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:

$$E[(\Delta E)^2] = \frac{E^2}{N}, \quad E[\Delta\mathbf{a}\,\Delta E] = 0, \quad E[\Delta\mathbf{a}\,\Delta\mathbf{a}^\dagger] = \frac{E}{N}(R^{-1} - E^{-1}\mathbf{a}\,\mathbf{a}^\dagger)$$

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

$$E[(\Delta S(k))^2] = \frac{1}{N} S(k)^2$$

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

$$E[|\Delta A(k)|^2] = \frac{E}{N}[\mathbf{s}_k^\dagger R^{-1}\mathbf{s}_k - E^{-1}|A(k)|^2]$$

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

$$E[(\Delta S(k))^2] = \frac{1}{N} S(k)^2 [2S(k)(\mathbf{s}_k^\dagger R^{-1}\mathbf{s}_k) - 1]$$

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:

$$\begin{aligned} \|\mathbf{x}\|_2 &= \sqrt{|x_1|^2 + |x_2|^2 + \cdots + |x_N|^2} = \sqrt{\mathbf{x}^T\mathbf{x}} \\ \|\mathbf{x}\|_1 &= |x_1| + |x_2| + \cdots + |x_N| \\ \|\mathbf{x}\|_\infty &= \max(|x_1|, |x_2|, \ldots, |x_N|) \end{aligned} \qquad \text{where} \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} \tag{15.1.1}$$

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:

$$|\mathbf{x}^T\mathbf{y}| \le \|\mathbf{x}\|\,\|\mathbf{y}\| \tag{15.1.3}$$

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:

$$\|A\|_F = \sqrt{\sum_{i,j} |A_{ij}|^2} = \sqrt{\text{tr}\left(A^TA\right)} \qquad \text{(Frobenius norm)} \tag{15.1.7}$$

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 $\{\mathbf{b}_1,\mathbf{b}_2,\ldots,\mathbf{b}_M\}$, 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 $\mathbf{b} \in Y$:

$$\mathbf{b} = \sum_{i=1}^{M} c_i\mathbf{b}_i = c_1\mathbf{b}_1 + c_2\mathbf{b}_2 + \cdots c_M\mathbf{b}_M = [\mathbf{b}_1,\mathbf{b}_2,\ldots,\mathbf{b}_M]\begin{bmatrix} c_1\\ c_2\\ \vdots\\ c_M\end{bmatrix} = B\mathbf{c} \tag{15.2.1}$$

where we defined the $N\times M$ basis matrix $B = [\mathbf{b}_1,\mathbf{b}_2,\ldots,\mathbf{b}_M]$ and the $M\times 1$ vector of expansion coefficients $\mathbf{c} = [c_1,c_2\ldots,c_M]^T$.

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

[†]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.

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):

$$\mathbf{c} = B\backslash\mathbf{b} = \text{pinv}(B)*\mathbf{b} = B^+\mathbf{b} \tag{15.2.3}$$

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 $(B^TB)_{ij} = \mathbf{b}_i^T\mathbf{b}_j$, $i,j = 1,2,\ldots,M$.

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.

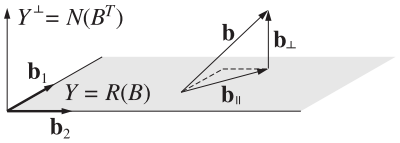


Fig. 15.2.1 Projection of $\mathbf{b}$ onto the subspace $Y = R(B)$ spanned by $B = [\mathbf{b}_1, \mathbf{b}_2]$.

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.

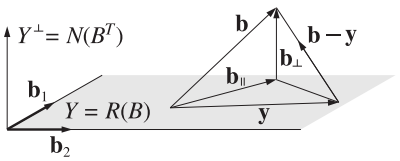


Fig. 15.2.2 The projection $\mathbf{b}_\parallel$ minimizes the distance $\|\mathbf{b} - \mathbf{y}\|$ to the subspace Y.

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 $U = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_M]$ 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 $\mathbf{u}_i$:

$$\mathbf{b}_j = \sum_{i=1}^{M} \mathbf{u}_i c_{ij}\,, \quad j = 1, 2, \ldots, M \tag{15.2.8}$$

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:

$$\mathcal{P} = B(B^TB)^{-1}B^T = UC\big(C^T(U^TU)C\big)^{-1}C^TU^T$$
$$= UC\big(C^{-1}(U^TU)^{-1}C^{-T}\big)C^TU^T = U(U^TU)^{-1}U^T$$

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 $i, j = 1, 2, \ldots, M$. Compactly, we may express this condition in terms of the basis matrix $U = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_M]$:

$$U^TU = I_M \qquad \text{(orthonormal basis)} \tag{15.2.10}$$

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 $U_1^T U_1 = U_2^T U_2 = I_3$, and C_2 is upper triangular. The projection matrices onto Y and $Y^\perp$ are:

$$\mathcal{P} = U_1 U_1^T = \frac{1}{4}\begin{bmatrix} 3 & -1 & -1 & -1 \\ -1 & 3 & -1 & -1 \\ -1 & -1 & 3 & -1 \\ -1 & -1 & -1 & 3 \end{bmatrix}, \quad \mathcal{Q} = I_4 - \mathcal{P} = \frac{1}{4}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

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:

$$\begin{array}{llll} R(A), & \text{subspace of } \mathbb{R}^N, & \dim = r, & R(A)^\perp = N(A^T) \\ N(A^T), & \text{subspace of } \mathbb{R}^N, & \dim = N-r, & N(A^T)^\perp = R(A) \\ R(A^T), & \text{subspace of } \mathbb{R}^M, & \dim = r, & R(A^T)^\perp = N(A) \\ N(A), & \text{subspace of } \mathbb{R}^M, & \dim = M-r, & N(A)^\perp = R(A^T) \end{array} \tag{15.3.1}$$

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:

$$\begin{aligned} \mathbb{R}^N &= R(A) \oplus N(A^T) = R(A) \oplus R(A)^\perp \\ \mathbb{R}^M &= R(A^T) \oplus N(A) = R(A^T) \oplus R(A^T)^\perp \end{aligned} \tag{15.3.2}$$

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].

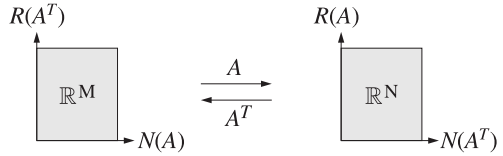


Fig. 15.3.1 The four fundamental subspaces associated with an $N\times M$ matrix A.

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,

$$\mathcal{J} = \|\mathbf{e}\|^2 = \mathbf{e}^T\mathbf{e} = \|\mathbf{b} - A\mathbf{x}\|^2 = (\mathbf{b} - A\mathbf{x})^T(\mathbf{b} - A\mathbf{x}) = \min \tag{15.4.1}$$

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:

$$\begin{aligned} A^T\mathbf{e} &= 0 \qquad \text{(orthogonality equations)} \\ A^TA\mathbf{x} &= A^T\mathbf{b} \qquad \text{(normal equations)} \end{aligned} \tag{15.4.2}$$

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

$$\mathbf{x} = (A^TA)^{-1}A^T\mathbf{b} \qquad \text{(full-rank overdetermined case)} \tag{15.4.3}$$

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.

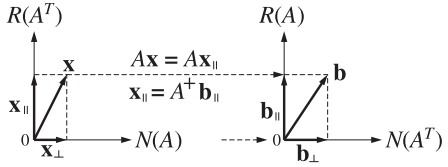


Fig. 15.4.1 Role of the fundamental subspaces in the least-squares solution of $A\mathbf{x} = \mathbf{b}$.

Using the direct-sum decompositions (15.3.2), we resolve both $\mathbf{b}$ and $\mathbf{x}$ into their unique orthogonal components:

$$\begin{aligned} &\mathbf{b} = \mathbf{b}_\parallel + \mathbf{b}_\perp, \quad \mathbf{b}_\parallel \in R(A), \quad \mathbf{b}_\perp \in N(A^T), \quad \mathbf{b} \in \mathbb{R}^N \\ &\mathbf{x} = \mathbf{x}_\parallel + \mathbf{x}_\perp, \quad \mathbf{x}_\parallel \in R(A^T), \quad \mathbf{x}_\perp \in N(A), \quad \mathbf{x} \in \mathbb{R}^M \end{aligned} \tag{15.4.4}$$

Because $\mathbf{x}_\perp$ lies in the null space of A, we have $A\mathbf{x}_\perp = 0$, and therefore, $A\mathbf{x} = A(\mathbf{x}_\parallel + \mathbf{x}_\perp) = A\mathbf{x}_\parallel$. Then, the error vector becomes:

$$\mathbf{e} = \mathbf{b} - A\mathbf{x} = (\mathbf{b}_\parallel - A\mathbf{x}_\parallel) + \mathbf{b}_\perp \equiv \mathbf{e}_\parallel + \mathbf{e}_\perp \tag{15.4.5}$$

Because both $\mathbf{b}_\parallel$ and $A\mathbf{x}_\parallel$ lie in $R(A)$, so does $\mathbf{e}_\parallel = \mathbf{b}_\parallel - A\mathbf{x}_\parallel$, 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 $\mathbf{e}_\parallel = 0$, or, $A\mathbf{x}_\parallel = \mathbf{b}_\parallel$. Because $\mathbf{b}_\parallel$ lies in $R(A)$, this system will have a solution $\mathbf{x}_\parallel$.

Moreover, because $\mathbf{x}_\parallel \in R(A^T)$, this solution will be unique. Indeed, if $\mathbf{b}_\parallel = A\mathbf{x}_\parallel = A\mathbf{x}'_\parallel$, for another vector $\mathbf{x}'_\parallel \in R(A^T)$, then $A(\mathbf{x}_\parallel - \mathbf{x}'_\parallel) = 0$, or, $\mathbf{x}_\parallel - \mathbf{x}'_\parallel$ 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 $\mathbf{x}_\parallel$ may be constructed by the pseudoinverse of A:

$$A\mathbf{x}_\parallel = \mathbf{b}_\parallel \quad \Rightarrow \quad \mathbf{x}_\parallel = A^+\mathbf{b}_\parallel = A^+\mathbf{b} \qquad \text{(minimum-norm solution)} \tag{15.4.6}$$

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 $A^+\mathbf{b}_\parallel = A^+\mathbf{b}$. 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 $\mathbf{x}_\parallel$ is also recognized to be that particular solution of the least-squares problem that has *minimum norm*. This follows from (15.4.4):

$$\|\mathbf{x}\|^2 = \|\mathbf{x}_\parallel\|^2 + \|\mathbf{x}_\perp\|^2 = \|A^+\mathbf{b}\|^2 + \|\mathbf{x}_\perp\|^2 \tag{15.4.8}$$

which shows that the norm $\|\mathbf{x}\|$ is minimum when $\mathbf{x}_\perp = 0$, or, when $\mathbf{x} = \mathbf{x}_\parallel$. Fig. 15.4.1 illustrates this property.

The minimum-norm solution is computed by MATLAB's built-in function **pinv**, $\mathbf{x}_\parallel =$ pinv$(A)*\mathbf{b}$. The solution obtained by MATLAB's backslash operator, $\mathbf{x} = A\backslash\mathbf{b}$, does not, in general, coincide with $\mathbf{x}_\parallel$. 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 $\mathbf{e}_\parallel$ and $\mathbf{e}_\perp$ are orthogonal:

$$\mathcal{J} = \|\mathbf{e}\|^2 = \|\mathbf{e}_\parallel\|^2 + \|\mathbf{e}_\perp\|^2 = \|\mathbf{b}_\parallel - A\mathbf{x}_\parallel\|^2 + \|\mathbf{b}_\perp\|^2 \tag{15.4.9}$$

This expression is minimized when $A\mathbf{x}_\parallel = \mathbf{b}_\parallel$, 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 $\mathbf{x} = \mathbf{x}_\parallel$ 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 $A^+ = (A^TA)^{-1}A^T$.

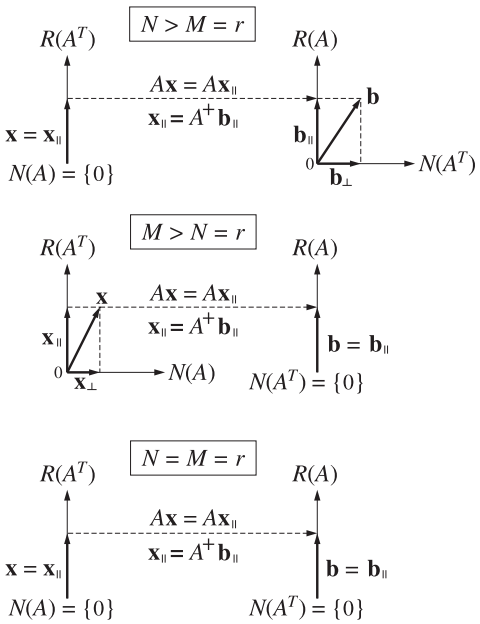


Fig. 15.4.2 Subspaces in the full-rank least-squares solutions of $A\mathbf{x} = \mathbf{b}$.

In the under-determined case, we have $\mathbf{b} = \mathbf{b}_\parallel$, 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 $\mathbf{x} = \mathbf{x}_\parallel + \mathbf{x}_\perp$. The minimum norm solution can be constructed as follows.

Because $\mathbf{x}_\parallel \in R(A^T)$, there is a coefficient vector $\mathbf{c} \in \mathbb{R}^N$ such that $\mathbf{x}_\parallel = A^T\mathbf{c}$. Then, the system reads $\mathbf{b} = A\mathbf{x} = A\mathbf{x}_\parallel = AA^T\mathbf{c}$. 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, $\mathbf{x}_\parallel = A^T\mathbf{c} = A^T(AA^T)^{-1}\mathbf{b}$. 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:

$$\begin{aligned} &1.\quad N > M = r, && \mathbf{x} = A^+\mathbf{b}, && A^+ = (A^TA)^{-1}A^T \\ &2.\quad M > N = r, && \mathbf{x} = A^+\mathbf{b} + \mathbf{x}_\perp, && A^+ = A^T(AA^T)^{-1} \\ &3.\quad N = M = r, && \mathbf{x} = A^{-1}\mathbf{b}, && A^+ = A^{-1} \end{aligned} \tag{15.4.10}$$

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:

$$\begin{cases} x = 1 \\ x = 2 \end{cases} \qquad \text{and} \qquad \begin{cases} 2x = 2 \\ x = 2 \end{cases}$$

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 $A\mathbf{x} = \mathbf{b}$:

$$\begin{matrix} a_1x = b_1 \\ a_2x = b_2 \end{matrix} \quad \Rightarrow \quad \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} x = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}, \quad A = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \tag{15.4.11}$$

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:

$$\begin{bmatrix} a_1 \\ a_2 \end{bmatrix}^+ = \frac{[a_1, a_2]}{a_1^2 + a_2^2} \quad \Rightarrow \quad x = A^+\mathbf{b} = \frac{1}{a_1^2 + a_2^2}[a_1, a_2]\begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = \frac{a_1b_1 + a_2b_2}{a_1^2 + a_2^2}$$

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:

$$\begin{bmatrix} w_1a_1 \\ w_2a_2 \end{bmatrix} x = \begin{bmatrix} w_1b_1 \\ w_2b_2 \end{bmatrix} \quad \Rightarrow \quad x = \frac{w_1^2a_1b_1 + w_2^2a_2b_2}{w_1^2a_1^2 + w_2^2a_2^2} \tag{15.4.12}$$

The differences between (15.4.11) and (15.4.12) can be explained by inspecting the corresponding performance indices that are being minimized:

$$\mathcal{J} = (a_1x - b_1)^2 + (a_2x - b_2)^2, \qquad \mathcal{J} = w_1^2(a_1x - b_1)^2 + w_2^2(a_2x - b_2)^2$$

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.

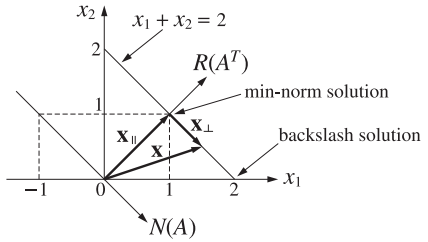


Fig. 15.4.3 The minimum norm solution is perpendicular to the straight line $x_1 + x_2 = 2$.

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:

$$a_1x_1 + a_2x_2 = b_1 \quad \Leftrightarrow \quad [a_1, a_2]\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = [b_1], \quad A = [a_1, a_2], \quad \mathbf{b} = [b_1]$$

The pseudoinverse of A and the min-norm solution are:

$$A^+ = A^T(AA^T)^{-1} = \frac{1}{a_1^2+a_2^2}\begin{bmatrix} a_1 \\ a_2 \end{bmatrix}, \quad \mathbf{x}_\parallel = A^+\mathbf{b} = \frac{1}{a_1^2+a_2^2}\begin{bmatrix} a_1b_1 \\ a_2b_1 \end{bmatrix}$$

Vectors in $N(A)$ and the most general least-squares solution are given by:

$$\mathbf{x}_\perp = \frac{1}{a_1^2+a_2^2}\begin{bmatrix} a_2z \\ -a_1z \end{bmatrix}, \quad \mathbf{x} = \mathbf{x}_\parallel + \mathbf{x}_\perp = \frac{1}{a_1^2+a_2^2}\begin{bmatrix} a_1b_1+a_2z \\ a_2b_1-a_1z \end{bmatrix}$$

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:

$$\mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_N \end{bmatrix} \quad \Rightarrow \quad \mathbf{a}^+ = \frac{\mathbf{a}^T}{\|\mathbf{a}\|^2} \quad \text{and} \quad \mathbf{a}^T = [a_1, a_2, \ldots, a_N] \quad \Rightarrow \quad (\mathbf{a}^T)^+ = \frac{\mathbf{a}}{\|\mathbf{a}\|^2}$$

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:

$$\begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_N \end{bmatrix} x = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{bmatrix} \quad \Rightarrow \quad x = \mathbf{a}^+\mathbf{b} = \frac{\mathbf{a}^T\mathbf{b}}{\mathbf{a}^T\mathbf{a}} = \frac{a_1b_1 + a_2b_2 + \cdots + a_Nb_N}{a_1^2 + a_2^2 + \cdots + a_N^2}$$

$$a_1x_1 + a_2x_2 + \cdots + a_Nx_N = b \quad \Rightarrow \quad \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} = \frac{1}{a_1^2 + a_2^2 + \cdots + a_N^2}\begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_N \end{bmatrix} b$$

15.5 The Singular Value Decomposition

Given an $N\times M$ matrix $A \in \mathbb{R}^{N\times M}$ of rank $r \leq \min(N, M)$, 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 $\mathbb{R}^{r\times r}$:

$$\boxed{\Sigma_r = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_r)} \tag{15.5.3}$$

with positive diagonal entries called the *singular values* of A and arranged in decreasing order:

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0 \tag{15.5.4}$$

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 $r = 1, 2, 3, 4$:

$$\left[\begin{array}{c|ccc} \sigma_1 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right], \left[\begin{array}{cc|cc} \sigma_1 & 0 & 0 & 0 \\ 0 & \sigma_2 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right], \left[\begin{array}{ccc|c} \sigma_1 & 0 & 0 & 0 \\ 0 & \sigma_2 & 0 & 0 \\ 0 & 0 & \sigma_3 & 0 \\ \hline 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right], \left[\begin{array}{cccc} \sigma_1 & 0 & 0 & 0 \\ 0 & \sigma_2 & 0 & 0 \\ 0 & 0 & \sigma_3 & 0 \\ 0 & 0 & 0 & \sigma_4 \\ \hline 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right]$$

The orthogonality of U, V may be expressed by $U^TU = UU^T = I_N$ and $V^TV = VV^T = I_M$. 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 $\mathbf{u}_i$, $i = 1, 2, \ldots, N$, and the columns of V by $\mathbf{v}_i$, $i = 1, 2, \ldots, M$, we may partition U, V in a compatible way as in Eq. (15.5.2):

$$\begin{aligned} U &= [\underbrace{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_r}_{U_r}, \underbrace{\mathbf{u}_{r+1}, \ldots, \mathbf{u}_N}_{\tilde{U}_r}] = [U_r \mid \tilde{U}_r] \\ V &= [\underbrace{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_r}_{V_r}, \underbrace{\mathbf{v}_{r+1}, \ldots, \mathbf{v}_M}_{\tilde{V}_r}] = [V_r \mid \tilde{V}_r] \end{aligned} \tag{15.5.5}$$

Then, Eq. (15.5.1) can be written in the form:

$$A = [U_r \mid \tilde{U}_r]\left[\begin{array}{c|c} \Sigma_r & 0 \\ \hline 0 & 0 \end{array}\right]\left[\begin{array}{c} V_r^T \\ \hline \tilde{V}_r^T \end{array}\right] = U_r\Sigma_rV_r^T \tag{15.5.6}$$

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:

$$\begin{aligned} &U_r^TU_r = I_r\,, \quad \tilde{U}_r^T\tilde{U}_r = I_{N-r}\,, \quad U_r^T\tilde{U}_r = 0\,, \quad U_rU_r^T + \tilde{U}_r\tilde{U}_r^T = I_N \\ &V_r^TV_r = I_r\,, \quad \tilde{V}_r^T\tilde{V}_r = I_{M-r}\,, \quad V_r^T\tilde{V}_r = 0\,, \quad V_rV_r^T + \tilde{V}_r\tilde{V}_r^T = I_M \end{aligned} \tag{15.5.8}$$

For example, we have:

$$U^TU = \left[\begin{array}{c|c} U_r^TU_r & U_r^T\tilde{U}_r \\ \hline \tilde{U}_r^TU_r & \tilde{U}_r^T\tilde{U}_r \end{array}\right] = \left[\begin{array}{c|c} I_r & 0 \\ \hline 0 & I_{N-r} \end{array}\right] = I_N\,, \quad U_rU_r^T + \tilde{U}_r\tilde{U}_r^T = UU^T = I_N$$

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 $\mathbf{u}_i, \mathbf{v}_i$:

$$\begin{array}{l} AV_r = U_r\Sigma_r \\ A\tilde{V}_r = 0 \\ A^T U_r = V_r\Sigma_r \\ A^T\tilde{U}_r = 0 \end{array} \quad\Leftrightarrow\quad \begin{array}{ll} A\mathbf{v}_i = \sigma_i\mathbf{u}_i\,, & i = 1,2,\ldots,r \\ A\mathbf{v}_i = 0\,, & i = r+1,\ldots,M \\ A^T\mathbf{u}_i = \sigma_i\mathbf{v}_i\,, & i = 1,2,\ldots,r \\ A^T\mathbf{u}_i = 0\,, & i = r+1,\ldots,N \end{array} \tag{15.5.10}$$

These equations show that $\mathbf{u}_i$ and $\mathbf{v}_i$, $i = 1,2,\ldots,r$, lie in the range spaces $R(A)$ and $R(A^T)$, respectively. Moreover, they provide orthonormal bases for these two subspaces. Similarly, $\mathbf{v}_i$, $i = r+1,\ldots,M$, and $\mathbf{u}_i$, $i = r+1,\ldots,N$, 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 $\lambda_i = \sigma_i^2$, $i = 1,2,\ldots,r$. 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:

$$A^TA = V\Lambda V^T\,, \quad \Lambda = \text{diag}(\lambda_1,\lambda_2,\ldots,\lambda_M) \in \mathbb{R}^{M\times M}$$

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 $\sigma_i = \sqrt{\lambda_i}$, $i = 1,2,\ldots,M$, and $\Sigma_1 = \Lambda^{1/2} = \text{diag}(\sigma_1,\ldots,\sigma_M) \in \mathbb{R}^{M\times M}$. Then, we define $U_1 = AV\Sigma_1^{-1}$, which is an $N\times M$ matrix with orthonormal columns:

$$U_1^TU_1 = \Sigma_1^{-1}V^T(A^TA)V\Sigma_1^{-1} = \Sigma_1^{-1}V^T(V\Sigma_1^2V^T)V\Sigma_1^{-1} = I_M$$

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 $a = \frac{1}{\sqrt{3}}$, $\beta = b = \frac{1}{\sqrt{2}}$, $\gamma = c = \frac{1}{\sqrt{6}}$, $\alpha = \frac{1}{\sqrt{3(3+\epsilon^2)}}$, and $\delta = \frac{1}{\sqrt{(3+\epsilon^2)}}$. □

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:

$$\begin{aligned} \sigma_i = \max_{\|\mathbf{u}\|=1} \max_{\|\mathbf{v}\|=1} \mathbf{u}^\dagger A\mathbf{v} = \mathbf{u}_i^\dagger A\mathbf{v}_i\,, \quad i = 2, \ldots, r \\ \text{subject to the constraints:} \quad \mathbf{u}^\dagger\mathbf{u}_j = \mathbf{v}^\dagger\mathbf{v}_j = 0\,, \quad j = 1, 2, \ldots, i-1 \end{aligned} \tag{15.5.18}$$

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:

$$\Sigma = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_M) \quad \Rightarrow \quad \Sigma^+ = \text{diag}(\sigma_1^+, \sigma_2^+, \ldots, \sigma_M^+) \tag{15.6.3}$$

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]:

$$\begin{aligned} AA^+A &= A\,, & (AA^+)^T &= AA^+ \\ A^+AA^+ &= A^+\,, & (A^+A)^T &= A^+A \end{aligned} \tag{15.6.8}$$

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,

$$(A^TA)^{-1}A^T = \left(V_r \Sigma_r^{-2} V_r^T\right)\left(V_r \Sigma_r U_r^T\right) = V_r \Sigma_r^{-1} U_r^T = A^+$$

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,

$$\mathbf{x}_\perp = \sum_{i=r+1}^{M} z_i \mathbf{v}_i = [\mathbf{v}_{r+1}, \mathbf{v}_{r+2}, \ldots, \mathbf{v}_M] \begin{bmatrix} z_{r+1} \\ z_{r+2} \\ \vdots \\ z_M \end{bmatrix} = \tilde{V}_r \mathbf{z}$$

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:

$$Ad\mathbf{x} + (dA)\mathbf{x} = d\mathbf{b} \quad \Rightarrow \quad d\mathbf{x} = A^{-1}[d\mathbf{b} - (dA)\mathbf{x}]$$

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:

$$A_k = \sum_{i=1}^{k} \sigma_i \mathbf{u}_i \mathbf{v}_i^T = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^T + \sigma_2 \mathbf{u}_2 \mathbf{v}_2^T + \cdots \sigma_k \mathbf{u}_k \mathbf{v}_k^T, \quad k = 1, 2, \ldots, r \tag{15.9.2}$$

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:

$$A_k = U \left[\begin{array}{c|c} \Sigma_k & 0 \\ \hline 0 & 0 \end{array}\right] V^T, \quad \Sigma_k = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_k, \underbrace{0, \ldots, 0}_{r-k \text{ zeros}}) \in \mathbb{R}^{r\times r} \tag{15.9.3}$$

Thus, A and A_k agree in their highest k singular values, but the last $r-k$ singular values of A, that is, $\sigma_{k+1}, \ldots, \sigma_r$, 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.

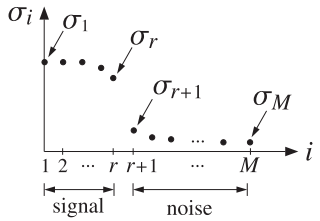


Fig. 15.9.1 Signal subspace vs. noise subspace singular values.

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 $i = r+1, \ldots, M$, 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:

$$\sigma_i = [1.0000,\ 1.0000,\ 0.0000,\ 0.0000]$$
$$\hat{\sigma}_i = [1.0004,\ 0.9984,\ 0.0012,\ 0.0005]$$

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 $\Sigma_r = \text{diag}(\sigma_1, \ldots, \sigma_r)$ and $\tilde{\Sigma}_r = \text{diag}(\sigma_{r+1}, \ldots, \sigma_M)$, 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.

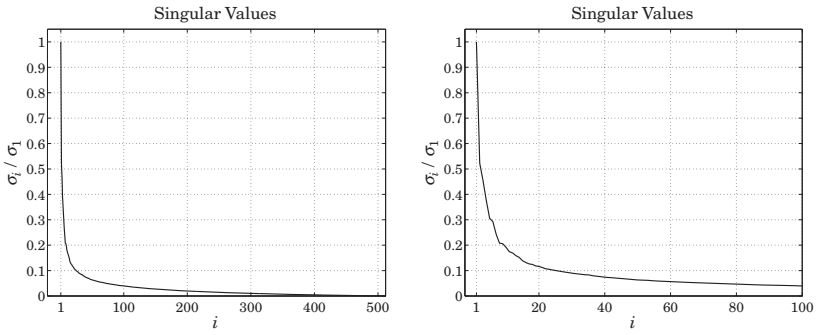


Fig. 15.9.2 Singular values of 512×512 image, with expanded view of first 100 values.

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:

$$\mathbf{x} = (A^TA+\lambda I)^{-1}A^T\mathbf{b} = \left[(A^TA)(A^TA+\lambda I)^{-1}\right](A^TA)^{-1}A^T\mathbf{b} = f(A)A^+\mathbf{b}$$

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:

$$\boxed{\mathcal{J} = \|\mathbf{b}-A\mathbf{x}\|^2 + \lambda\|D\mathbf{x}\|^2 = \min} \quad\Rightarrow\quad \boxed{\mathbf{x} = \left(A^TA+\lambda D^TD\right)^{-1}A^T\mathbf{b}} \tag{15.10.5}$$

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, $\mathbf{x} \in \mathbb{R}^M$,

$$\begin{aligned} (L_0)\text{:} \quad & \mathcal{J} = \|\mathbf{b} - A\mathbf{x}\|_2^2 + \lambda\, \|\mathbf{x}\|_0 = \min \\ (L_1)\text{:} \quad & \mathcal{J} = \|\mathbf{b} - A\mathbf{x}\|_2^2 + \lambda\, \|\mathbf{x}\|_1 = \min \\ (L_2)\text{:} \quad & \mathcal{J} = \|\mathbf{b} - A\mathbf{x}\|_2^2 + \lambda\, \|\mathbf{x}\|_2^2 = \min \end{aligned} \tag{15.11.2}$$

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 $\mathbf{x} \in \mathbb{R}^M$,

$$\begin{aligned} \|\mathbf{x}\|_p^p &= \sum_{i=1}^{M} |x_i|^p \approx \sum_{i=1}^{M} \frac{|x_i|^2}{|x_i|^q + \varepsilon} = \mathbf{x}^T W(\mathbf{x})\, \mathbf{x} \\ W(\mathbf{x}) &= \operatorname{diag}\left[\frac{1}{|\mathbf{x}|^q + \varepsilon}\right] = \operatorname{diag}\left[\frac{1}{|x_1|^q + \varepsilon}\,,\ \frac{1}{|x_2|^q + \varepsilon}\,,\ \ldots,\ \frac{1}{|x_M|^q + \varepsilon}\right] \end{aligned} \tag{15.11.3}$$

Alternatively, one can define $W(\mathbf{x})$ as the pseudo-inverse of the diagonal matrix of the powers $|x_i|^q$, $i = 1, 2, \ldots, M$, that is, in MATLAB language,[†]

$$W(\mathbf{x}) = \operatorname{pinv}\left(\operatorname{diag}\left[|x_1|^q\,,\ |x_2|^q\,,\ \ldots,\ |x_M|^q\right]\right) \tag{15.11.4}$$

Then, the ℓ_p-regularized least-squares problem can be written in the form,

$$\mathcal{J} = \|\mathbf{b} - A\mathbf{x}\|_2^2 + \lambda\, \|\mathbf{x}\|_p^p = \|\mathbf{b} - A\mathbf{x}\|_2^2 + \lambda\, \mathbf{x}^T W(\mathbf{x})\, \mathbf{x} = \min \tag{15.11.5}$$

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,

$$\boxed{\begin{aligned} &\text{for } k = 1, 2, \ldots, K, \text{ do:} \\ &\quad W_{k-1} = W(\mathbf{x}^{(k-1)}) \\ &\quad \mathbf{x}^{(k)} = \arg\min_{\mathbf{x}} \|\mathbf{b} - A\mathbf{x}\|_2^2 + \lambda\, \mathbf{x}^T W_{k-1} \mathbf{x} \end{aligned}} \quad \text{(IRLS)} \tag{15.11.6}$$

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,

$$\mathcal{J} = \|\mathbf{b} - A\mathbf{x}\|_2^2 + \lambda \|D\mathbf{x}\|_p^p = \|\mathbf{b} - A\mathbf{x}\|_2^2 + \lambda\, \mathbf{x}^T D^T W(D\mathbf{x})\, D\mathbf{x} = \min \tag{15.11.7}$$

which leads to to the following iterative algorithm,

$$\boxed{\begin{aligned} &\text{for } k = 1, 2, \ldots, K, \text{ do:} \\ &\quad W_{k-1} = W(D\mathbf{x}^{(k-1)}) \\ &\quad \mathbf{x}^{(k)} = \arg\min_{\mathbf{x}} \|\mathbf{b} - A\mathbf{x}\|_2^2 + \lambda\, \mathbf{x}^T D^T W_{k-1} D\mathbf{x} \end{aligned}} \quad \text{(IRLS)} \tag{15.11.8}$$

[†]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:

$$\mathbf{x}^{(0)} = (A^T A + \lambda D^T D)^{-1} A^T \mathbf{b}$$

The solution of the optimization problem at the kth step is:

$$\mathbf{x}^{(k)} = (A^T A + \lambda D^T W_{k-1} D)^{-1} A^T \mathbf{b}$$

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:

$$h_n = \cos\big(0.15(n - n_0)\big) \exp\big(-0.004(n - n_0)^2\big), \quad n = 0, 1, \ldots, M$$

where $M = 53$ and $n_0 = 25$. The input is a sparse spike train consisting of S spikes:

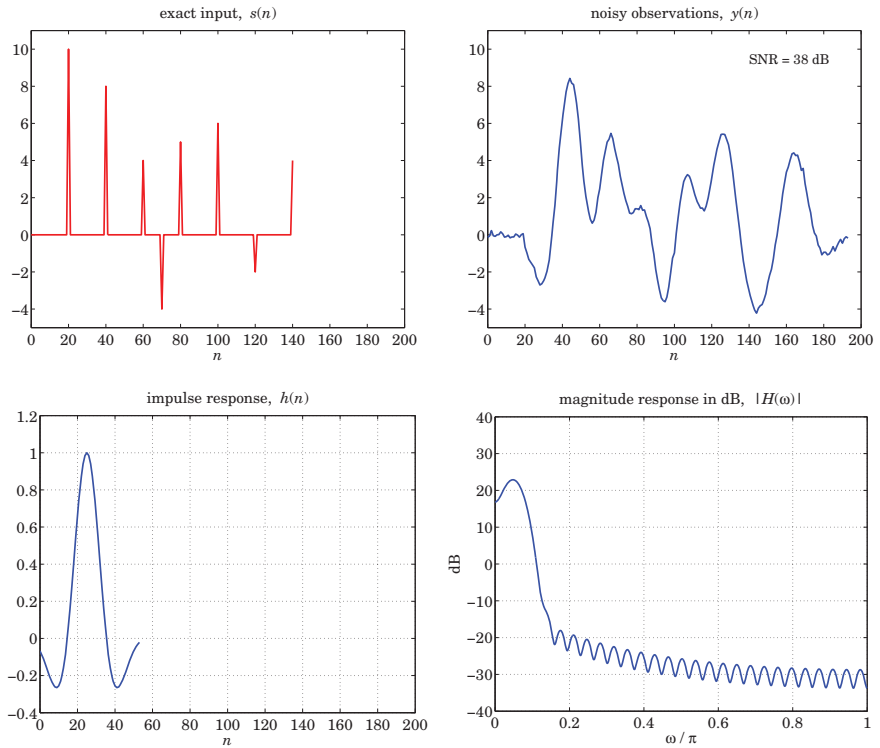
$$s_n = \sum_{i=1}^{S} a_i \delta(n - n_i), \quad n = 0, 1, \ldots, L-1 \tag{15.11.9}$$

where $S = 8$ and the spike locations and amplitudes are given as follows:

$$n_i = [20, 40, 60, 70, 80, 100, 120, 140], \quad a_i = [10, 8, 4, -4, 5, 6, -2, 4]$$

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, $SNR = 20\log_{10}(\max|H\mathbf{s}| / \sigma_v) = 38$.

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.

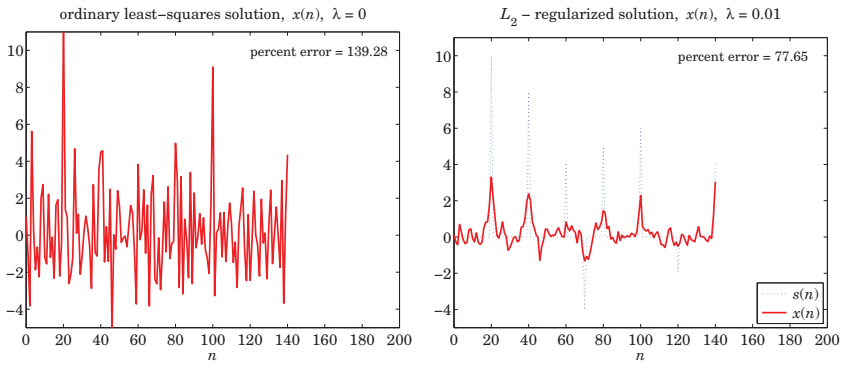


We note that $H(\omega)$ occupies a low frequency band, thus, we expect the effective deconvolution inverse filtering operation by $1/H(\omega)$ to be very sensitive to even small amounts of noise in y_n, even though the noise is barely visible in y_n itself. The three criteria of Eq. (15.11.2) to be 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, $\mathbf{x}_{\text{ord}} = (H^T H)^{-1} H^T \mathbf{y}$, or, $\mathbf{x}_{\text{ord}} = H \backslash \mathbf{y}$, in MATLAB.

Similarly, the ℓ_2-regularized solution with non-zero λ is, $\mathbf{x}_2 = (\lambda I + H^T H)^{-1} H^T \mathbf{y}$. 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, $\lambda = 0.1$, $p = 1$, $q = 1$, $\varepsilon = 10^{-5}$, and $K = 100$ iterations.

The ℓ_0 case was computed with the IRLS algorithm using parameters, $\lambda = 0.1$, $p = 0$, $q = 2$, $\varepsilon = 10^{-5}$, 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, $P(k) = 100 \cdot \|\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\|_2 / \|\mathbf{x}^{(k-1)}\|_2$.

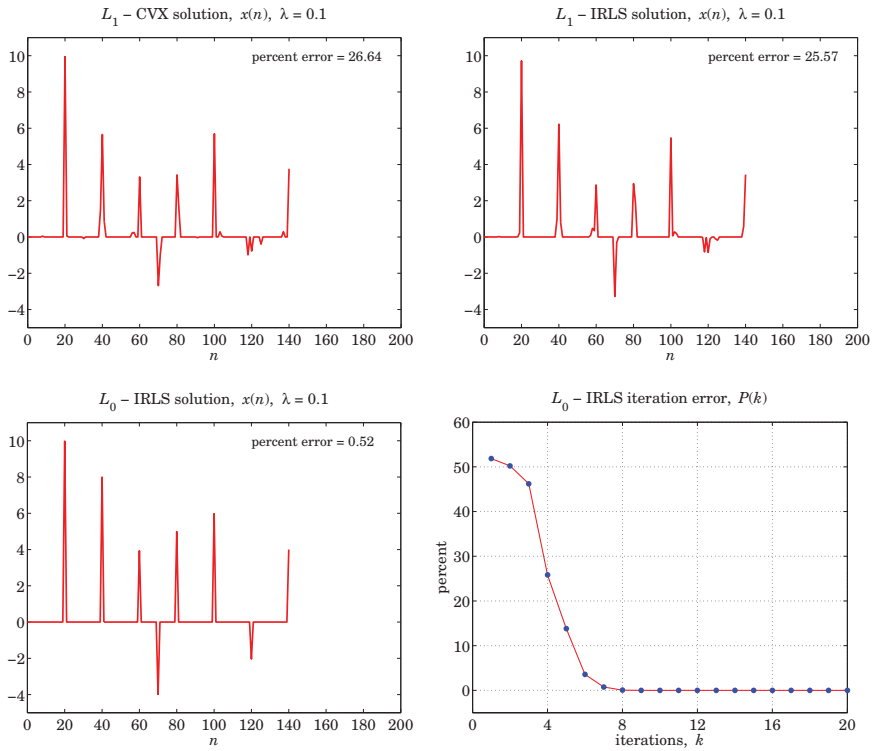


Fig. 15.11.1 Deconvolved signals based on the ℓ_1 and ℓ_0 criteria.

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 $\lambda = 0.1$, $\varepsilon = 10^{-6}$, 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
```

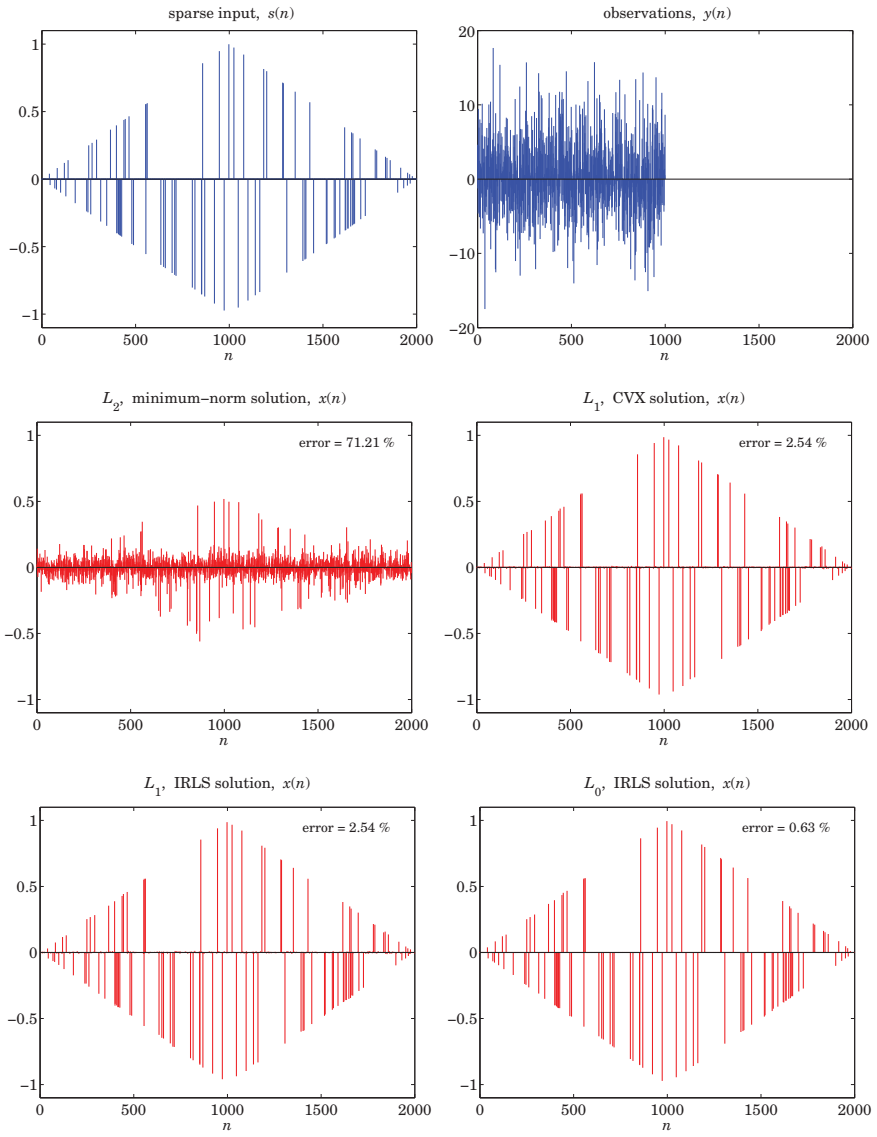



Fig. 15.11.2 Recovered signals based on the ℓ_2, ℓ_1, and ℓ_0 criteria.

```
Perr = 100 * norm(x-s)/norm(s);           % reconstruction error = 2.54 %

figure; stem(n,x,'r-');
```

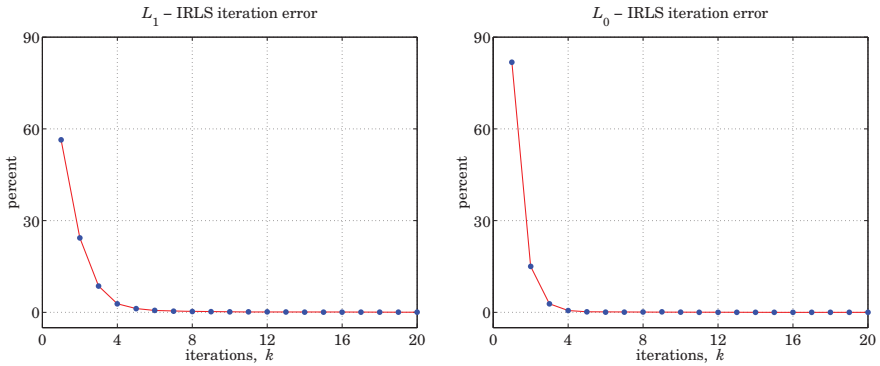


Fig. 15.11.3 IRLS iteration error based on the ℓ_1, and ℓ_0 criteria.

```
% ------------------------------------

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 $\{y_0(n), y_1(n), \ldots, y_M(n)\}$ 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:

$$\hat{x}(n) = \mathbf{h}^T\mathbf{y}(n) = [h_0, h_1, \ldots, h_M]\begin{bmatrix} y_0(n) \\ y_1(n) \\ \vdots \\ y_M(n) \end{bmatrix} = \sum_{m=0}^{M} h_m y_m(n) \tag{15.12.2}$$

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.

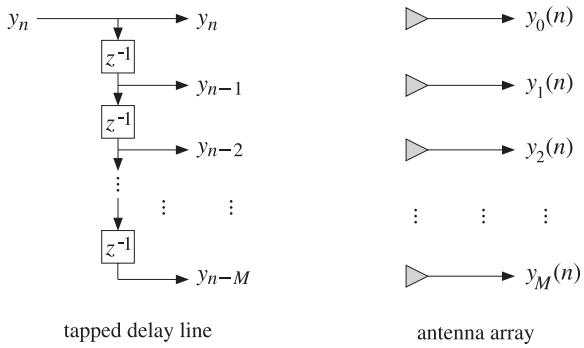


Fig. 15.12.1 Time-series processing versus spatial processing

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, $R_{ij}=E[y_i^*(n)y_j(n)]$, $i,j,=0,1,\ldots,M$.

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 $\mathbf{h}=[h_0,h_1,\ldots,h_M]^T$ 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 $\mathbf{y}(n)$, $n=0,1,\ldots,N-1$, 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 $\mathbf{y}_i=[y_i(0),\ldots,y_i(n),\ldots,y_i(N-1)]^T$. Therefore,

$$Y=[\mathbf{y}_0,\mathbf{y}_1,\ldots,\mathbf{y}_M]=\begin{bmatrix} \mathbf{y}^T(0) \\ \vdots \\ \mathbf{y}^T(n) \\ \vdots \\ \mathbf{y}^T(N-1) \end{bmatrix} \tag{15.12.10}$$

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 $Y^\dagger=Y^{*T}=[\mathbf{y}^*(0),\mathbf{y}^*(1),\ldots,\mathbf{y}^*(N-1)]$, 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:

$$\hat{R}=\sum_{n=0}^{N-1}\mathbf{y}^*(n)\mathbf{y}^T(n)=[\mathbf{y}^*(0),\mathbf{y}^*(1),\ldots,\mathbf{y}^*(N-1)]\begin{bmatrix} \mathbf{y}^T(0) \\ \mathbf{y}^T(1) \\ \vdots \\ \mathbf{y}^T(N-1) \end{bmatrix}=Y^\dagger Y$$

$$\hat{\mathbf{r}}=\sum_{n=0}^{N-1}\mathbf{y}^*(n)x(n)=[\mathbf{y}^*(0),\mathbf{y}^*(1),\ldots,\mathbf{y}^*(N-1)]\begin{bmatrix} x(0) \\ x(1) \\ \vdots \\ x(N-1) \end{bmatrix}=Y^\dagger\mathbf{x}$$

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:

$$\mathcal{E} = E[|e(n)|^2] = \min \quad \Rightarrow \quad \hat{\mathcal{E}} = \sum_{n=0}^{N-1} |e(n)|^2 = \mathbf{e}^\dagger\mathbf{e} = \|\mathbf{x} - Y\mathbf{h}\|^2 = \min \tag{15.12.14}$$

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, $n = 0, 1, \ldots, N-1$, 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, $\hat{x}(n)$, $n = M, \ldots, N-1$, 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:

$$Y_{\text{on}} = \begin{bmatrix} \mathbf{y}^T(0) \\ \vdots \\ \mathbf{y}^T(M-1) \end{bmatrix}, \quad Y_{\text{off}} = \begin{bmatrix} \mathbf{y}^T(N) \\ \vdots \\ \mathbf{y}^T(N-1+M) \end{bmatrix}$$

Then, we define Y for the autocorrelation and covariance cases:

$$Y_{\text{aut}} = \left[\begin{array}{c} \mathbf{y}^T(0) \\ \vdots \\ \mathbf{y}^T(M-1) \\ \hline \mathbf{y}^T(M) \\ \vdots \\ \mathbf{y}^T(N-1) \\ \hline \mathbf{y}^T(N) \\ \vdots \\ \mathbf{y}^T(N-1+M) \end{array}\right] = \left[\begin{array}{c} Y_{\text{on}} \\ \hline Y_{\text{cov}} \\ \hline Y_{\text{off}} \end{array}\right], \quad Y_{\text{cov}} = \begin{bmatrix} \mathbf{y}^T(M) \\ \vdots \\ \mathbf{y}^T(N-1) \end{bmatrix} \tag{15.12.20}$$

To clarify these expressions, consider an example where $N = 6$ and $M = 2$. The observation sequence is y_n, $n = 0, 1, \ldots, 5$. 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:

$$e_+(n) = [y_n, y_{n-1}, \ldots, y_{n-M}] \begin{bmatrix} 1 \\ a_1 \\ \vdots \\ a_M \end{bmatrix} = \mathbf{y}^T(n)\mathbf{a} \tag{15.13.1}$$

$$e_-(n) = [y_n, y_{n-1}, \ldots, y_{n-M}] \begin{bmatrix} a_M^* \\ \vdots \\ a_1^* \\ 1 \end{bmatrix} = \mathbf{y}^T(n)\mathbf{a}^{R*} \tag{15.13.2}$$

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:

$$\begin{aligned} \hat{\mathcal{E}}_{\text{aut}} &= \sum_{n=0}^{N-1+M} |e_+(n)|^2 = \min \\ \hat{\mathcal{E}}_{\text{cov}} &= \sum_{n=M}^{N-1} |e_+(n)|^2 = \min \\ \hat{\mathcal{E}}_{\text{fb}} &= \sum_{n=M}^{N-1} \left[|e_+(n)|^2 + |e_-(n)|^2\right] = \min \end{aligned} \tag{15.13.3}$$

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, $n = 0, 1, \ldots, N-1$. 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 $\text{MA}(q)$, 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:

$$B(z) = b_0 + b_1z^{-1} + b_2z^{-2} + \cdots + b_qz^{-q}, \quad S_{yy}(\omega) = \sigma_\epsilon^2\,|B(\omega)|^2 \tag{15.14.2}$$

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 $\mathbf{a} = [1, a_1, \ldots, a_M]^T$ are found by applying any least-squares LP method to the given sequence $\mathbf{y} = [y_0, y_1, \ldots, y_{N-1}]^T$, including Burg's method.

Finally, the desired MA filter $\mathbf{b} = [1, b_1, \ldots, b_q]^T$ is obtained by designing an order-q least-squares inverse to $\mathbf{a} = [1, a_1, \ldots, a_M]^T$ 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 $\mathbf{u} = [1, 0, \ldots, 0]^T$ 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 $A_{\rm bot}$ and the bottom part of $\mathbf{b}$:

$$A_{\rm bot} = [\mathbf{a}_1, A_1], \quad \mathbf{b} = \begin{bmatrix} 1 \\ \boldsymbol{\beta} \end{bmatrix} \quad \Rightarrow \quad A_{\rm bot}\mathbf{b} = [\mathbf{a}_1, A_1]\begin{bmatrix} 1 \\ \boldsymbol{\beta} \end{bmatrix} = \mathbf{a}_1 + A_1\boldsymbol{\beta} = 0$$

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 $i, j = 0, 1, \ldots, q-1$, and R_{aa} is the sample autocorrelation of the filter $\mathbf{a}$:

$$R_{aa}(k) = \sum_{m=0}^{M-|k|} a_{m+|k|}^* a_m, \quad -M \le k \le M \tag{15.14.8}$$

In other words, as observed by Durbin, the MA filter $\mathbf{b}$ may obtained by fitting an AR(q) model to the AR filter $\mathbf{a}$ 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:

$$H(z) = \frac{B(z)}{A(z)} = \frac{1 + b_1 z^{-1} + \cdots + b_q z^{-q}}{1 + a_1 z^{-1} + \ldots + a_p z^{-p}} \tag{15.14.9}$$

The sequence y_n is generated by driving $H(z)$ by zero-mean white-noise ϵ_n:

$$y_n + a_1 y_{n-1} + \cdots + a_p y_{n-p} = \epsilon_n + b_1 \epsilon_{n-1} + \cdots + b_q \epsilon_{n-q} \tag{15.14.10}$$

The corresponding power spectrum of y_n is:

$$S_{yy}(\omega) = \sigma_\epsilon^2 |H(\omega)|^2 = \sigma_\epsilon^2 \left| \frac{B(\omega)}{A(\omega)} \right|^2 = \sigma_\epsilon^2 \left| \frac{1 + b_1 e^{-j\omega} + \cdots + b_q e^{-jq\omega}}{1 + a_1 e^{-j\omega} + \cdots + a_p e^{-jp\omega}} \right|^2$$

If the innovations sequence ϵ_n were known, then by considering Eq. (15.14.10) at successive time instants, say, $n = 0, 1, \ldots, N-1$, one could solve for the model parameters $\mathbf{a} = [1, a_1, \ldots, a_p]^T$ and $\mathbf{b} = [1, b_1, \ldots, b_q]^T$. To see how this might be done, we rewrite (15.14.10) vectorially in the form:

$$[y_n, y_{n-1}, \ldots, y_{n-p}] \begin{bmatrix} 1 \\ a_1 \\ \vdots \\ a_p \end{bmatrix} = [\epsilon_n, \epsilon_{n-1}, \ldots, \epsilon_{n-q}] \begin{bmatrix} 1 \\ b_1 \\ \vdots \\ b_q \end{bmatrix} \tag{15.14.11}$$

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 $n = 0, 1 \ldots, N-1$, 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 $\mathbf{y} = [y_0, y_1 \ldots, y_{N-1}]^T$ and $\mathbf{e} = [\epsilon_0, \epsilon_1 \ldots, \epsilon_{N-1}]^T$ by the OSP function **datamat**:

$$\begin{aligned} Y &= \text{datamat}(\mathbf{y}, p, \texttt{'pre'}) \\ E &= \text{datamat}(\mathbf{e}, q, \texttt{'pre'}) \end{aligned}$$

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 $\mathbf{y} = [y_0, y_1 \ldots, y_{N-1}]^T$ is available.

A possible strategy to overcome this problem, originally proposed by Durbin, is to replace the unknown exact innovation vector $\mathbf{e} = [\epsilon_0, \epsilon_1 \ldots, \epsilon_{N-1}]^T$ by an estimated one $\hat{\mathbf{e}} = [\hat{\epsilon}_0, \hat{\epsilon}_1 \ldots, \hat{\epsilon}_{N-1}]^T$ 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:

$$\boxed{\begin{array}{l} \mathbf{a}_M = \text{lpf}(\text{yw}(\mathbf{y}, M)) \\ \hat{\mathbf{e}} = \text{filter}(\mathbf{a}_M, 1, \mathbf{y}) \\ \hat{E} = \text{datamat}(\hat{\mathbf{e}}, q, \texttt{'pre'}) \end{array}} \tag{15.14.16}$$

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:

$$[Y_1, -\hat{E}_1]\begin{bmatrix} \hat{\boldsymbol{\alpha}} \\ \hat{\boldsymbol{\beta}} \end{bmatrix} = -(\mathbf{y}_0 - \hat{\mathbf{e}}_0) \quad \Rightarrow \quad \begin{bmatrix} \hat{\boldsymbol{\alpha}} \\ \hat{\boldsymbol{\beta}} \end{bmatrix} = -[Y_1, -\hat{E}_1] \backslash (\mathbf{y}_0 - \hat{\mathbf{e}}_0) \tag{15.14.17}$$

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, $\mathbf{y}_0 = \mathbf{y} = [y_0, y_1 \ldots, y_{N-1}]^T$ and $\hat{\mathbf{e}}_0 = \hat{\mathbf{e}} = [\hat{\epsilon}_0, \hat{\epsilon}_1 \ldots, \hat{\epsilon}_{N-1}]^T$. 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:

$$\begin{bmatrix} \hat{\boldsymbol{\alpha}} \\ \hat{\boldsymbol{\beta}} \end{bmatrix} = -[Y_1, -\hat{E}_1] \backslash \mathbf{y} \tag{15.14.19}$$

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:

$$\boxed{\begin{array}{ll} \mathbf{v} = \text{filter}(1, \hat{\mathbf{b}}, \mathbf{y}); & V = \text{datamat}(\mathbf{v}, p, \texttt{'pre'}); \\ \mathbf{w} = \text{filter}(1, \hat{\mathbf{b}}, \hat{\mathbf{e}}); & W = \text{datamat}(\mathbf{w}, q, \texttt{'pre'}); \end{array}} \tag{15.14.20}$$

The resulting least-squares problem is then:

$$V\mathbf{a} = W\mathbf{b} \quad \Rightarrow \quad [\mathbf{v}_0, V_1]\begin{bmatrix} 1 \\ \boldsymbol{\alpha} \end{bmatrix} = [w_0, W_1]\begin{bmatrix} 1 \\ \boldsymbol{\beta} \end{bmatrix} \tag{15.14.21}$$

with performance index $\mathcal{J} = \|V\mathbf{a} - W\mathbf{b}\|^2 = \min$. The solution of (15.14.21) is:

$$\begin{bmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{bmatrix} = -[V_1, -W_1] \backslash (\mathbf{v}_0 - \mathbf{w}_0) \tag{15.14.22}$$

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 $\mathbf{y}(n) = [y_0(n), y_1(n), \ldots, y_M(n)]^T$ 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,

$$V = [\mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_M], \quad R\mathbf{v}_i = \lambda_i\mathbf{v}_i, \quad i = 0, 1, \ldots, M \tag{15.15.2}$$

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:

$$RV = V\Lambda, \quad \Lambda = \text{diag}\{\lambda_0, \lambda_1, \ldots, \lambda_M\} \quad\Rightarrow\quad V^\dagger RV = \Lambda \tag{15.15.4}$$

The components of the transformed vector, $\mathbf{z}(n) = [z_0(n), z_1(n), \ldots, z_M(n)]^T$, are called *principal components*. They can be expressed as the dot products of the eigenvectors $\mathbf{v}_i$ with $\mathbf{y}(n)$:

$$\mathbf{z}(n) = V^T\mathbf{y}(n) \quad\Rightarrow\quad \begin{bmatrix} z_0(n) \\ z_1(n) \\ \vdots \\ z_M(n) \end{bmatrix} = \begin{bmatrix} \mathbf{v}_0^T\mathbf{y}(n) \\ \mathbf{v}_1^T\mathbf{y}(n) \\ \vdots \\ \mathbf{v}_M^T\mathbf{y}(n) \end{bmatrix}, \quad \text{or,}$$

$$\boxed{z_i(n) = \mathbf{v}_i^T\mathbf{y}(n)}, \quad i = 0, 1, \ldots, M \tag{15.15.5}$$

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:

$$\boxed{E[z_i^*(n)z_j(n)] = \lambda_i\delta_{ij}}, \quad i,j = 0, 1, \ldots, M \tag{15.15.7}$$

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 $\mathbf{y}^\dagger\mathbf{y} = \text{tr}(\mathbf{y}^*\mathbf{y}^T)$, we find:

$$\sigma_y^2 = \text{tr}(E[\mathbf{y}^*(n)\mathbf{y}^T(n)]) = \text{tr}(R) = \lambda_0 + \lambda_1 + \cdots + \lambda_M \tag{15.15.9}$$

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:

$$\mathbf{y}(n) = V^*\mathbf{z}(n) = [\mathbf{v}_0^*, \mathbf{v}_1^*, \ldots, \mathbf{v}_M^*]\begin{bmatrix} z_0(n) \\ z_1(n) \\ \vdots \\ z_M(n) \end{bmatrix} = \sum_{i=0}^{M} \mathbf{v}_i^* z_i(n) \tag{15.15.11}$$

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:

$$E[\|\mathbf{y}(n) - \hat{\mathbf{y}}(n)\|^2] = E[\sum_{i=r}^{M} |z_i(n)|^2] = \sum_{i=r}^{M} \lambda_i \tag{15.15.13}$$

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 $\mathbf{y}(n)$, $n = 0, 1, \ldots, N-1$:

$$\hat{R} = \frac{1}{N}\sum_{n=0}^{N-1} \mathbf{y}^*(n)\mathbf{y}^T(n) \tag{15.16.1}$$

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:

$$\mathbf{z}(n) = V^T\mathbf{y}(n), \quad n = 0, 1, \ldots, N-1 \tag{15.16.3}$$

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,

$$Y = ZV^\dagger \quad \Rightarrow \quad \mathbf{y}(n) = V^*\mathbf{z}(n), \quad n = 0, 1, \ldots, N-1 \tag{15.16.6}$$

or, explicitly in terms of the PCA signals $z_i(n)$:

$$\mathbf{y}(n) = \sum_{i=0}^{M} \mathbf{v}_i^* z_i(n), \quad n = 0, 1, \ldots, N-1 \tag{15.16.7}$$

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:

$$\boxed{\sum_{n=0}^{N-1} z_i^*(n)\, z_j(n) = \lambda_i \delta_{ij}}, \quad i,j = 0,1,\ldots,M \tag{15.16.9}$$

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 $i = 0,1,\ldots,M$, and the matrices:

$$U = Z\Sigma^{-1}, \quad \Sigma = \text{diag}\{\sigma_0, \sigma_1, \ldots, \sigma_M\} = \Lambda^{1/2} \tag{15.16.10}$$

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

$$\boxed{Y = U\Sigma V^\dagger} \qquad \text{(economy SVD)} \tag{15.16.12}$$

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 $\mathbf{u}(n) = \Sigma^{-1}\mathbf{z}(n) = \Sigma^{-1}V^T\mathbf{y}(n)$, $n = 0,1,\ldots,N-1$, have U as their data matrix and correspond to normalized versions of the principal components with unit sample covariance matrix:

$$U^\dagger U = \sum_{n=0}^{N-1} \mathbf{u}^*(n)\mathbf{u}^T(n) = I_{M+1} \quad \Leftrightarrow \quad \sum_{n=0}^{N-1} u_i^*(n)\, u_j(n) = \delta_{ij}$$

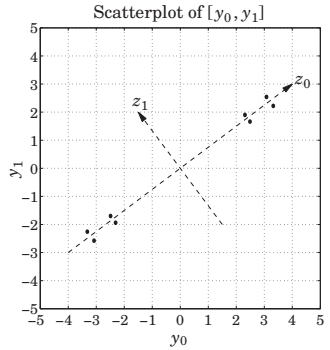
where $u_i(n)$ is the ith component of $\mathbf{u}(n) = [u_0(n), u_1(n), \ldots, u_M(n)]^T$. It is the same as $z_i(n)$, but normalized to unit norm.

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:

$$\frac{\sigma_0^2}{\sigma_0^2+\sigma_1^2} = 0.9975 = 99.75\%, \quad \frac{\sigma_1^2}{\sigma_0^2+\sigma_1^2} = 0.0025 = 0.25\%$$

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

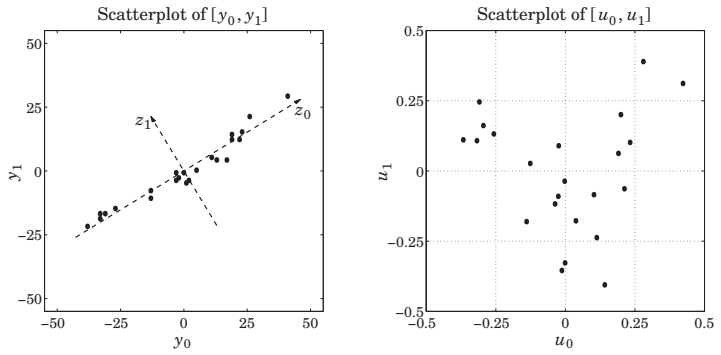


Fig. 15.16.1 Scatterplots of original data and their principal components.

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:

$$\frac{\sigma_0^2}{\sigma_0^2+\sigma_1^2} = 0.989 = 98.9\%, \quad \frac{\sigma_1^2}{\sigma_0^2+\sigma_1^2} = 0.011 = 1.1\%$$

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,

$$z_0 = \mathbf{v}_0^T\mathbf{y} = 0.8542\, y_0 + 0.52\, y_1$$
$$z_1 = \mathbf{v}_1^T\mathbf{y} = -0.52\, y_0 + 0.8542\, y_1$$

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 $u_0(n), u_1(n)$, $n = 0, 1, \ldots, N-1$. 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 rank 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:

$$y_n = \cos(2\pi f_1 n) + \cos(2\pi f_2 n) + 0.707\,v_n\,, \quad n = 0, 1, \ldots, 24$$

[†]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.

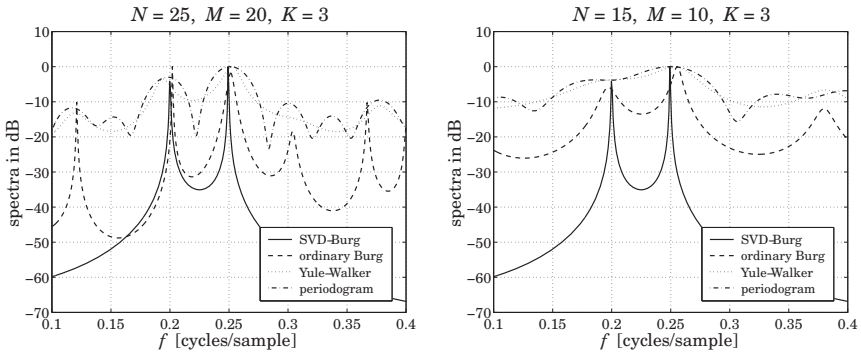


Fig. 15.17.1 SVD-enhanced linear prediction spectra.

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:

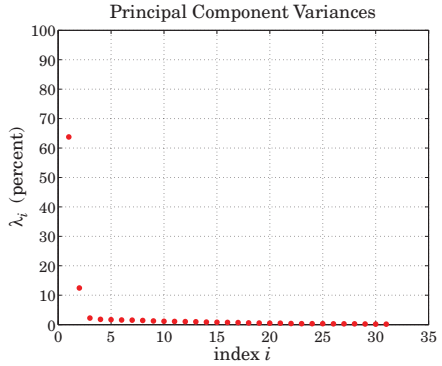


Fig. 15.17.2 Percentage variances of the first 31 principal components.

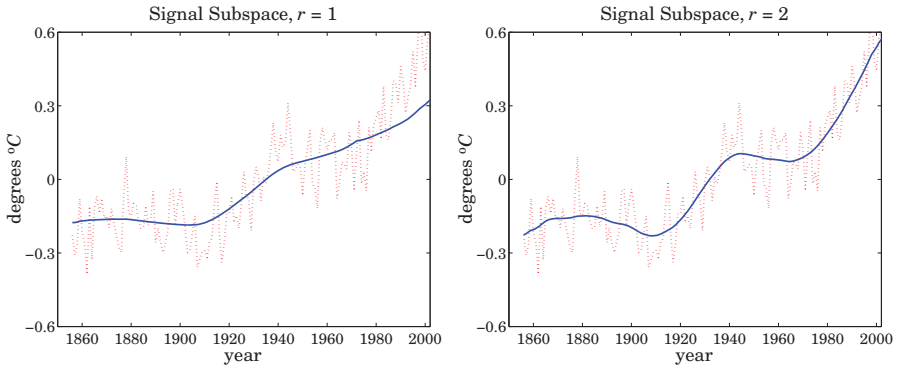


Fig. 15.17.3 Principal component signals of ranks $r = 1, 2$.

```
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

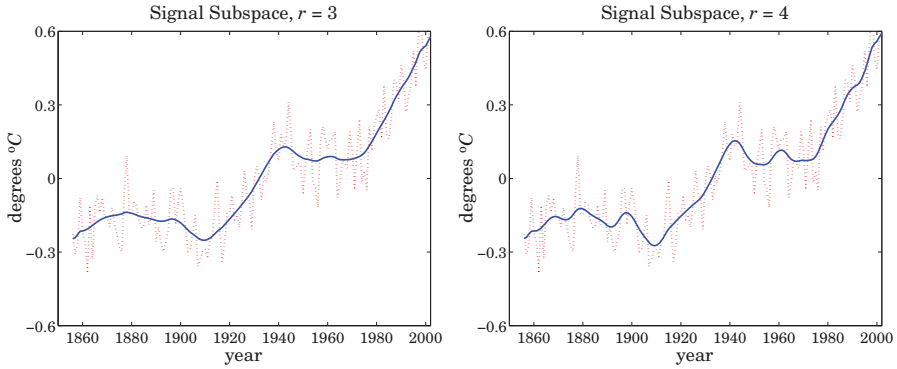


Fig. 15.17.4 Principal component signals of ranks $r = 3, 4$.

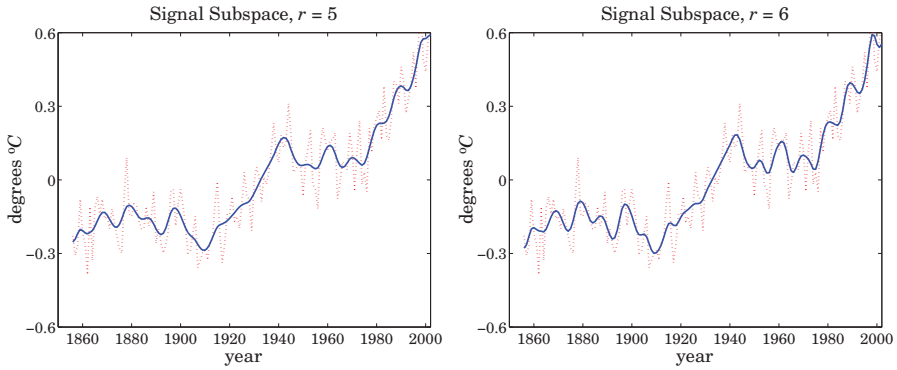


Fig. 15.17.5 Principal component signals of ranks $r = 5, 6$.

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.

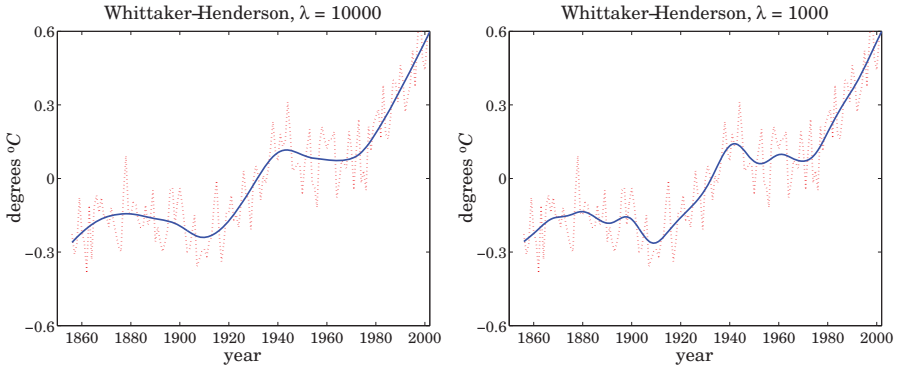


Fig. 15.17.6 Whittaker-Henderson smoothing method.

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:

$$y_n = \sum_{i=1}^{r} A_i e^{-\alpha_i n} e^{j\omega_i n} + v_n = \sum_{i=1}^{r} A_i z_i^n + v_n \tag{15.19.2}$$

where $z_i = e^{-\alpha_i + j\omega_i}$, $i = 1, 2, \ldots, r$. 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 $\mathbf{y} = [y_0, y_1, \ldots, y_{N-1}]^T$ 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:

$$Y = \sum_{i=1}^{r} A_i \begin{bmatrix} z_i^M \\ \vdots \\ z_i^n \\ \vdots \\ z_i^{N-1} \end{bmatrix} [1, z_i^{-1}, z_i^{-2}, \ldots, z_i^{-M}] \tag{15.19.4}$$

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 $i = 1, 2 \ldots, r$, and the remaining $M - r$ roots are arbitrary. The polynomial $B(z)$ defines an $(M + 1)$-dimensional vector $\mathbf{b} = [b_0, b_1, \ldots, b_M]^T$ through its inverse z-transform:

$$B(z) = b_0 + b_1 z^{-1} + \cdots + b_M z^{-M} = [1, z^{-1}, z^{-2}, \ldots, z^{-M}] \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_M \end{bmatrix} \tag{15.19.6}$$

Then, the root conditions can be stated in the form:

$$B(z_i) = [1, z_i^{-1}, z_i^{-2}, \ldots, z_i^{-M}] \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_M \end{bmatrix} = 0, \quad i = 1, 2, \ldots, r \tag{15.19.7}$$

This implies that the vector $\mathbf{b}$ must lie in the null space of Y:

$$Y\mathbf{b} = \sum_{i=1}^{r} A_i \begin{bmatrix} z_i^M \\ \vdots \\ z_i^n \\ \vdots \\ z_i^{N-1} \end{bmatrix} [1, z_i^{-1}, z_i^{-2}, \ldots, z_i^{-M}] \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_M \end{bmatrix} = 0 \tag{15.19.8}$$

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 $[z_i^M, \ldots, z_i^n, \ldots, z_i^{N-1}]^T$ are linearly independent,[†]

[†]This follows from the fact that the $r \times r$ Vandermonde matrix $V_{ki} = z_i^{k-1}$, $k, i = 1, 2, \ldots, r$, 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].

and therefore, we must have:

$$Y\mathbf{b} = \sum_{i=1}^{r} A_i \begin{bmatrix} z_i^M \\ \vdots \\ z_i^n \\ \vdots \\ z_i^{N-1} \end{bmatrix} B(z_i) = 0 \quad \Rightarrow \quad B(z_i) = 0 \tag{15.19.9}$$

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,

$$\begin{aligned} Y_1 &= \sum_{i=1}^{r} A_i \begin{bmatrix} z_i^M \\ \vdots \\ z_i^n \\ \vdots \\ z_i^{N-1} \end{bmatrix} [1, z_i^{-1}, z_i^{-2}, \ldots, z_i^{-(M-1)}] \\ Y_0 &= \sum_{i=1}^{r} A_i \begin{bmatrix} z_i^M \\ \vdots \\ z_i^n \\ \vdots \\ z_i^{N-1} \end{bmatrix} [z_i^{-1}, z_i^{-2}, \ldots, z_i^{-M}] \end{aligned} \tag{15.19.10}$$

They were obtained by keeping the first or last M entries of $[1, z_i^{-1}, z_i^{-2}, \ldots, z_i^{-M}]$:

$$[\underbrace{1, z_i^{-1}, z_i^{-2}, \ldots, z_i^{-(M-1)}}_{\text{first } M}, z_i^{-M}] = [1, \underbrace{z_i^{-1}, z_i^{-2}, \ldots, z_i^{-(M-1)}, z_i^{-M}}_{\text{last } M}]$$

Both matrices Y_1, Y_0 have dimension $(N - M) \times M$. Noting that

$$[1, z_i^{-1}, z_i^{-2}, \ldots, z_i^{-(M-1)}] = z_i [z_i^{-1}, z_i^{-2}, \ldots, z_i^{-M}],$$

we may rewrite Y_1 in the form:

$$Y_1 = \sum_{i=1}^{r} z_i A_i \begin{bmatrix} z_i^M \\ \vdots \\ z_i^n \\ \vdots \\ z_i^{N-1} \end{bmatrix} [z_i^{-1}, z_i^{-2}, \ldots, z_i^{-M}] \tag{15.19.11}$$

Therefore, the matrix pencil $Y_1 - \lambda Y_0$ can be written as:

$$Y_1 - \lambda Y_0 = \sum_{i=1}^{r} (z_i - \lambda) A_i \begin{bmatrix} z_i^M \\ \vdots \\ z_i^n \\ \vdots \\ z_i^{N-1} \end{bmatrix} [z_i^{-1}, z_i^{-2}, \ldots, z_i^{-M}] \tag{15.19.12}$$

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 $\lambda \neq z_i$, 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:

$$Y = \begin{bmatrix} y_3 & y_2 & y_1 & y_0 \\ y_4 & y_3 & y_2 & y_1 \\ y_5 & y_4 & y_3 & y_2 \\ y_6 & y_5 & y_4 & y_3 \\ y_7 & y_6 & y_5 & y_4 \\ y_8 & y_7 & y_6 & y_5 \\ y_9 & y_8 & y_7 & y_6 \end{bmatrix} \quad \Rightarrow \quad Y_1 = \begin{bmatrix} y_3 & y_2 & y_1 \\ y_4 & y_3 & y_2 \\ y_5 & y_4 & y_3 \\ y_6 & y_5 & y_4 \\ y_7 & y_6 & y_5 \\ y_8 & y_7 & y_6 \\ y_9 & y_8 & y_7 \end{bmatrix}, \; Y_0 = \begin{bmatrix} y_2 & y_1 & y_0 \\ y_3 & y_2 & y_1 \\ y_4 & y_3 & y_2 \\ y_5 & y_4 & y_3 \\ y_6 & y_5 & y_4 \\ y_7 & y_6 & y_5 \\ y_8 & y_7 & y_6 \end{bmatrix}$$

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 $\mathbf{y} = [y_0, y_1, \ldots, y_M]^T$ is defined by:

$$\begin{aligned} &\epsilon_0 = y_0 \\ &\text{for } m = 1, 2, \ldots, M \text{ do:} \\ &\qquad \epsilon_m = y_m - \sum_{i=0}^{m-1} \frac{E[\epsilon_i^* y_m]}{E[\epsilon_i^* \epsilon_i]}\, \epsilon_i \end{aligned} \tag{15.20.1}$$

The constructed random vector $\boldsymbol{\epsilon} = [\epsilon_0, \epsilon_1 \ldots, \epsilon_M]^T$ has uncorrelated components $E[\epsilon_i^* \epsilon_j] = 0$, if $i \neq j$. 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 $D = E[\boldsymbol{\epsilon}^*\boldsymbol{\epsilon}^T] = \text{diag}\{E_0, E_1, \ldots, E_M\}$, 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 $q_i = \epsilon_i / E_i^{1/2}$, $i = 0, 1, \ldots, M$, normalized to have *unit variance*. Then, the random vector $\mathbf{q} = [q_0, q_1, \ldots, q_M]^T$ 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 $\mathbf{y}(n)$, $n = 0, 1, \ldots, N-1$. 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,

$$Y = QG \quad \Rightarrow \quad \begin{bmatrix} \mathbf{y}^T(0) \\ \vdots \\ \mathbf{y}^T(n) \\ \vdots \\ \mathbf{y}^T(N-1) \end{bmatrix} = \begin{bmatrix} \mathbf{q}^T(0) \\ \vdots \\ \mathbf{q}^T(n) \\ \vdots \\ \mathbf{q}^T(N-1) \end{bmatrix} G \quad \Rightarrow \quad \mathbf{y}^T(n) = \mathbf{q}^T(n)G, \quad \text{or,}$$

$$\mathbf{y}(n) = G^T\mathbf{q}(n), \quad n = 0, 1, \ldots N-1 \tag{15.20.10}$$

which is the same as Eq. (15.20.6).

Writing $\mathbf{q}^T(n) = [q_0(n), q_1(n), \ldots, q_M(n)]$, the ith column of Q is the time signal $q_i(n)$, $n = 0, 1, \ldots, N-1$. 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 $i, j, = 0, 1, \ldots, M$:

$$E[q_i^* q_j] = \delta_{ij} \quad \Rightarrow \quad \sum_{n=0}^{N-1} q_i^*(n) q_j(n) = \delta_{ij} \tag{15.20.12}$$

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
$R = E[\mathbf{y}^*\mathbf{y}^T] = V\Lambda V^\dagger$	$R = E[\mathbf{y}^*\mathbf{y}^T] = G^\dagger G = BDB^\dagger$
$\mathbf{y} = V^*\mathbf{z} = V^*\Sigma\mathbf{u}$	$\mathbf{y} = G^T\mathbf{q} = B^*\boldsymbol{\epsilon}$
$E[\mathbf{z}^*\mathbf{z}^T] = \Lambda = \Sigma^2$	$E[\boldsymbol{\epsilon}^*\boldsymbol{\epsilon}^T] = D$
$E[\mathbf{u}^*\mathbf{u}^T] = I$	$E[\mathbf{q}^*\mathbf{q}^T] = I$
SVD	**QR**
$Y = U\Sigma V^\dagger = ZV^\dagger$	$Y = QG$
$\hat{R} = Y^\dagger Y = V\Lambda V^\dagger = V\Sigma^2 V^\dagger$	$\hat{R} = Y^\dagger Y = G^\dagger G$
$\mathbf{y}(n) = V^*\mathbf{z}(n) = V^*\Sigma\mathbf{u}(n)$	$\mathbf{y}(n) = G^T\mathbf{q}(n) = B^*\boldsymbol{\epsilon}(n)$
$\sum_{n=0}^{N-1} \mathbf{u}^*(n)\mathbf{u}^T(n) = I$	$\sum_{n=0}^{N-1} \mathbf{q}^*(n)\mathbf{q}^T(n) = I$

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:

$$\mathbf{y} = \begin{bmatrix} \mathbf{y}_a \\ \hline \mathbf{y}_b \end{bmatrix}, \quad \text{where} \quad \mathbf{y}_a = \begin{bmatrix} y_{a1} \\ y_{a2} \\ \vdots \\ y_{ap} \end{bmatrix}, \quad \mathbf{y}_b = \begin{bmatrix} y_{b1} \\ y_{b2} \\ \vdots \\ y_{bq} \end{bmatrix} \tag{15.21.1}$$

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 $c_1 \geq c_2$, 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}:

$$C_{ab} = F_a C F_b^\dagger, \quad C = \text{diag}\{c_1, c_2 \ldots, c_r\} \in \mathbb{C}^{p \times q} \tag{15.21.11}$$

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:

$$\begin{aligned} A &= V_a \Sigma_a^{-1} F_a = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_p] = p \times p \text{ matrix} \\ B &= V_b \Sigma_b^{-1} F_b = [\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_q] = q \times q \text{ matrix} \end{aligned} \tag{15.21.12}$$

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:

$$c_i = E[w_{ai}^* w_{bi}], \quad w_{ai} = \mathbf{a}_i^T \mathbf{y}_a, \quad w_{bi} = \mathbf{b}_i^T \mathbf{y}_b, \quad i = 1, 2, \ldots, r \tag{15.21.16}$$

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 $\mathbf{y}_a(n), \mathbf{y}_b(n)$, $n = 0, 1, \ldots, N-1$. 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:

$$\begin{aligned} &Y_a = U_a \Sigma_a V_a^\dagger = \text{economy SVD of } Y_a, \quad U_a \in \mathbb{C}^{N\times p}, \quad U_a^\dagger U_a = I_p \\ &Y_b = U_b \Sigma_b V_b^\dagger = \text{economy SVD of } Y_b, \quad U_b \in \mathbb{C}^{N\times q}, \quad U_b^\dagger U_b = I_q \\ &C_{ab} = U_a^\dagger U_b = \text{cross-covariance in } u\text{-basis}, \quad C_{ab} \in \mathbb{C}^{p\times q} \\ &C_{ab} = F_a C F_b^\dagger = \text{full SVD}, \quad C = \text{diag}\{c_1, c_2 \ldots, c_r\}, \quad r = \min(p, q) \\ &A = V_a \Sigma_a^{-1} F_a = \text{CCA coefficients}, \quad A \in \mathbb{C}^{p\times p} \\ &B = V_b \Sigma_b^{-1} F_b = \text{CCA coefficients}, \quad B \in \mathbb{C}^{q\times q} \\ &W_a = Y_a A, \quad W_a \in \mathbb{C}^{N\times p} \text{ with orthonormal columns}, \quad W_a^\dagger W_a = I_p \\ &W_b = Y_b B, \quad W_b \in \mathbb{C}^{N\times q} \text{ with orthonormal columns}, \quad W_b^\dagger W_b = I_q \\ &W_a^\dagger W_b = C = p\times q \text{ diagonal matrix of canonical correlations} \end{aligned} \tag{15.21.19}$$

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 $w_{ai}(n), w_{bi}(n)$, $n = 0, 1, \ldots, N-1$, we note that they have unit norm as N-dimensional vectors, and the ith canonical correlation is given by the time-average:

$$c_i = \sum_{n=0}^{N-1} w_{ai}^*(n)\, w_{bi}(n)\,, \quad i = 1, 2, \ldots, r \tag{15.21.21}$$

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:

$$c_i = \cos\theta_i\,, \quad i = 1, 2 \ldots, r = \min(p, q) \tag{15.21.22}$$

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
```

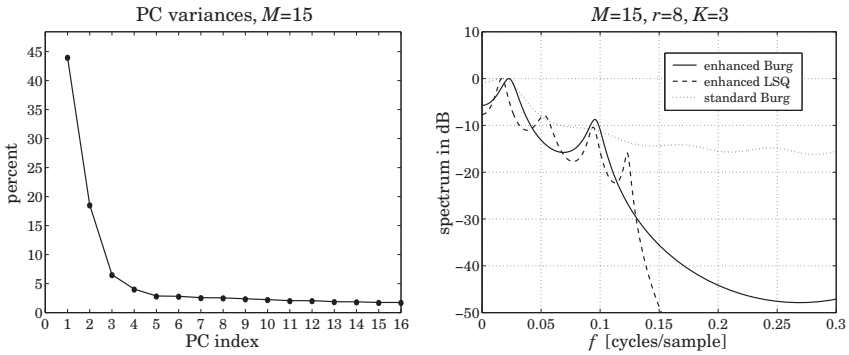
c. Using the *enhanced* data matrix Y_e, calculate the least-squares prediction error filter, $\mathbf{a}_{LS}$, by solving $Y_e\mathbf{a} = 0$.

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.

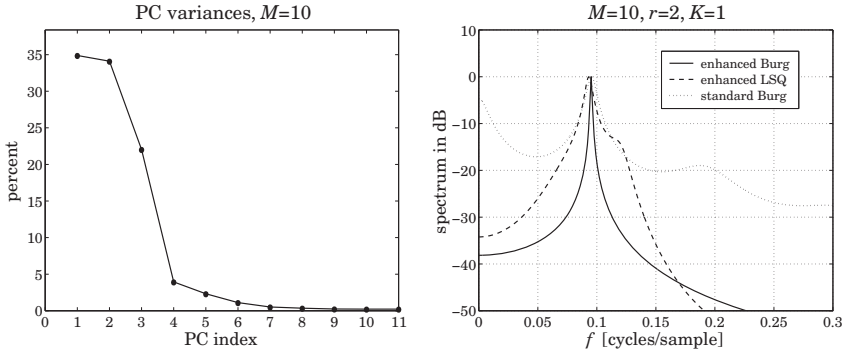


15.3 *Computer Experiment – Sunspot Numbers.* The Wolf sunspot numbers are of great historical importance in the development of spectral analysis methods (periodogram and parametric). Sunspot activity is cyclical and variation in the sunspot numbers has been correlated with weather and other terrestrial 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