Estimation Workshop,* Hamilton, Ontario, (198l), p. 134.

[1120] K. C. Sharman and T. S. Durrani, Eigenfilter Approaches to Adaptive Array Processing, *Proc. IEE,* part F, **130**, 22 (1983).

[1121] M. Wax and T. Kailath, Optimum Localization of Multiple Sources by Passive Arrays, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-31**, 1210 (1983).

[1122] G. Bienvenu and L. Kopp, Optimality of High Resolution Array Processing Using the Eigensystem Approach, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-31**, 1235 (1983).

[1123] G. Bienvenu and H. Mermoz, Principles of High-Resolution Array Processing, in S. Y. Kung, H. J. Whitehouse, and T. Kailath, Eds., *VLSI and Modern Signal Processing,* Englewood Cliffs, NJ, Prentice-Hall, 1985.

[1124] N. L. Owsley, High-Resolution Spectrum Analysis by Dominant-Mode Enhancement, *Ibid.*

[1125] M. Wax and T. Kailath, Detection of Signals by Information Theoretic Criteria, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-33**, 387 (1985).

[1126] T. J. Shan, M. Wax, and T. Kailath, On Spatial Smoothing for Direction-of-Arrival Estimation of Coherent Signals, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-33**, 806 (1985).

[1127] A. Di, Multiple Source Location–A Matrix Decomposition Approach, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-33**, 1086 (1985).

[1128] S. R. De Graaf and D. H. Johnson, Capability of Array Processing Algorithms to Estimate Source Bearings, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-33**, 1368 (1985).

[1129] W. F. Gabriel, Using Spectral Estimation Techniques in Adaptive Processing Antenna Systems, *IEEE Trans. Antennas Propag.*, **AP-34**, 291 (1986).

[1130] I. Karasalo, Estimating the Covariance Matrix by Signal Subspace Averaging, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-34**, 8 (1986).

[1131] G. Vezzosi, Estimation of Phase Angles from the Cross-Spectral Matrix, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-34**, 405 (1986).

[1132] G. Su and M. Morf, Modal Decomposition Signal Subspace Algorithms, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-34**, 585 (1986).

[1133] K. C. Sharman and T. S. Durrani, A Comparative Study of Modern Eigenstructure Methods for Bearing Estimation–A New High Performance Approach, *Proc. 1986 IEEE Int. Conf. Decision and Control,* Athens, p. 1737.

[1134] U. Nickel, Angular Superresolution with Phased Array Radar: A Review of Algorithms and Operational Constraints, *IEE Proc.*, **134**, Pt. F, 53 (1987).

[1135] A. Paulraj and T. Kailath, Eigenstructure Methods for Direction of Arrival Estimation in the Presence of Unknown Noise Fields, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-34**, 13 (1986).

[1136] F. B. Tuteur and Y. Rockah, A New Method for Signal Detection and Estimation Using the Eigenstructure of the Covariance Difference, *Proc. 1986 IEEE Int. Conf. Acoust., Speech, Signal Process.*, Tokyo, p. 2811.

[1137] F. B. Tuteur and Y. Rockah, The Covariance Difference Method in Signal Detection, *Proc. Third ASSP Workshop on Spectrum Estimation and Modeling, Boston,* 1986, p. 120.

[1138] S. Prasad, R. Williams, A. Mahalanabis, and L. Sibul, A Transform Based Covariance Differencing Approach to Bearing Estimation, *Proc. 1987 IEEE Int. Conf. Acoust., Speech, Signal Process.*, Dallas, p. 1119.

[1139] S. J. Orfanidis, A Reduced MUSIC Algorithm, *Proc. Third ASSP Workshop on Spectrum Estimation and Modeling,* Boston, 1986, p. 165.

[1140] M. Wax and T. Kailath, Extending the Threshold of the Eigenstructure Methods, *Proc. 1985 IEEE Int. Conf. Acoust., Speech, Signal Process.*, Tampa, FL, p. 556.

[1141] R. Kumaresan and A. K. Shaw, High Resolution Bearing Estimation Without Eigendecomposition, *Proc. 1985 IEEE Int. Conf. Acoust., Speech, Signal Process.*, Tampa, FL, p. 576.

[1142] Y. Bresler and A. Macovski, Exact Maximum Likelihood Parameter Estimation of Superimposed Exponential Signals in Noise, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-34**, 1081 (1986).

[1143] Y. Bresler and A. Macovski, On the Number of Signals Resolvable by a Uniform Linear Array, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-34**, 1361 (1986).

[1144] R. Roy, A. Paulraj, and T. Kailath, Estimation of Signal Parameters via Rotational Invariance Techniques–ESPRIT, *Proc. 19th Asilomar Conf. Circ., Syst. and Computers,* Asilomar, CA, 1985, p. 83.

[1145] R. Roy, A. Paulraj, and T. Kailath, ESPRIT- A Subspace Rotation Approach to Estimation of Parameters of Cisoids in Noise, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-34**, 1340 (1986).

[1146] R. Roy, A. Paulraj, and T. Kailath, Comparative Performance of ESPRIT and MUSIC for Direction-of-Arrival Estimation, *Proc. 1987 IEEE Int. Conf. Acoust., Speech, Signal Process.*, Dallas, p. 2344.

[1147] F. Haber and M. Zoltowski, Spatial Spectrum Estimation in a Coherent Signal Environment Using an Array in Motion, *IEEE Trans. Antennas Propag.*, **AP-34**, 301 (1986).

[1148] A. J. Luthra, A Solution to the Adaptive Nulling Problem with a Look-Direction Constraint in the Presence of Coherent Jammers, *IEEE Trans. Antennas Propag.*, **AP-34**, 702 (1986).

[1149] S. Kesler, J. Kesler, and G. Levita, Experiments in Resolving Coherent Targets in the Near Field, *Proc. Third ASSP Workshop on Spectrum Estimation and Modeling,* Boston, 1986, p. 168.

[1150] S. S. Reddi, On a Spatial Smoothing Technique for Multiple Source Location, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-35**, 709 (1987), and *ibid.*, p. 1352.

[1151] J. A. Cadzow, Y. S. Kim, D. C. Shiue, Y. Sun, and G. Xu, Resolution of coherent Signals Using a Linear Array, *Proc. 1987 IEEE Int. Conf. Acoust., Speech, Signal Process.*, Dallas, p. 1597.

[1152] R. Williams, S. Prasad, A. Mahalanabis, and L. Sibul, Localization of Coherent Sources Using a Modified Spatial Smoothing Technique. *Proc. 1987 IEEE Int. Conf. Acoust., Speech, Signal Process.*, Dallas, p. 2352.

[1153] A. M. Bruckstein, T. J. Shan, and T. Kailath, The Resolution of Overlapping Echos, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-33**, 1357 (1985).

[1154] I. Isenberg and R. D. Dyson, The Analysis of Fluorescent Decay by a Method of Moments, *Biophys. J.*, **9**, 1337 (1969).

[1155] A. J. Evans and R. Fischl, Optimal Least-Squares Time-Domain Synthesis of Recursive Digital Filters, *IEEE Trans. Audio Electroacoust.*, **AU-21**, 61 (1973).

[1156] A. J. Berni, Target Identification by Natural Resonance Estimation, *IEEE Trans. Aerosp. Electron. Syst.*, **AES-11**, 147 (1975).

[1157] M. L. Van Blaricum and R. Mittra, Problems and Solutions Associated with Prony's Method for Processing Transient Data, *IEEE Trans. Antennas Propag.*, **AP-26**, 174 (1978).

[1158] T. L. Henderson, Geometric Methods for Determining System Poles from Transient Response, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-29**, 982 (1981).

[1159] R. Kumaresan and D. W. Tufts, Estimating the Parameters of Exponentially Damped Sinusoids and Pole-Zero Modeling in Noise, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-30**, 833 (1982).

[1160] M. Wax, R. O. Schmidt, and T. Kailath, Eigenstructure Method for Retrieving the Poles from the Natural Response, *Proc. 1983 IEEE Int. Conf. Decision and Control,* San Antonio, TX, p. 1343.

[1161] R. Kumaresan, L. L. Scharf, and A. K. Shaw, An Algorithm for Pole-Zero Modeling and Spectral Analysis, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-34**, 637 (1986).

[1162] J. A. Cadzow and M. M. Wu, Analysis of Transient Data in Noise, *IEE Proc.*, **134**, Pt. F, 69 (1987).

[1163] S. J. Orfanidis, Pole Retrieval by Eigenvector Methods, *Proc. 1987 IEEE Int. Conf. Acoust., Speech, Signal Process.*, Dallas, p. 1505.

[1164] B. N. Parlett, *The Symmetric Eigenvalue Problem,* Englewood Cliffs, NJ, Prentice-Hall, 1980.

[1165] G. H. Golub and V. Pereyra, The Differentiation of Pseudo-Inverses and Non-Linear Least-Squares Problems Whose Variables Separate, *SIAM J. Numer. Anal.*, **10**, 413 (1973).

[1166] G. H. Golub and C. F. Van Loan, *Matrix Computations,* Baltimore, Johns Hopkins University Press, 1983.

[1167] H. Cox, Resolving Power and Sensitivity to Mismatch of Optimum Array Processors, *J. Acoust. Soc. Am.*, **54**, 771 (1973).

[1168] F. Gabriel, Adaptive Arrays–An Introduction, *Proc. IEEE,* **64**, 239 (1976).

[1169] B. Widrow, et al., Adaptive Antenna Systems, *Proc. IEEE,* **55**, 2143 (1967).

[1170] C. L. Zham, Application of Adaptive Arrays to Suppress Strong Jammers in the Presence of Weak Signals, *IEEE Trans. Aerosp. Electron. Syst.*, **AES-9**, 260 (1973).

[1171] T. W. Anderson, *The Statistical Analysis of Time Series,* New York, Wiley, 1971.

[1172] D. N. Lawley and A. E. Maxwell, *Factor Analysis as a Statistical Method,* London, Butterworth, 1971.

[1173] C. R. Rao, *Linear Statistical Inference and Its Applications,* (2nd ed.), New York, Wiley, 1973.

[1174] D. R. Cox and D. V. Hinkley, *Theoretical Statistics,* London, Chapman and Hall, 1974.

[1175] D. R. Brillinger, *Time Series, Data Analysis and Theory,* New York, Holt, Rinehart and Winston, 1975.

[1176] M. G. Kendall and A. Stuart, *The Advanced Theory of Statistics,* vol. 2, (4th edition), London, Griffin, 1979.

[1177] M. G. Kendall and A. Stuart, *The Advanced Theory of Statistics,* vol. 3, (3d edition), New York, Hafner Press. 1976.

[1178] M. S. Srivastava and C. G. Khatri, *An Introduction to Multivariate Statistics,* New York, North Holland, 1979.

[1179] T. W. Anderson, *An Introduction to Multivariate Statistical Analysis,* (2nd ed.), New York, Wiley 1984.

[1180] J. Cryer, *Times Series Analysis,* Boston, Duxbury Press, 1986.

[1181] K. Dzhaparidze, *Parameter Estimation and Hypothesis Testing in Spectral Analysis of Stationary Time Series,* New York, Springer-Verlag, 1986.

[1182] P. J. Brockwell and R. A. Davis, *Time Series: Theory and Methods,* New York, Springer-Verlag, 1987.

[1183] H. B. Mann and A. Wald, On the Statistical Treatment of Linear Stochastic Difference Equations, *Econometrica,* **11**, 173 (1943).

[1184] P. Whittle, The Analysis of Multiple Stationary Time Series, *J. Roy. Stat. Soc.*, Ser. B, **15**, 125 (1953).

[1185] J. Capon and N. R. Goodman, Probability Distributions for Estimators of the Frequency-Wavenumber Spectrum, *Proc. IEEE,* **58**, 1785 (1971).

[1186] O . Barndorff-Nielsen and G. Schou, On the Parametrization of Autoregressive Models by Partial Autocorrelations, *J. Multiv. Anal.*, **3**, 408 (1973).

[1187] M. Pagano, Estimation of Models of Autoregressive Signal Plus White Noise, *Ann. Stat.*, **2**, 99 (1974).

[1188] K. N. Berk, Consistent Autoregressive Spectral Estimates, *Ann. Stat.*, **2**, 489 (1974).

[1189] A. B. Baggeroer, Confidence Intervals for Regression (MEM) Spectral Estimates, *IEEE Trans. Inform. Th.*, **IT-22**, 534 (1976).

[1190] H. Sakai, Statistical Properties of AR Spectral Analysis, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-27**, 402 (1979).

[1191] R. D. Martin, The Cramér-Rao Bound and Robust M-Estimates for Autoregressions, *Biometrika,* **69**, 437 (1982).

[1192] S. M. Kay and J. Makhoul, On the Statistics of the Estimated Reflection Coefficients of an Autoregressive Process, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-31**, 1447 (1983).

[1193] M. Aktar, B. Sankur, and Y. Istefanopulos, Properties of the Maximum Likelihood and Pisarenko Spectral Estimates, *Signal Processing,* **8**, 401 (1985).

[1194] B. Porat and B. Friedlander, Computation of the Exact Information Matrix of Gaussian Time Series with Stationary Random Components, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-34**, 118 (1986).

[1195] S. Kay and D. Sengupta, Spectral Estimation of Non-Gaussian Autoregressive Processes, *Proc. Third ASSP Workshop on Spectrum Estimation and Modeling,* Boston, 1986, p. 10.

[1196] D. Burshtein and E. Weinstein, Confidence Intervals for the Maximum Entropy Spectrum, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-35**, 504 (1987).

[1197] M. A. Girschick, On the Sampling Theory of Roots of Determinantal Equations, *Ann. Math. Stat.*, **10**, 203 (1939).

[1198] D. N. Lawley, Tests of Significance for the Latent Roots of Covariance and Correlation Matrices, *Biometrika,* **43**, 128 (1956).

[1199] T. W. Anderson, Asymptotic Theory for Principal Component Analysis, *Ann. Math. Stat.*, **34**, 122 (1963).

[1200] R. P. Gupta, Asymptotic Theory for Principal Component Analysis in the Complex Case, *J. Indian Stat. Assoc.*, **3**, 97 (1965).

[1201] D. E. Tyler, Asymptotic Inference for Eigenvectors, *Ann. Stat.*, **9**, 725 (1981).

[1202] H. Sakai, Statistical Analysis of Pisarenko's Method for Sinusoidal Frequency Estimation, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-32**, 95 (1984).

[1203] K. Shaman, T. S. Durrani, M. Wax, and T. Kailath, Asymptotic Performance of Eigenstructure Spectral Analysis Methods, *Proc. 1984 IEEE Int. Conf. Acoust., Speech, Signal Process.*, San Diego, CA, p. 455.

[1204] D. J. Jeffries and D. R. Farrier, Asymptotic Results for Eigenvector Methods, *IEE Proc.*, **132**, Pt. F, 589 (1985).

[1205] M. Kaveh and A. J. Barabell, The Statistical Performance of the MUSIC and the Minimum-Norm Algorithms for Resolving Plane Waves in Noise, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-34**, 331 (1986).

[1206] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes,* New York, Springer-Verlag, 1986.

[1207] N. R. Goodman, Statistical Analysis Based on a Certain Multivariate Complex Gaussian Distribution, *Ann. Math. Stat.*, **34**, 152 (1963).

[1208] K. S. Miller, *Complex Stochastic Processes,* Reading, MA, Addison-Wesley, 1974.

LCMV and GSC Beamforming

[1209] O. L. Frost, “An algorithm for linearly constrained adaptive array processing,” *Proc. IEEE*, **60**, 926 (1972).

[1210] S. Applebaum and D. Chapman, “Adaptive arrays with main beam constraints,” *IEEE Trans. Antennas Propagat.*, **AP-24**, 650 (1976).

[1211] C. W. Jim, “A comparison of two LMS constrained optimal structures,” *Proc. IEEE*, **65**, 1730 (1977).

[1212] L. J. Griffiths and C. W. Jim, “An alternative approach to linearly constrained adaptive beamforming,” *IEEE Trans. Antennas Propagat.*, **AP-20**, 27 (1982).

[1213] L. J. Griffiths and K. M. Buckley, “Quiescent pattern control in linearly constrained adaptive arrays,” *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-35**, 917 (1987).

[1214] C-Y. Tseng and L. J. Griffiths, “A systematic procedure for implementing the blocking matrix in decomposed form,” *Proc. 22nd Asilomar Conf. Signals Systems and Computers*, vol. 2, pp. 808, 1988.

[1215] B. D. Van Veen and K. M. Buckley, “Beamforming: A versatile approach to spatial filtering,” *IEEE Acoust. Speech Signal Processing Mag.*, **5**, no.2, 4 (1988).

[1216] H. Krim and M. Viberg, “Two decades of array signal processing research: the parametric approach,” *IEEE Signal Proc. Mag.* **13**, no.4, 67 (1996).

[1217] L. C. Godara, “Applications of Antenna Arrays to Mobile Communications, Part I,” *Proc. IEEE*, **85**, 1031 (1997), and “Part II,” *ibid.*, p.1195.

[1218] J. A. Apolinario, M. L. R. de Campos, and C. P. Bernal,“The constrained conjugate-gradient algorithm,” *IEEE Signal Proc. Lett.*, **7**, 351 (2000).

[1219] B. R. Breed and J. Strauss, “A short proof of the equivalence of LCMV and GSC beamforming,” *IEEE Signal Proc. Lett.*, **9**, 168 (2002).

[1220] S. Werner, J. A. Apolinario, and M. L. R. de Campos, “On the Equivalence of RLS Implementations of LCMV and GSC Processors,” *IEEE Signal Proc. Lett.*, **10**, 356 (2003).

[1221] L. S. Resende, J. M. T. Romano, and M. G. Bellanger, “A fast least-squares algorithm for linearly constrained adaptive filtering,” *IEEE Trans. Signal Process.*, **44**, 1168 (1996).

Markowitz Portfolios

[1222] H. Markowitz, “Portfolio Selection,” *J. Finance*, **7**, 77 (1962).

[1223] W. F. Sharpe, “Capital asset prices: A theory of market equilibrium under conditions of risk,” *J. Finance*, **19**, 425 (1964).

[1224] H. Markowitz, *Mean-Variance Analysis in Portfolio Choice and Capital Markets*, Wiley (2000).

[1225] R. Merton, “An analytic derivation of the efficient portfolio frontier,” *J. Financial Quant. Anal.* **7**, 1851 (1972).

[1226] H. M. Markowitz, “Foundations of Portfolio Theory,” *J. Finance*, **46**, 469 (1991).

[1227] W. F. Sharpe, “Capital Asset Prices with and without Negative Holdings,” *J. Finance*, **46**, 489 (1991).

[1228] H. M. Markowitz, “The General Mean-Variance Portfolio Selection Problem [and Discussion],” *Phil. Trans.: Phys. Sci. Eng.*, **347**, 543 (1994).

[1229] H. M. Markowitz, “The Early History of Portfolio Theory: 1600-1960,” *Financial Analysts J.*, **55**, no.4, p.5, 1999.

[1230] K. V. Fernando, “Practical Portfolio Optimization,” Numerical Algorithms Group, Tech. Report, `https://www.nag.co.uk/doc/techrep/Pdf/tr2_00.pdf`

[1231] P. A. Forsyth, “An Introduction to Computational Finance Without Agonizing Pain,” 2007, available online from, `https://cs.uwaterloo.ca/~paforsyt/agon.pdf`

[1232] H. Ahmadi and D. Sitdhirasdr, “Portfolio Optimization is One Multiplication, the Rest is Arithmetic,” *J. Appl. Fin. & Banking*, **6** 81 (2016); `http://www.scienpress.com/download.asp?ID=1729`

[1233] J. B. Heaton, N. G. Polson, and J. H.Witte, “Deep learning for finance: deep portfolios,” *Appl. Stoch. Models Bus. Ind.*, **33**, 3 (2017); with discussions, *ibid.*,, p.13, and p.16, and rejoinder, p.19.

SVD – Books

[1234] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3/e, Johns Hopkins University Press, Baltimore, 1996.

[1235] D. S. Watkins, *Fundamentals of Matrix Computations*, 2/e, Wiley, New York, 2002.

[1236] A. Björck, *Numerical Methods for Least Squares Problems*, SIAM Press, Philadelphia, 1996.

[1237] T. W. Anderson, *Introduction to Multivariate Statistical Analysis*, 2/e, Wiley, New York, 1984.

[1238] D. F. Morrison, *Multivariate Statistical Methods*, 3/e, McGraw-Hill, New York, 1990.

[1239] R. W. Preisendorfer, *Principal Component Analysis in Meteorology and Oceanography*, Elsevier, Amsterdam, 1988.

[1240] D. C. Kahaner, C. Moler, and S. Nash, *Numerical Methods and Software*, Prentice Hall, Englewood Cliffs, NJ, 1989.

[1241] D. S. Wilks, *Statistical Methods in the Atmospheric Sciences*, Academic Press, New York, 1995.

[1242] H. von Storch and F. W. Zwiers, *Statistical Analysis in Climate Research*, Cambridge Univ. Press, Cambridge, 1999.

[1243] I. T. Jollife, *Principal Component Analysis*, 2/e, Springer-Verlag, New York, 2002.

[1244] K. I. Diamantaras and S. Y. Kung, *Principal Component Neural Networks*, Wiley, New York, 1996.

[1245] R. Gittins, *Canonical Analysis*, Springer-Verlag, New York, 1985.

[1246] B. Parlett, *Symmetric Eigenvalue Problem*, Prentice Hall, Upper Saddle River, NJ, 1980.

[1247] E. F. Deprettere, ed., *SVD and Signal Processing*, North-Holland, New York, 1988.

[1248] R. J. Vaccaro, ed., *SVD and Signal Processing II*, Elsevier, New York, 1991.

[1249] M. Moonen and B. de Moor, *SVD and Signal Processing III*, Elsevier, New York, 1995.

[1250] S. Van Huffel and J. Vandewalle, *The Total Least Squares Problem*, SIAM, Philadelphia, 1991.

[1251] H. D. I. Abarbanel, *Analysis of Observed Chaotic Data*, Springer-Verlag, New York, 1996.

[1252] H. Kantz and T. Schreiber, *Nonlinear Time Series Analysis*, Cambridge Univ. Press, Cambridge, 1997.

[1253] A. S. Weigend and N. A. Gershenfeld, eds., *Time Series Prediction: Forecasting the Future and Understanding the Past* Addison-Wesley, Reading, MA, 1994. The time-series data and most of the chapters are available on the web via FTP from: `ftp://ftp.santafe.edu/pub/Time-Series/`.

SVD – Applications

[1254] G. Strang, “The Fundamental Theorem of Linear Algebra,” *Am. Math. Monthly*, **100**, 848 (1993).

[1255] D. Kalman, “A Singularly Valuable Decomposition: The SVD of a Matrix,” *College Math. J.*, **27**, 2 (1996).

[1256] C. Mulcahy and J. Rossi, “A Fresh Approach to the Singular Value Decomposition,” *Coll. Math. J.*, **29**, 199 (1998).

[1257] C. Long, “Visualization of Matrix Singular Value Decomposition,” *Math. Mag.*, **56**, 161 (1983).

[1258] V. C. Klema and A. J. Laub, “The Singular Value Decomposition: Its Computation and Some Applications,” *IEEE Trans. Aut. Contr.*, **AC-25**, 164 (1980).

[1259] E. Biglieri and K. Yao, “Some Properties of Singular Value Decomposition and Their Applications to Digital Signal Processing,” *Sig. Process.*, **18**, 277 (1989).

[1260] A. van der Veen, E. F. Deprettere, and A. L. Swindlehurst, “Subspace Based Signal Analysis Using Singular Value Decomposition,” *Proc. IEEE*, **81**, 1277 (1993).

[1261] J. Mandel, “Use of the Singular Value Decomposition in Regression Analysis,” *Amer. Statistician*, **36**, 15 (1982).

[1262] I. J. Good, “Some Applications of the Singular Decomposition of a Matrix,” *Technometrics*, **11**, 823 (1969).

[1263] D. D. Jackson, “Interpretation of Inaccurate, Insufficient and Inconsistent Data,” *Geophys. J. Roy. Astron. Soc.*, **28**, 97 (1972).

[1264] D. W. Tufts, R. Kumaresan, and I. Kirsteins, “Data Adaptive Signal Estimation by Singular Value Decomposition of a Data Matrix,” *Proc. IEEE*, **70**, 684 (1982).

[1265] D. W. Tufts and R. Kumaresan, “Estimation of Frequencies of Multiple Sinusoids: Making Linear Prediction Perform Like Maximum Likelihood,” *Proc. IEEE*, **70**, 975 (1982).

[1266] D. W. Tufts and R. Kumaresan, “Singular Value Decomposition and Improved Frequency Estimation Using Linear Prediction,” *IEEE Trans. Acoust., Speech, Sig. Process.*, **ASSP-30**, 671 (1982).

[1267] R. Kumaresan and D. W. Tufts, “Estimating the Parameters of Exponentially Damped Sinusoids and Pole-Zero Modeling in Noise,” *IEEE Trans. Acoust., Speech, Sig. Process.*, **ASSP-30**, 833 (1982).

[1268] J. A. Cadzow, “Signal Enhancement—Composite Property Mapping Algorithm,” *IEEE Trans. Acoust., Speech, Sig. Process.*, **ASSP-36**, 49 (1988).

[1269] L. L. Scharf, “The SVD and Reduced Rank Signal Processing,” *Sig. Process.*, **25**, 113 (1991).

[1270] J. A. Cadzow and D. M. Wilkes, “Enhanced Rational Signal Modeling,” *Sig. Process.*, **25**, 171 (1991).

[1271] B. De Moor, “The Singular Value Decomposition and Long and Short Spaces of Noisy Matrices,” *IEEE Trans. Sig. Process.*, **SP-41**, 2826 (1993).

[1272] H. Yang and M. A. Ingram, “Design of Partially Adaptive Arrays Using the Singular-Value Decomposition,” *IEEE Trans. Antennas Propagat.*, **AP-45**, 843 (1997).

[1273] S. Y. Kung, K. S. Arun, and D. V. B. Rao, “State Space and Singular Value Decomposition Based Approximation Methods for the Harmonic Retrieval Problem, *J. Opt. Soc. Am.*, **73**, 1799 (1983).

[1274] H. Barkhuijsen, R. De Beer, W. Bovée, and D. Van Ormon, “Retrieval of Frequencies, Amplitudes, Damping Factors, and Phases from Time-Domain Signals Using a Linear Least-Squares Process,” *J. Magn. Reson.*, **61**, 465 (1985).

[1275] J. E. Hudson, “Decomposition of Antenna Signals into Plane Waves,” *IEE Proc.*, pt. H, **132**, 53 (1985).

[1276] R. Roy, A. Paulraj, and T. Kailath, “ESPRIT–A Subspace Rotation Approach to Estimation of Parameters of Cisoids in Noise,” *IEEE Trans. Acoust., Speech, Sig. Process.*, **ASSP-34**, 1340 (1986).

[1277] A. J. Mackay and A. McCowen, “An Improved Pencil-of-Functions Method and Comparisons with Traditional Methods of Pole Extraction,” *IEEE Trans. Antennas Propagat.*, **AP-35**, 435 (1987).

[1278] P. De Groen and B. De Moor, “The Fit of a Sum of Exponentials to Noisy Data,” *J. Comp. Appl. Math.*, **20**, 175 (1987).

[1279] Y. Hua and T. K. Sarkar, “Generalized Pencil-of-Function Method for Extracting Poles of an EM System from Its Transient Response,” *IEEE Trans. Antennas Propagat.*, **AP-37**, 229 (1989).

[1280] Y. Hua and T. K. Sarkar, “Matrix Pencil Method for Estimating Parameters of Exponentially Damped/Undamped Sinusoids in Noise,” *IEEE Trans. Acoust., Speech, Sig. Process.*, **ASSP-38**, 814 (1990).

[1281] Y. Hua and T. K. Sarkar, “On SVD for Estimating Generalized Eigenvalues of Singular Matrix Pencil in Noise,” *IEEE Trans. Sig. Process.*, **SP-39**, 892 (1991).

[1282] T. K. Sarkar and O. Pereira, “Using the Matrix Pencil Method to Estimate the Parameters of a Sum of Complex Exponentials,” *IEEE Ant. Propagat. Mag.*, **37**, no.1, 48 (1995).

[1283] Y. Y. Lin, P. Hodgkinson, M. Ernst, and A. Pines, “A Novel Detection-Estimation Scheme for Noisy NMR Signals: Applications to Delayed Acquisition Data,” *J. Magn. Reson.*, **128**, 30 (1997).

[1284] A. Driouach, A. Rubio Bretones, and R. Gómez Martin, “Application of Parametric Problems to Inverse Scattering Problems,” *IEE Proc.-Microw., Antenn., Propag.*, **143**, 31 (1996).

[1285] C. C. Chen and L. Peters, “Buried Unexploded Ordnance Identification via Complex Natural Resonances,” *IEEE Trans. Antennas Propagat.*, **AP-45**, 1645 (1997).

[1286] E. M. Dowling, R. D. DeGroat, and D. A. Linebarger, “Exponential Parameter Estimation in the Presence of Known Components and Noise,” *IEEE Trans. Antennas Propagat.*, **AP-42**, 590 (1994).

[1287] S. Van Huffel, “Enhanced Resolution Based on Minimum Variance Estimation and Exponential Data Modeling,” *Sig. Process.*, **33**, 333 (1993).

[1288] S. Van Huffel and H. Zha, “The Total Least Squares Problem,” in C. R. Rao, ed., *Handbook of Statistics*, vol. 9, Elsevier, New York, 1993.

[1289] S. Van Huffel, H. Chen, C. Decanniere, and P. Van Hecke, “Algorithm for Time- Domain NMR Data Fitting Based on Total Least Squares,” *J. Magn. Reson.*, Series A, **110**, 228 (1994).

[1290] V. U. Reddy and L. S. Biradar, “SVD-Based Information Theoretic Criteria for Detection of the Number of Damped/Undamped Sinusoids and Their Performance Analysis,” *IEEE Trans. Sig. Process.*, **41**, 2872 (1993).

[1291] G. Zhu, W. Y. Choy, and B. C. Sanctuary, “Spectral Parameter Estimation by an Iterative Quadratic Maximum Likelihood Method,” *J. Magn. Reson.*, **135**, 37 (1998).

[1292] R. Romano, M. T. Santini, and P. L. Indovina, “A Time-Domain Algorithm for NMR Spectral Normalization,” *J. Magn. Reson.*, **146**, 89 (2000).

[1293] M. Hansson, T. Gänsler, and G. Salomonsson, “Estimation of Single Event- Related Potentials Utilizing the Prony Method,” *IEEE Trans. Biomed. Eng.*, **BME-43**, 51 (1996).

[1294] P. P. Kanjilal, S. Palit, and G. Saha, “Fetal ECG Extraction from Single- Channel Maternal ECG Using Singular Value Decomposition,” *IEEE Trans. Biomed. Eng.*, **BME-44**, 51 (1997).

[1295] D. Callaerts, B. De Moor, J. Vandewalle, and W. Sansen, “Comparison of SVD Methods to Extract the Foetal Electrocardiogram from Cutaneous Electrode Signals,” *Med. Biol. Eng. Comp.*, **28**, 217 (1990).

[1296] H. C. Andrews and C. L. Patterson, “Outer Product Expansions and Their Uses in Image Processing,” *Am. Math. Monthly*, **82**, 1 (1975).

[1297] H. C. Andrews and C. L. Patterson, “Singular Value Decompositions and Digital Image Processing,” *IEEE Trans. Acoust., Speech, Sig. Process.*, **ASSP-24**, 26 (1976).

[1298] USC image database web site: `http://sipi.usc.edu/services/database`.

[1299] J. Durbin, “Efficient Estimation of Parameters of Moving-Average Models,” *Biometrika*, **46**, 306 (1959).

[1300] J. Durbin, “The Fitting of Time Series Models,” *Rev. Int. Statist. Inst.*, **28**, 233 (1961).

[1301] D. Q. Mayne and F. Firoozan, “Linear Identification of ARMA Processes,” *Automatica*, **18**, 461 (1982); and, “An efficient multistage linear identification method for ARMA processes,” *Proc. IEEE Conf. Decision Contr.*, **1**, 435 (1977); and, “Linear Estimation of ARMA Systems,” *IFAC Proc. Volumes*, **11**, no.1, 1907 (1978).

[1302] E. J. Hannan and J. Rissanen, “Recursive Estimation of Mixed Autoregressive-Moving Average Order”, *Biometrika*, **69**, 81 (1982).

SVD – Principal Component Analysis

[1303] H. Hotelling, “Analysis of a Complex of Statistical Variables into Principal Components,” *J. Educ. Psychol.*, **24**, 417 (1933).

[1304] H. Hotelling, “The Most Predictable Criterion,” *J. Educ. Psychol.*, **26**, 139 (1935).

[1305] C. R. Rao, “The Use and Interpretation of Principal Component Analysis in Applied Research,” *Sankhya*, **26**, 329 (1964).

[1306] P. Jolicoeur and J. E. Mosimann, “Size and Shape Variation in the Painted Turtle: A Principal Component Analysis,” *Growth*, **24**, 339 (1960).

[1307] C. S. Bretherton, C. Smith, and J. M. Wallace, “An Intercomparison of Methods for Finding Coupled Patterns in Climate Data,” *J. Climate*, **5**, 541 (1992).

[1308] A. S. Hadi and R. F. Ling, “Some Cautionary Notes on the Use of Principal Components Regression,” *Amer. Statistician*, **52**, 15 (1998).

[1309] D. N. Naik and R. Khattree, “Revisiting Olympic Track Records: Some Practical Considerations in the Principal Component Analysis,” *Amer. Statistician*, **50**, 140 (1996).

[1310] B. G. Kermani, S. S. Schiffman, and H. T. Nagle, “A Novel Method for Reducing the Dimensionality in a Sensor Array,” *IEEE Trans. Instr. Meas.* **IM-47**, 728 (1998).

[1311] J. J. Gerbrands, “On the Relationships Between SVD, KLT, and PCA,” *Patt. Recogn.*, **14**, 375 (1981).

[1312] M. Turk and A. Pentland, “Eigenfaces for Recognition,” *J. Cogn. Neurosci.*, **3**, 71 (1991).

[1313] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman, “Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection,” *IEEE Trans. Patt. Anal. Mach. Intel.*, **PAMI-19**, 711 (1997).

[1314] J. Karhunen and J. Jountsensalo, “Generalizations of Principal Component Analysis, Optimization Problems, and Neural Networks,” *Neural Netw.*, **8**, 549 (1995).

SVD – Canonical Correlation Analysis

[1315] H. Hotelling, “Relations Between Two Sets of Variates,” *Biometrika*, **28**, 321 (1936).

[1316] K. E. Muller, “Understanding Canonical Correlation Through the General Linear Model and Principal Components,” *Amer. Statistician*, **36**, 342 (1982).

[1317] A. C. Pencher, “Interpretation of Canonical Discriminant Functions, Canonical Variates, and Principal Components,” *Amer. Statistician*, **46**, 217 (1992).

[1318] L. L. Scharf and J. K. Thomas, “Wiener Filters in Canonical Coordinates for Transform Coding, Filtering, and Quantizing,” *IEEE Trans. Sig. Process.*, **SP-46**, 647 (1998).

[1319] N. A. Campbell and W. R. Atchley, “The Geometry of Canonical Variate Analysis,” *Syst. Zool.*, **30**, 268 (1981).

[1320] S. N. Afriat, “Orthogonal and Oblique Projectors and the Characteristics of Pairs of Vector Spaces,” *Proc. Camb. Phil. Soc.*, **53**, 800 (1957).

[1321] A. Björck and G. H. Golub, “Numerical Methods for Computing the Angles Between Linear Subspaces,” *Math. Comp.*, **27**, 579 (1973).

SVD – SSA and Chaotic Dynamics

[1322] D. Broomhead and G. P. King, “Extracting Qualitative Dynamics from Experimental Data,” *Physica D*, **20**, 217 (1986).

[1323] R. Vautard, P. Yiou, and M. Ghil, “Singular-Spectrum Analysis: A Toolkit for Short, Noisy, Chaotic Signals,” *Physica D*, **58**, 95 (1992).

[1324] R. Vautard and M. Ghil, “Singular-Spectrum Analysis in Nonlinear Dynamics With Applications to Paleoclimatic Time Series,” *Physica D*, **35**, 395 (1989).

[1325] C. L. Keppenne and M. Ghil, “Adaptive Spectral Analysis and Prediction of the Southern Oscillation Index,” *J. Geophys. Res.*, **97**, 20449 (1992).

[1326] M. R. Allen and L. A. Smith, “Monte Carlo SSA: Detecting Oscillations in the Presence of Coloured Noise,” *J. Climate*, **9**, 3373 (1996).

[1327] SSA toolkit web page, `www.atmos.ucla.edu/tcd/ssa\`.

[1328] M. Ghil and R. Vautard, “Interdecadal Oscillations and the Warming Trend in Global Temperature Time Series,” *Nature*, **350**, 324 (1991).

[1329] M. E. Mann and J. Park, “Spatial Correlations of Interdecadal Variation in Global Surface Temperatures,” *Geophys. Res. Lett.*, **20**, 1055 (1993).

[1330] C. Penland, M. Ghil, and K. Weickmann, “Adaptive Filtering and Maximum Entropy Spectra with Application to Changes in Atmospheric Angular Momentum,” *J. Geophys. Res.*, **96**, 22659 (1991).

[1331] M. Palus and I. Dvorak, “Singular-Value Decomposition in Attractor Reconstruction: Pitfalls and Precautions,” *Physica D*, **55**, 221 (1992).

[1332] V. M. Buchstaber, “Time Series Analysis and Grassmannians,” *Amer. Math. Soc. Transl.*, **162**, 1 (1994).

[1333] J. B. Elsner and A. A. Tsonis, *Singular Spectrum Analysis: A New Tool in Time Series Analysis*, Plenum Press, New York, 1996.

[1334] N. Golyandina, V. Nekrutkin, and A. Zhigliavsky, *Analysis of Time Series Structure: SSA and Related Techniques*, Chapman & Hall/CRC Press, Boca Raton, FL, 2002.

[1335] J. D. Farmer and J. J. Sidorowich, “Exploiting Chaos to Predict the Future and Reduce Noise,” in Y. C. Lee, ed., *Evolution, Learning, and Cognition*, World Scientific, Singapore, 1988.

[1336] A. Basilevsky and D. P. J. Hum, “Karhunen-Loève Analysis of Historical Time Series With an Application to Plantation Births in Jamaica,” *J. Amer. Statist. Assoc.*, **74**, 284 (1979).

[1337] D. L. Danilov, “Principal Components in Time Series Forecast,” *J. Comp. Graph. Statist.*, **6**, 112 (1997).

[1338] R. Cawley and G-H. Hsu, “Local-Geometric-Projection Method for Noise Reduction in Chaotic Maps and Flows,” *Phys. Rev. A*, **46**, 3057 (1992).

[1339] C. Penland and T. Magorian, “Prediction of Niño 3 Sea Surface Temperatures Using Linear Inverse Modeling,” *J. Clim.*, **6**, 1067 (1993).

[1340] T. Sauer, “Time Series Prediction by Using Delay Coordinate Embedding,” in Ref. [1253].

Adaptive Filters

[1341] B. Widrow and M. Hoff, Adaptive Switching Circuits, *IRE Wescon Conv. Rec.*, pt. 4, 96–104 (1960).

[1342] B. Widrow, Adaptive Filters, in R. Kalman and N. DeClaris, Eds., *Aspects of Network and System Theory*, New York, Holt, Rinehart and Winston, 1971.

[1343] M. Honig and D. Messerschmitt, *Adaptive Filters: Structures, Algorithms, and Applications*, Boston, Kluwer Academic, 1984.

[1344] C. F. N. Cowan and P. M. Grant, *Adaptive Filters*, Englewood Cliffs, NJ, Prentice-Hall, 1985.

[1345] A. A. Giordano and F. M. Hsu, *Least Square Estimation with Applications to Digital Signal Processing*, New York, Wiley, 1985.

[1346] B. Widrow and S. D. Steams, *Adaptive Signal Processing*, Englewood Cliffs, NJ, Prentice-Hall, 1985.

[1347] S. T. Alexander, *Adaptive Signal Processing*, New York, Springer-Verlag, 1986.

[1348] S. Haykin, *Adaptive Filter Theory*, Englewood Cliffs, NJ, Prentice-Hall, 1986.

[1349] J. R. Treichler, C. R. Johnson, and M. G. Larimore, *Theory and Design of Adaptive Filters*, New York, Wiley, 1987.

[1350] B. Widrow, et al., Adaptive Noise Cancelling–Principles and Applications, *Proc. IEEE*, **63**, 1692 (1975).

[1351] B. Widrow, et al., Adaptive Antenna Systems, *Proc. IEEE*, **55**, 2143 (1967).

[1352] S. P. Applebaum, Adaptive Arrays, *IEEE Trans. Antennas Prop.*, **AP-24**, 585 (1976).

[1353] F. Gabriel, Adaptive Arrays–An Introduction, *Proc. IEEE*, **64**, 239 (1976).

[1354] A. M. Vural and M. T. Stark, A Summary and the Present Status of Adaptive Array Processing Techniques, *19th IEEE Conference on Decision and Control*, (1980), p.931.

[1355] R. A. Monzingo and T. W. Miller, *Introduction to Adaptive Arrays*, New York, Wiley, 1980.

[1356] B. Widrow, et al., Stationary and Nonstationary Learning Characteristics of the LMS Adaptive Filter, *Proc. IEEE*, **64**, 1151 (1976).

[1357] R. W. Lucky, J. Salz, and E. J. Weldon, Jr., *Principles of Data Communication*, New York, McGraw-Hill, 1968.

[1358] J. G. Proakis, *Digital Communications*, New York, McGraw-Hill, 1983.

[1359] A. P. Clark, *Equalizers for Digital Modems*, New York, Halsted Press, 1985.

[1360] N. A. M. Vierhoeckx, H. Elzen, F. Snijders, and P. Gerwen, Digital Echo Cancellation for Baseband Data Transmission, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-27**, 768 (1979).

[1361] M. M. Sondhi and D. A. Berkley, Silencing Echoes on the Telephone Network, *Proc. IEEE*, **66**, 948 (1980).

[1362] D. L. Duttweiler and Y. S. Chen, A Single Chip VLSI Echo Canceler, *Bell Syst. Tech. J.*, **59**, 149 (1980).

[1363] D. L. Duttweiler, Bell’s Echo-Killer Chip, *IEEE Spectrum*, **17**, 34 (1980).

[1364] D. G. Messerschmitt, Echo Cancellation in Speech and Data Transmission, *IEEE J. Selected Areas in Commun.*, **SAC-2**, 283 (1984).

[1365] C. W. Gritton and D. W. Lin, Echo Cancellation Algorithms, *ASSP Mag.*, **1**, no.2, 30 (1984).

[1366] W. A. Harrison, J. S. Lim, and E. Singer, A New Application of Adaptive Noise Cancellation, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-34**, 21 (1986).

[1367] G. S. Müller and C. K. Pauw, Acoustic Noise Cancellation, *Proc. 1986 Int. Conf. Acoust., Speech, Signal Process.*, Tokyo, p.913.

[1368] J. J. Rodriguez, J. S. Lim, and E. Singer, Adaptive Noise Reduction in Aircraft Communication Systems, *Proc. 1987 Int. Conf. Acoust., Speech, Signal Process.*, Dallas, p.169.

[1369] G. A. Powell, P. Darlington, and P. D. Wheeler, Practical Adaptive Noise Reduction in the Aircraft Cockpit Environment, *Proc. 1987 Int. Conf. Acoust., Speech, Signal Process.*, Dallas, p.173.

[1370] J. Dunlop, M. Al-Kindi, and L. Virr, Application of Adaptive Noise Cancelling to Diver Voice Communications, *Proc. 1987 Int. Conf. Acoust., Speech, Signal Process.*, Dallas, p.1708.

[1371] J. V. Candy, T. Casper, and R. Kane, Plasma Estimation: A Noise Cancelling Application, *Automatica*, **22**, 223 (1986).

[1372] W. Ciciora, G. Sgrignoli, and W. Thomas, A Tutorial on Ghost Cancelling in Television Systems, *IEEE Trans. Consum. Electron.*, **CE-25**, 9 (1979).

[1373] J. Glover, Adaptive Noise Cancelling Applied to Sinusoidal Interferences, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-25**, 484 (1977).

[1374] B. Widrow, J. McCool, and M. Ball, The Complex LMS Algorithm, *Proc. IEEE*, **63**, 719 (1975).

[1375] B. Widrow, K. Duvall, R. Gooch, and W. Newman, Signal Cancellation Phenomena in Adaptive Antennas: Causes and Cures, *IEEE Trans. Antennas Prop.*, **AP-30**, 469 (1982).

[1376] M. J. Shensa, Non-Wiener Solutions of Adaptive Noise Canceller with a Noisy Reference, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-28**, 468 (1980).

[1377] S. J. Elliot and P. Darlington, Adaptive Cancellation of Periodic, Synchronously Sampled Interference, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-33**, 715 (1985).

[1378] S. J. Orfanidis, F. Aafif, and E. Micheli-Tzanakou, Visual Evoked Potential Extraction by Adaptive Filtering, *Proc. 9th IEEE EMBS Conf.*, Boston, November 1987.

[1379] J. R. Treichler, Transient and Convergent Behavior of the Adaptive Line Enhancer, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-27**, 53 (1979).

[1380] D. W. Tufts, L. J. Griffiths, B. Widrow, J. Glover, J. McCool, and J. Treichler, Adaptive Line Enhancement and Spectrum Analysis, *Proc. IEEE*, **65**, 169 (1977).

[1381] J. R. Zeidler, et al., Adaptive Enhancement of Multiple Sinusoids in Uncorrelated Noise, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-26**, 240 (1978).

[1382] L. J. Griffiths, Rapid Measurement of Digital Instantaneous Frequency, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-23**, 207 (1975).

[1383] D. Morgan and S. Craig, Real-Time Linear Prediction Using the Least Mean Square Gradient Algorithm, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-24**, 494 (1976).

[1384] P. Eykhoff, *System Identification: Parameter and State Estimation*, New York, Wiley, 1974.

[1385] K. J. Åström and P. Eykhoff, System Identification–A Survey, *Automatica*, **7**, 123 (1971).

[1386] G. C. Goodwin and R. L. Payne, *Dynamic System Identification, Experimental Design and Data Analysis*, New York, Academic, 1977.

[1387] L. Ljung and T. Söderström, *Theory and Practice of Recursive Identification*, Cambridge, MA, MIT Press, 1983.

[1388] L. Ljung, *System Identification: Theory for the User*, Englewood Cliffs, NJ, Prentice-Hall, 1987.

[1389] K. J. Åström and B. Wittenmark, *Computer Controlled Systems*, Englewood Cliffs, NJ, Prentice-Hall, 1984.

[1390] K. J. Åström, Adaptive Feedback Control, *Proc. IEEE*, **75**, 185 (1987).

[1391] N. Sundararajan and R. C. Montgomery, Identification of Structural Dynamics Systems Using Least-Squares Lattice Filters, *J. Guidance and Control*, **6**, 374 (1983).

[1392] N. Sundararajan, J. P. Williams, and R. C. Montgomery, Adaptive Modal Control of Structural Dynamic Systems Using Recursive Lattice Filters, *J. Guidance and Control*, **8**, 223 (1985).

[1393] W. S. Hodgkiss and J. A. Presley, Jr., Adaptive Tracking of Multiple Sinusoids whose Power Levels are Widely Separated, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-29**, 710 (1981).

[1394] W. F. Gabriel, Spectral Analysis and Adaptive Array Superresolution Techniques, *Proc. IEEE*, **68**, 654 (1980).

[1395] W. F. Gabriel, Using Spectral Estimation Techniques in Adaptive Processing Antenna Systems, *IEEE Trans. Antennas Propag.*, **AP-34**, 291 (1986).

[1396] F. M. Hsu and A. A. Giordano, Digital Whitening Techniques for Improving Spread Spectrum Communications Performance in the Presence of Narrowband Jamming and Interference, *IEEE Trans. Commun.*, **COM-26**, 209 (1978).

[1397] J. W. Ketchum and J. G. Proakis, Adaptive Algorithms for Estimating and Suppressing Narrow-Band Interference in PN Spread-Spectrum Systems, *IEEE Trans. Commun.*, **COM-30**, 913 (1982).

[1398] L. M. Li and L. B. Milstein, Rejection of Narrow-Band Interference in PN Spread-Spectrum Systems Using Transversal Filters, *IEEE Trans. Commun.*, **COM-30**, 925 (1982).

[1399] R. A. Iltis and L. B. Milstein, Performance Analysis of Narrow-Band Interference Rejection Techniques in DS Spread-Spectrum Systems, *IEEE Trans. Commun.*, **COM-32**, 1169 (1984).

[1400] E. Masry, Closed-Form Analytical Results for the Rejection of Narrow-Band Interference in PN Spread-Spectrum Systems–Part I: Linear Prediction Filters, *IEEE Trans. Commun.*, **COM-32**, 888 (1984).

[1401] E. Masry, Closed-Form Analytical Results for the Rejection of Narrow-Band Interference in PN Spread-Spectrum Systems–Part II: Linear Interpolation Filters, *IEEE Trans. Commun.*, **COM-33**, 10 (1985).

[1402] A. Reichman and R. A. Scholtz, Adaptive Spread-Spectrum Systems Using Least-Squares Lattice Filters, *IEEE J. Selected Areas Commun.*, **SAC-3**, 652 (1985).

[1403] P. A. Thompson, An Adaptive Spectral Analysis Technique for Unbiased Frequency Estimation in the Presence of White Noise, *Proc. 13th Asilomar Conf. Circuits, Systems, and Computers*, p.529 (Nov. 1979).

[1404] M. G. Larimore and R. J. Calvert, Convergence Studies of Thompson's Unbiased Adaptive Spectral Estimator, *Proc. 14th Asilomar Conf. Circuits, Systems, and Computers*, p.258 (Nov. 1980).

[1405] V. U. Reddy, B. Egard, and T. Kailath, Least Squares Type Algorithm for Adaptive Implementation of Pisarenko's Harmonic Retrieval Method, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-30**, 399 (1982).

[1406] F. K. Soong and A. M. Petersen, On the High Resolution and Unbiased Frequency Estimates of Sinusoids in White Noise–A New Adaptive Approach, *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, p.1362 (April 1982).

[1407] A. Cantoni and L. Godara, Resolving the Directions of Sources in a Correlated Field Incident on an Array, *J. Acoust. Soc. Am.*, **67**, 1247 (1980).

[1408] S. J. Orfanidis and L. M. Vail, Zero-Tracking Adaptation Algorithms, *Proc. ASSP Spectrum Estimation Workshop, II*, Tampa, FL (November 1983).

[1409] S. J. Orfanidis and L. M. Vail, Zero Tracking Adaptive Filters, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-34**, 1566 (1986).

[1410] Z. Rogowski, I. Gath, and E. Bental, On the Prediction of Epileptic Seizures, *Biol. Cybernetics*, **42**, 9 (1981).

[1411] L. J. Griffiths, A Continuously-Adaptive Filter Implemented as a Lattice Structure, *Int. Conf. Acoust., Speech, Signal Process.*, Hartford CT, p.87 (1977).

[1412] J. Makhoul, A Class of All-Zero Lattice Digital Filters: Properties and Applications, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-26**, 304 (1978).

[1413] E. H. Satorius and S. T. Alexander, Channel Equalization Using Adaptive Lattice Algorithms, *IEEE Trans. Commun.*, **COM-27**, 899 (1979).

[1414] C. J. Gibson and S. Haykin, Learning Characteristics of Adaptive Lattice Filtering Algorithms, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-28**, 681 (1980).

[1415] M. L. Honig and D. G. Messerschmitt, Convergence Properties of the Adaptive Digital Lattice Filter, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-29**, 642 (1981).

[1416] R. S. Medaugh and L. J. Griffiths, A Comparison of Two Fast Linear Predictors, *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Atlanta, GA (March 1981), p.293.

[1417] C. Giraudon, Results on Active Sonar Optimum Array Processing, in J. W. R. Griffiths, et al., Eds., *Signal Processing*, New York, Academic, 1973.

[1418] W. D. White, Cascade Preprocessors for Adaptive Antennas, *IEEE Trans. Antennas Propag.*, **AP-24**, 670 (1976).

[1419] D. H. Brandwood and C. J. Tarran, Adaptive Arrays for Communications, *IEE Proc.*, **129**, Pt. F, 223 (1982).

[1420] J. G. McWhirter and T. J. Shepherd, Adaptive Algorithms in the Space and Time Domains, *IEE Proc.*, **130**, Pts. F and H, 17 (1983).

[1421] F. Ling, D. Manolakis, and J. G. Proakis, A Recursive Modified Gram-Schmidt Algorithm for Least-Squares Estimation, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-34**, 829 (1986).

[1422] D. D. Falconer and L. Ljung, Application of Fast Kalman Estimation to Adaptive Equalization, *IEEE Trans. Commun.*, **COM-26**, 1439 (1976).

[1423] L. Ljung, M. Morf, and D. Falconer, Fast Calculations of Gain Matrices for Recursive Estimation Schemes, *Int. J. Control*, **27**, 1 (1978).

[1424] G. C. Carayannis, D. Manolakis, and N. Kalouptsidis, A Fast Sequential Algorithm for Least-Squares Filtering and Prediction, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-31**, 1394 (1983).

[1425] J. Cioffi and T. Kailath, Fast, Recursive Least-Squares, Transversal Filters for Adaptive Processing, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-34**, 304 (1984).

[1426] L. S. DeJong, Numerical Aspects of Recursive Realization Algorithms, *SIAM J. Control Optimiz.*, **16**, 646 (1978).

[1427] M. S. Mueller, On the Rapid Initial Convergence of Least-Squares Equalizer Adjustment Algorithms, *Bell Syst. Tech. J.*, **60**, 2345 (1981).

[1428] D. W. Lin, On the Digital Implementation of the Fast Kalman Algorithm, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-32**, 998 (1984).

[1429] F. Ling and J. G. Proakis, Numerical Accuracy and Stability: Two Problems of Adaptive Estimation Algorithms Caused by Round-Off Error, *Proc. 1984 IEEE Int. Conf. Acoust., Speech, Signal Process.*, San Diego, CA, p.30.3.1.

[1430] C. G. Samson and V. U. Reddy, Fixed Point Error Analysis of the Normalized Ladder Algorithm, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-31**, 1177 (1983).

[1431] S. Ljung and L. Ljung, Error Propagation Properties of Recursive Least-Squares Adaptation Algorithms, *Automatica*, **21**, 157 (1985).

[1432] D. Manolakis, G. Carayannis, and V. Zemas, Fast RLS Algorithms for Adaptive Filtering: Some Engineering Problems, *Proc. 1987 IEEE Int. Conf. Circuits and Systems*, Philadelphia, PA, p.985.

[1433] S. H. Ardalan and S. T. Alexander, Fixed-Point Roundoff Error Analysis of the Exponentially Windowed RLS Algorithm for Time-Varying Systems, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-35**, 770 (1987).

[1434] C. Caraiscos and B. Liu, A Roundoff Error Analysis of the LMS Adaptive Algorithm, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-32**, 34 (1984).

[1435] J. M. Ciofi, Limited-Precision Effects in Adaptive Filtering, *IEEE Trans. Circ. Syst.*, **CAS-34**, 821 (1987).

[1436] M. Morf and D. T. L. Lee, Recursive Least-Squares Ladder Forms for Fast Parameter Tracking, *Proc. 17th IEEE Conf. Decision Contr.*, p.1326 (1979).

[1437] E. H. Satorius and M. J. Shensa, Recursive Lattice Filters–A Brief Overview, *Proc. 19th IEEE Conf. Decision Contr.*, p.955 (1980).

[1438] D. Lee, M. Morf, and B. Friedlander, Recursive Square-Root Ladder Estimation Algorithms, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-29**, 627 (1981).

[1439] M. J. Shensa, Recursive Least-Squares Lattice Algorithms: A Geometrical Approach, *IEEE Trans. Autom. Control*, **AC-26**, 695 (1981).

[1440] E. H. Satorius and J. D. Pack, Application of Least-Squares Lattice Algorithms to Channel Equalization, *IEEE Trans. Commun.*, **COM-29**, 136 (1981).

[1441] E. Schichor, Fast Recursive Estimation Using the Lattice Structure, *Bell Syst. Tech. J.*, **61**, 97 (1981).

[1442] M. S. Mueller, Least-Squares Algorithms for Adaptive Equalizers, *Bell Syst. Tech. J.*, **60**, 1905 (1981).

[1443] B. Friedlander, Lattice Filters for Adaptive Processing, *Proc. IEEE*, **70**, 829 (1982).

[1444] G. C. Carayannis, D. Manolakis, and N. Kalouptsidis, A Unified View of Parametric Processing Algorithms for Prewindowed Signals, *Signal Processing*, **10**, 335 (1986).

[1445] F. Ling, D. Manolakis, and J. G. Proakis, Numerically Robust Least-Squares Lattice-Ladder Algorithms with Direct Updating of the Reflection Coefficients, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-34**, 837 (1986).

[1446] P. E. Gill, G. H. Golub, W. Murray, and M. A. Saunders, Methods of Modifying Matrix Factorizations, *Math. Comp.*, **28**, 505 (1974).

[1447] P. E. Gill, W. Murray, and M. A. Saunders, Methods for Computing and Modifying the LVD Factors of a Matrix, *Math. Comp.*, **29**, 1051 (1975).

[1448] D. Godard, Channel Equalization Using a Kalman Filter for Fast Data Transmission, *IBM J. Res. Dev.*, **18**, 267 (1974).

[1449] R. D. Gitlin and F. R. Magee, Self-Orthogonalizing Adaptive Equalization Algorithms, *IEEE Trans. Commun.*, **COM-25**, 666 (1977).

[1450] R. W. Chang, A New Equalizer Structure for Fast Start-up Digital Communication, *Bell Syst. Tech. J.*, **50**, 1969 (1971).

[1451] J. G. McWhirter and T. J. Shepherd, Least-Squares Lattice Algorithm for Adaptive Channel Equalization–A Simplified Derivation, *IEE Proc.*, **130**, Pt. F, 532 (1983).

[1452] J. Mendel, *Discrete Techniques of Parameter Estimation*, New York, Marcel Dekker, 1973.

[1453] L. E. Brennan, J. D. Mallet, and I. S. Reed, Adaptive Arrays in Airborne MTI Radar, *IEEE Trans. Antenn. Propag.*, **AP-24**, 607 (1976).

[1454] L. E. Brennan and I. S. Reed, Theory of Adaptive Radar, *IEEE Trans. Aerosp. Electron. Syst.*, **AES-9**, 237 (1973).

[1455] L. E. Brennan, J. D. Mallet, and I. S. Reed, Rapid Convergence Rate in Adaptive Arrays, *IEEE Trans. Aerosp. Electron. Syst.*, **AES-10**, 853 (1974).

[1456] J. Cioffi, When Do I Use an RLS Adaptive Filter? *Proc. 19th IEEE Asilomar Conf. Circ., Syst., Computers*, 1986, p.636.

[1457] E. Eleftheriou and D. D. Falconer, Tracking Properties and Steady-State Performance of RLS Adaptive Filter Algorithms, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-34**, 1097 (1986).

[1458] G. H. Golub, Some Modified Matrix Eigenvalue Problems, *SIAM Rev.*, **15**, 318 (1973).

[1459] J. R. Bunch, C. P. Nielsen, and D. C. Sorensen, Rank-One Modification of the Symmetric Eigenproblem, *Numer. Math.*, **31**, 31 (1978).

[1460] K. J. Bathe and E. L. Wilson, *Numerical Methods in Finite Element Analysis*, Englewood Cliffs, NJ, Prentice-Hall, 1976.

[1461] W. Bühring, Adaptive Orthogonal Projection for Rapid Converging Interference Suppression, *Electron. Lett.*, **14**, 515 (1978).

[1462] N. L. Owsley, Adaptive Data Orthogonalization, *Proc. 1978 Int. Conf. Acoust., Speech, Signal Process.*, Tulsa, p.109.

[1463] J. Karhunen, Adaptive Algorithms for Estimating Eigenvectors of Correlation Type Matrices, *Proc. 1984 Int. Conf. Acoust., Speech, Signal Process.*, San Diego, CA, p.14.6.1.

[1464] Y. H. Hu, Adaptive Methods for Real Time Pisarenko Spectrum Estimate, *Proc. 1985 Int. Conf. Acoust., Speech, Signal Process.*, Tampa, FL, p.105.

[1465] K. C. Sharman, T. S. Durrani and L. Vergara-Dominguez, Adaptive Algorithms for Eigenstructure Based Spectral Estimation and Filtering, *Proc. 1986 IEEE Int. Conf. Decision and Control*, Athens, p.2224.

[1466] K. C. Sharman and T. S. Durrani, Eigenfilter Approaches to Adaptive Array Processing, *IEE Proc.*, **130**, Pt. F, 22 (1983).

[1467] J. F. Yang and M. Kaveh, Adaptive Signal-Subspace Algorithms for Frequency Estimation and Tracking, *Proc. 1987 Int. Conf. Acoust., Speech, Signal Process.*, Dallas, p.1593.

[1468] C. Samson, A Unified Treatment of Fast Algorithms for Identification, *Int. J. Control*, **35**, 909 (1982).

[1469] M. Honig, Recursive Fixed-Order Covariance Least-Squares Algorithms, *Bell Syst. Tech. J.*, **62**, 2961 (1983).

[1470] H. Lev-Ari and T. Kailath, Least-Squares Adaptive Lattice and Transversal Filters: A Unified Geometric Theory, *IEEE Trans. Inform. Th.*, **IT-30**, 222 (1984).

[1471] N. Kalouptsidis, G. Carayannis, and D. Manolakis, Fast Design of FIR Least-Squares Filters with Optimum Lag, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-32**, 48 (1984).

[1472] N. Kalouptsidis, G. Carayannis, and D. Manolakis, Efficient Recursive-in-Order Least Squares FIR Filtering and Prediction, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-33**, 1175 (1985).

[1473] A. Nehorai and M. Morf, A Unified Derivation for Fast Estimation Algorithms by the Conjugate Direction Method, *Lin. Alg. Appl.*, **72**, 119 (1985).

[1474] J. D. Wang and H. J. Trussell, A Unified Derivation of the Fast RLS Algorithms, *Proc. 1986 Int. Conf. Acoust., Speech, Signal Process.*, Tokyo, p.261.

[1475] S. T. Alexander, Fast Adaptive Filters: A Geometrical Approach, *ASSP Mag.*, **3**, no. 4, 18 (1986).

[1476] N. Kalouptsidis and S. Theodoridis, Fast Adaptive Least Squares Algorithms for Power Spectral Estimation, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-35**, 661(1987).

[1477] D. Manolakis, F. Ling, and J. G. Proakis, Efficient Time-Recursive Least-Squares Algorithms for Finite-Memory Adaptive Filtering, *IEEE Trans. Circ. Syst.*, **CAS-34**, 400 (1987).

[1478] J. G. McWhirter, Recursive Least-Squares Minimization Using a Systolic Array, *Proc. SPIE, Real-Time Signal Processing IV*, **431**, 105 (1983).

[1479] F. Ling and J. G. Proakis, A Generalized Multichannel Least Squares Lattice Algorithm Based on Sequential Processing Stages, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-32**, 381 (1984).

[1480] C. R. Ward, A. J. Robson, P. J. Hargrave, and J. G. McWhirter, Application of a Systolic Array to Adaptive Beamforming, *IEE Proc.*, **131**, Pt. F, 638 (1984).

[1481] H. Sakai, A Parallel Least-Squares Linear Prediction Method Based on the Circular Lattice Filter, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-34**, 640 (1986).

[1482] R. Schreiber, Implementation of Adaptive Array Algorithms, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-34**, 1038 (1986).

[1483] H. Kimura and T. Osada, Canonical Pipelining of Lattice Filters, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-35**, 878 (1987).

[1484] H. Lev-Ari, Modular Architectures for Adaptive Multichannel Lattice Algorithms, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-35**, 543 (1987).

[1485] T. H. Meng and D. G. Messerschmitt, Arbitrarily High Sampling Rate Adaptive Filters, *IEEE Trans. Acoust., Speech, Signal Process.*, **ASSP-35**, 455 (1987).

[1486] M. G. Bellanger, *Adaptive Digital Filters and Signal Analysis*, New York, Marcel Dekker, 1987.

[1487] S. J. Orfanidis, The Double/Direct RLS Lattice, *Proc. 1988 Int. Conf. Acoust., Speech, Signal Process.*, New York.

Index