9

Periodic Signal Extraction

Many physical, financial, and social time series have a natural periodicity in them, such as daily, monthly, quarterly, yearly. The observed signal can be regarded as having three components: a periodic (or nearly periodic) seasonal part s_n, a smooth trend t_n, and a residual irregular part v_n that typically represents noise,

$$y_n = s_n + t_n + v_n$$

The model can also be assumed to be multiplicative, $y_n = s_n t_n v_n$. The signal processing task is to extract both the trend and the seasonal components, t_n and s_n, from the observed signal y_n.

For example, many climatic signals, such as CO_2 emissions, are characterized by an annual periodicity. Government agencies routinely estimate and remove the seasonal component from business and financial data and only the "seasonally-adjusted" signal $a_n = t_n + v_n$ is available, such as the US GDP that we considered in Example 8.3.1. Further processing of the deseasonalized signal a_n, using for example a trend extraction filter such as the Hodrick-Prescott filter, can reveal additional information, such as business cycles.

Periodic signals appear also in many engineering applications. Some examples are: (a) Electrocardiogram recordings are subject to power frequency interference (e.g., 60 Hz and its higher harmonics) which must be removed by appropriate filters. (b) All biomedical signals require some sort of signal processing for their enhancement. Often weak biomedical signals, such as brain signals from visual responses or muscle signals, can be evoked periodically with the responses accumulated (averaged) to enhance their SNR; (c) TV video signals have two types of periodicities in them, one due to line-scanning and one due to the frame rate. In the pre-HDTV days, the chrominance (color) TV signals were put on a subcarrier signal and added to the luminance (black & white) signal, and the composite signal was then placed on another carrier for transmission. The subcarrier's frequency was chosen carefully so as to shift the line- and frame-harmonics of the chrominance signal relative to those of the luminance so that at the receiving end the two could be separated by appropriately designed comb filters [30]. (d) GPS signals contain a repetitive code word that repeats with a period of one millisecond. The use of comb filters can enhance their reception. (e) Radars send out repetitive pulses so that

the returns from slowly moving targets have a quasi-periodic character. By accumulating these return, the SNR can be enhanced. As we see below, signal averaging is a form of comb filtering.

In this chapter, we discuss the design of comb and notch filters for extracting periodic signals or canceling periodic interference. We discuss also the specialized comb filters, referred to as "seasonal filters," that are used by standard seasonal decomposition methods, such as the census X-11 method, and others.

9.1 Notch and Comb Filters for Periodic Signals

To get started, we begin with the signal plus interference model $y_n = s_n + v_n$ in which either the signal or the noise is periodic, but not both.

If the noise v_n is periodic, its spectrum will be concentrated at the harmonics of some fundamental frequency, say ω_1. The noise reduction filter must be an ideal *notch filter* with notches at the harmonics $k\omega_1$, $k = 0, 1, \ldots$, as shown in Fig. 9.1.1. If the filter notches are narrow, then the distortion of the desired signal s_n will be minimized.

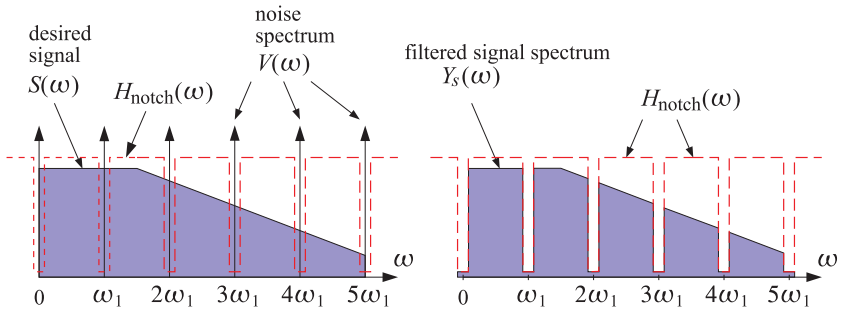


Fig. 9.1.1 Notch filter for reducing periodic interference.

On the other hand, if the desired signal s_n is periodic and the noise is a wideband signal, the signal enhancement filter for extracting s_n must be an ideal *comb filter* with peaks at the harmonics of the desired signal, as shown in Fig. 9.1.2. If the comb peaks are narrow, then only a minimal amount of noise will pass through the filter (that is, the portion of the noise whose power lies within the narrow peaks.)

A discrete-time periodic signal s_n with a period of D samples admits the following finite D-point DFT and inverse DFT representation [29] in terms of the D harmonics that lie within the Nyquist interval, $\omega_k = 2\pi k/D = k\omega_1$, for $k = 0, 1, \ldots, D-1$,

$$\begin{aligned} \text{(DFT)} \quad & S_k = \sum_{n=0}^{D-1} s_n e^{-j\omega_k n}, \quad k = 0, 1, \ldots, D-1 \\ \text{(IDFT)} \quad & s_n = \frac{1}{D} \sum_{k=0}^{D-1} S_k e^{j\omega_k n}, \quad n = 0, 1, \ldots, D-1 \end{aligned} \tag{9.1.1}$$

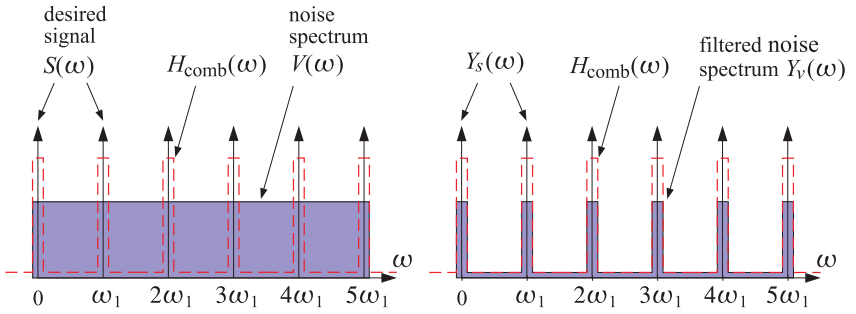


Fig. 9.1.2 Comb filter for enhancing a periodic signal.

where S_k is the D-point DFT of one period $[s_0, s_1, \ldots, s_{D-1}]$ of the time signal. Because of the periodicity, the IDFT formula is actually valid for all n in the interval $-\infty < n < \infty$.

We note that a periodic continuous-time signal $s(t)$ does not necessarily result into a periodic discrete-time signal when sampled at some arbitrary rate. For the sampled signal $s_n = s(nT)$ to be periodic in n with a period of D samples, where T is the sampling interval, the sampling rate $f_s = 1/T$ must be D times the fundamental harmonic f_1, that is, $f_s = Df_1$, or equivalently, one period $T_{\text{per}} = 1/f_1$ must contain D samples, $T_{\text{per}} = DT$. This implies periodicity in n,

$$s_{n+D} = s\big((n+D)T\big) = s(nT + DT) = s(nT + T_{\text{per}}) = s(nT) = s_n$$

The assumed periodicity of s_n implies that the sum of any D successive samples, $(s_n + s_{n-1} + \cdots + s_{n-D+1})$, is a constant independent of n. In fact, it is equal to the DFT component S_0 at DC $(\omega_k = 0)$,

$$s_n + s_{n-1} + \cdots + s_{n-D+1} = S_0\,, \quad -\infty < n < \infty \tag{9.1.2}$$

In a seasonal + trend model such as $y_n = s_n + t_n + v_n$, we may be inclined to associate any DC term with the trend t_n rather with the periodic signal s_n. Therefore, it is common to assume that the DC component of s_n is absent, that is, the sum (9.1.2) is zero, $S_0 = 0$. In such cases, the comb filter for extracting s_n must be designed to have peaks only at the non-zero harmonics, $\omega_k = k\omega_1$, $k = 1, 2, \ldots, D-1$. Similarly, the notch filter for removing periodic noise must not have a notch at DC.

The typical technique for designing notch and comb filters for periodic signals is by frequency scaling, that is, the mapping of frequencies $\omega \to \omega D$, or equivalently, the mapping of the z-domain variable

$$z \to z^D \tag{9.1.3}$$

The effect of the transformation is to shrink the spectrum by a factor of D and then replicate it D times to fill the new Nyquist interval. An example is shown in Fig. 9.1.3 for $D = 4$. Starting with a lowpass filter $H_{\text{LP}}(\omega)$, the frequency-scaled filter will be a comb filter, $H_{\text{comb}}(\omega) = H_{\text{LP}}(\omega D)$. Similarly, a highpass filter is transformed into a notch filter $H_{\text{notch}}(\omega) = H_{\text{HP}}(\omega D)$.

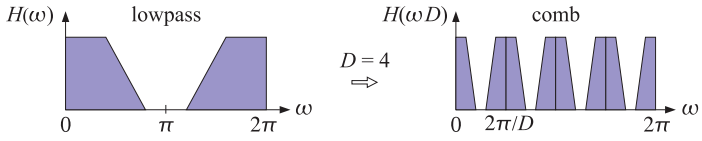


Fig. 9.1.3 Mapping of a lowpass filter to a comb filter by frequency scaling.

In the z-domain, we have the following simple prescriptions for turning lowpass and highpass filters into comb and notch filters:

$$\boxed{\begin{aligned} H_{\text{comb}}(z) &= H_{\text{LP}}(z^D) \\ H_{\text{notch}}(z) &= H_{\text{HP}}(z^D) \end{aligned}} \tag{9.1.4}$$

For example, the simplest comb and notch filters are generated by,

$$\begin{aligned} H_{\text{LP}}(z) &= \frac{1}{2}(1 + z^{-1}) \\ H_{\text{HP}}(z) &= \frac{1}{2}(1 - z^{-1}) \end{aligned} \quad \Rightarrow \quad \begin{aligned} H_{\text{comb}}(z) &= \frac{1}{2}(1 + z^{-D}) \\ H_{\text{notch}}(z) &= \frac{1}{2}(1 - z^{-D}) \end{aligned} \tag{9.1.5}$$

Their magnitude responses are shown in Fig. 9.1.4 for $D = 10$. The harmonics $\omega_k = 2\pi k/D = 2\pi k/10$, $k = 0, 1, \ldots, 9$ are the peaks/notches of the comb/notch filters. The original lowpass and highpass filter responses are shown as the dashed lines. The factors $1/2$ in Eq. (9.1.5) normalize the peak gains to unity. The magnitude responses of the two filters are:

$$\big|H_{\text{comb}}(\omega)\big|^2 = \cos^2(\omega D/2)\,, \quad \big|H_{\text{notch}}(\omega)\big|^2 = \sin^2(\omega D/2) \tag{9.1.6}$$

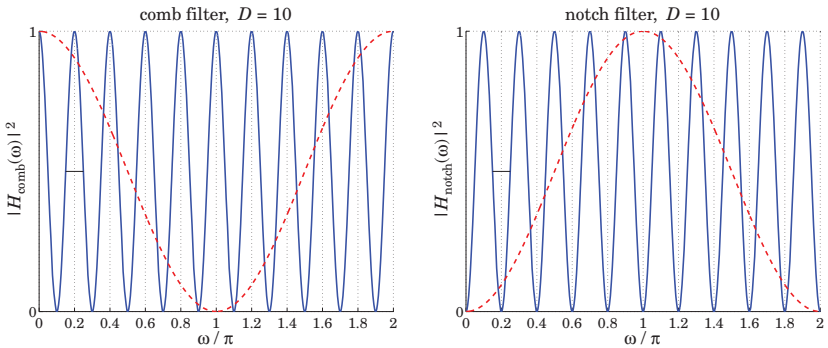


Fig. 9.1.4 Simple comb and notch filters with $D = 10$.

The filters are complementary, as well as power-complementary, in the sense,

$$H_{\text{comb}}(z) + H_{\text{notch}}(z) = 1\,, \quad \big|H_{\text{comb}}(\omega)\big|^2 + \big|H_{\text{notch}}(\omega)\big|^2 = 1 \tag{9.1.7}$$

The 3-dB widths $\Delta\omega$ of the comb peaks or the notch dips are fixed by the period D. Indeed, they are defined by the condition $\sin^2(D\Delta\omega/4) = 1/2$, which gives $\Delta\omega = \pi/D$. They are indicated on Fig. 9.1.4 as the short horizontal lines at the half-power level.

In order to control the width, we must consider IIR or higher order FIR filters. For example, we may start with the lowpass filter given in Eq. (2.3.5), and its highpass version,

$$H_{\rm LP}(z) = b\,\frac{1+z^{-1}}{1-az^{-1}}\,,\quad b=\frac{1-a}{2}\,,\qquad H_{\rm HP}(z) = b\,\frac{1-z^{-1}}{1-az^{-1}}\,,\quad b=\frac{1+a}{2} \tag{9.1.8}$$

where $0 < a < 1$. The transformation $z \to z^D$ gives the comb and notch filters [30]:

$$\begin{aligned} H_{\rm comb}(z) &= b\,\frac{1+z^{-D}}{1-az^{-D}}\,,\quad b=\frac{1-a}{2}\\ H_{\rm notch}(z) &= b\,\frac{1-z^{-D}}{1-az^{-D}}\,,\quad b=\frac{1+a}{2}\end{aligned} \tag{9.1.9}$$

The filters remain complementary, and power-complementary, with magnitude responses:

$$\begin{aligned} |H_{\rm comb}(\omega)|^2 &= \frac{\beta^2}{\beta^2+\tan^2(\omega D/2)}\,,\quad \beta\equiv\frac{1-a}{1+a}\\ |H_{\rm notch}(\omega)|^2 &= \frac{\tan^2(\omega D/2)}{\beta^2+\tan^2(\omega D/2)}\end{aligned} \tag{9.1.10}$$

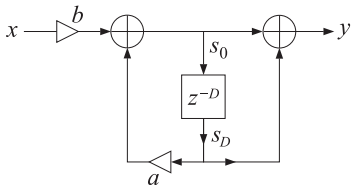
Their 3-dB width $\Delta\omega$ is controlled by the pole parameter a through the relation [30]:

$$\tan\left(\frac{D\Delta\omega}{4}\right) = \frac{1-a}{1+a} = \beta \tag{9.1.11}$$

The noise reduction ratio of the comb filter is the same as that of the lowpass filter $H_{\rm LP}(z)$, which was calculated in Chap. 2,

$$\mathcal{R} = \frac{1-a}{2} = \frac{\beta}{1+\beta} \tag{9.1.12}$$

and can be made as small as desired by increasing a towards unity, but at the expense of also increasing the time constant of the filter. The canonical (direct-form II) realization of the comb filter and its sample processing algorithm using a circular buffer implementation of the multiple delay z^{-D} is as follows in the notation of [30], where p is the circular pointer,



for each x do:
$s_D = *(p+D)$
$s_0 = bx + as_D$
$y = s_0 + s_D$
$*p = s_0$
$p{-}{-}$

Example 9.1.1: Fig. 9.1.5 shows two examples designed with $D = 10$ and 3-dB widths $\Delta\omega = 0.05\pi$ and $\Delta\omega = 0.01\pi$. By comparison, the simple designs had $\Delta\omega = \pi/D = 0.1\pi$. For $\Delta\omega = \pi/D$, we have $\tan(D\Delta\omega/4) = \tan(\pi/4) = 1$, which implies that $a = 0$ and $b = 1/2$, reducing to the simple designs of Eq. (9.1.5). □

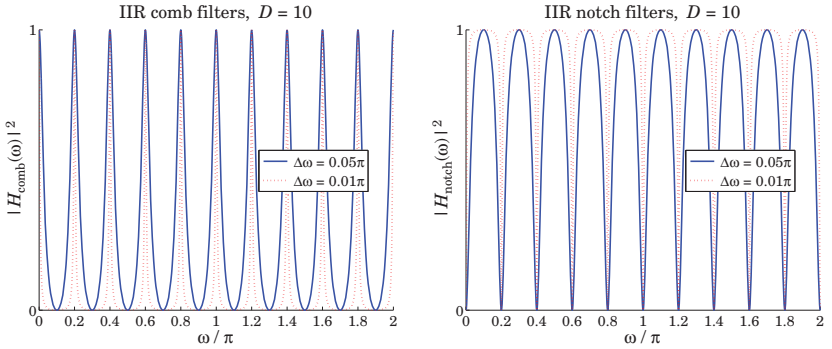


Fig. 9.1.5 Recursive comb and notch filters with $D = 10$.

Example 9.1.2: Fig. 9.1.6 shows on the left a simulated electrocardiogram (ECG) signal corrupted by 60 Hz power frequency interference and its harmonics. On the right, it shows the result of filtering by an IIR notch filter. The underlying ECG is recovered well after the initial transients die out.

The sampling rate was $f_s = 600$ Hz and the fundamental frequency of the noise, $f_1 = 60$ Hz. This gives for the period $D = f_s/f_1 = 10$. The ECG beat was taken to be 1 sec and therefore there were 600 samples in each beat for a total of 1200 samples in the two beats shown in the figure.

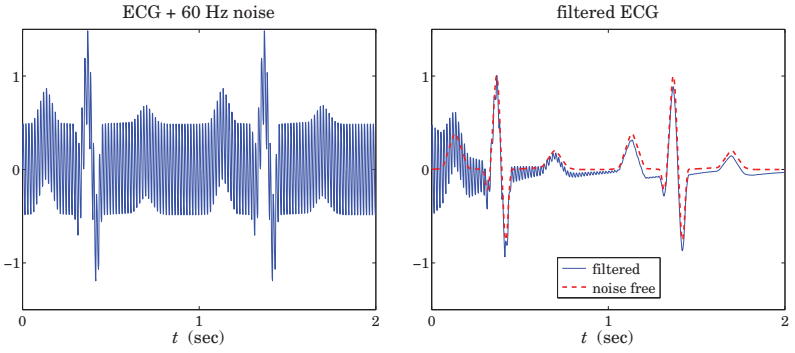


Fig. 9.1.6 Eliminating 60 Hz harmonics from ECG signal.

The IIR notch filter was designed to achieve a 3-dB width of $\Delta f = f_1/50$, that is, a Q-factor of $Q = f_1/\Delta f = 50$. Therefore, in units of rads/sample, the notch width is $\Delta\omega = 2\pi\Delta f/f_s = 2\pi/(DQ) = 0.004\pi$, which results in the filter parameters $a = 0.9391$ and $b = (1+a)/2 = 0.9695$. Thus, the designed notch filter was:

$$H_{\rm notch}(z) = 0.9695\,\frac{1-z^{-10}}{1-0.9391z^{-10}}$$

The noise was simulated by adding the following harmonic components,

$$v_n = \sum_{k=1}^{D/2-1} A_k \sin(\omega_k n)\,, \quad \text{with } \omega_k = \frac{2\pi k}{D}\,, \quad A_k = \frac{1}{2k^2}$$

where the amplitudes A_k were arbitrarily chosen. Note that only the non-zero harmonics that lie in the interval $0 < \omega < \pi$ were used.

The 40-dB time-constant of the notch filter was $n_{\text{eff}} = D\,\ln(0.01)\,/\,\ln(a) = 732$ samples or equivalently, $\tau = n_{\text{eff}}/f_s = 732/600 = 1.22$ sec. It is evident from the figure that beyond this time, the transients essentially die out. The MATLAB code for generating these graphs was as follows:

```
nbeats = 2; L = 600; M = 15;                % 600 samples per beat
s = ecgsim(nbeats,L,M);                     % simulated ECG
n = (0:length(s)-1)'; t = n/L;              % time in seconds

D = 10;  v = 0;
for k=1:D/2-1,
   v = v + (0.5/k^2) * sin(2*pi*k*n/D);     % generate noise
end

y = s + v;                                  % noisy ECG

Q = 50; beta = tan(pi/2/Q);
a = (1-beta)/(1+beta); b = (1+a)/2;         % filter parameters
aD = up([1,-a],D);                          % upsampled denominator coefficients
bD = up([b,-b],D);                          % upsampled numerator coefficients

x = filter(bD,aD,y);                        % filtered ECG

figure; plot(t,y);                          % left graph
figure; plot(t,x, t,s,':');                 % right graph
```

The MATLAB function `ecgsim`, which is part of the OSP toolbox, was used to generate the simulated ECG. It is based on the function `ecg` from [30]. The function `up` is used to upsample the highpass filter's coefficient vectors by a factor of D, generating the coefficient vectors of the notch filter, so that the built-in filtering function `filter` can be used. □

The upsampling operation used in the previous example is the time-domain equivalent of the transformation $z \to z^D$ and it amounts to inserting $D-1$ zeros between any two original filter coefficients. For example, applied to the vector $[h_0, h_1, h_2, h_3]$ with $D = 4$, it generates the upsampled vector:

$$[h_0, h_1, h_2, h_3] \;\to\; [h_0, 0, 0, 0, h_1, 0, 0, 0, h_2, 0, 0, 0, h_3]$$

The function `up` implements this operation,

```
g = up(h,D);        % upsampled vector
```

It is similar to MATLAB's built-in function `upsample`, except it does not append $D-1$ zeros at the end. The difference is illustrated by the following example,

```
      up([1,2,3,4],4) = [1  0  0  0  2  0  0  0  3  0  0  0  4]
upsample([1,2,3,4],4) = [1  0  0  0  2  0  0  0  3  0  0  0  4  0  0  0]
```


In addition to enhancing periodic signals or removing periodic interference, comb and notch filters have many other applications. The transformation $z \to z^D$ is widely used in audio signal processing for the design of reverberation algorithms emulating the delays arising from reflected signals within rooms or concert halls or in other types of audio effects [30]. The mapping $z \to z^D$ is also used in multirate signal processing applications, such as decimation or interpolation [30]. The connection to multirate applications can be seen by writing the frequency mapping $\omega' = \omega D$ in terms of the physical frequency f in Hz and sampling rate f_s,

$$\frac{2\pi f'}{f'_s} = \frac{2\pi f}{f_s} D$$

In the signal enhancement context, the sampling rates are the same $f'_s = f_s$, but we have frequency scaling $f' = fD$. On the other hand, in multirate applications, the frequencies remain the same $f' = f$ and the above condition implies the sampling rate change $f'_s = f_s/D$, which can be thought of as decimation by a factor of D from the high rate f_s to the low rate f'_s, or interpolation from the low to the high rate.

9.2 Notch and Comb Filters with Fractional Delay

The implementation of comb and notch filters requires that the sampling rate be related to the fundamental harmonic by $f_s = Df_1$ with D an integer, so that z^{-D} represents a D-fold multiple delay. In some applications, one may not have the freedom of choosing the sampling rate and the equation $D = f_s/f_1$ may result into a non-integer number.

One possible approach, discussed at the end of this section, is simply to design individual comb/notch filters for each desired harmonic $f_k = kf_1 = kf_s/D$, $k = 1, 2, \ldots$, that lies within the Nyquist interval, and then either cascade the filters together in the notch case, or add them in parallel in the comb case.

Another approach is to approximate the desired non-integer delay z^{-D} by an FIR filter and then use the IIR comb/notch structures of Eq. (9.1.9). Separating D into its integer and fractional parts, we may write:

$$D = D_{\text{int}} + d \tag{9.2.1}$$

where $D_{\text{int}} = \text{floor}(D)$ and $0 < d < 1$. The required multiple delay can be written then as $z^{-D} = z^{-D_{\text{int}}} z^{-d}$. The fractional part z^{-d} can be implemented by replacing it with an FIR filter that approximates it, that is, $H(z) \approx z^{-d}$, so that $z^{-D} \approx z^{-D_{\text{int}}} H(z)$. Then, the corresponding IIR comb/notch filters (9.1.9) will be approximated by

$$\begin{aligned} H_{\text{comb}}(z) &= b\,\frac{1 + z^{-D_{\text{int}}} H(z)}{1 - a\,z^{-D_{\text{int}}} H(z)}\,, \quad b = \frac{1-a}{2} \\ H_{\text{notch}}(z) &= b\,\frac{1 - z^{-D_{\text{int}}} H(z)}{1 - a\,z^{-D_{\text{int}}} H(z)}\,, \quad b = \frac{1+a}{2} \end{aligned} \tag{9.2.2}$$

Fig. 9.2.1 shows a possible realization. There exist many design methods for such approximate fractional delay filters [162]. We encountered some in Sec. 3.6. For example,

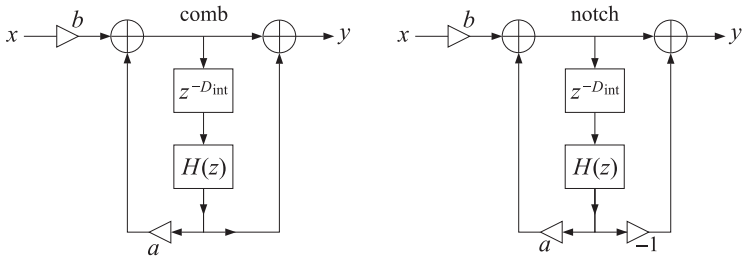


Fig. 9.2.1 Comb and notch filters with fractional delay.

the transfer functions of the causal Lagrange interpolation filters of orders 1 and 2 approximating the required non-integer delay d can be obtained from Eq. (3.6.30),

$$\begin{aligned} H(z) &= d + (1-d)z^{-1} \\ H(z) &= \frac{1}{2}(d-1)(d-2) - d(d-2)z^{-1} + \frac{1}{2}d(d-1)z^{-2} \end{aligned} \tag{9.2.3}$$

Such interpolation filters accurately cover only a fraction, typically 10–20%, of the Nyquist interval, and therefore, would be appropriate only if the first few harmonics are significant. A more effective approach suggested by [168] is to impose linear constraints on the design of $H(z)$ that preserve the required filter response at all the harmonics.

For integer delay D, the comb filter peaks or the notch filter nulls occur at the D-th roots of unity $z_k = e^{j\omega_k}$, $\omega_k = 2\pi k/D$, which satisfy $z_k^D = 1$.

For non-integer D, we require the same constraints for the delay filter $z^{-D_{\rm int}}H(z)$, that is, $z_k^{-D_{\rm int}}H(z_k) = 1$, or in terms of the frequency response, $e^{-j\omega_k D_{\rm int}}H(\omega_k) = 1$, where again $z_k = e^{j\omega_k}$, $\omega_k = 2\pi k/D$. Since $e^{-j\omega_k D} = 1$, we have,

$$e^{-j\omega_k D_{\rm int}}H(\omega_k) = 1 = e^{-j\omega_k D} = e^{-j\omega_k(D_{\rm int}+d)} \quad \Rightarrow \quad H(\omega_k) = e^{-j\omega_k d}$$

These are the constraints to be imposed on the design of $H(z)$. In order to obtain a real-valued impulse response for this filter, we must work with the harmonics that lie in the symmetric Nyquist interval, that is, $-\pi \le \omega_k \le \pi$, or,

$$-\pi \le \frac{2\pi k}{D} \le \pi \quad \Rightarrow \quad -\frac{D}{2} \le k \le \frac{D}{2}$$

Writing $D_{\rm int} = 2p + q$ and $D = D_{\rm int} + d = 2p + q + d$, with integer p and $q = 0, 1$, the above condition reads:

$$-p - \frac{1}{2}(q+d) \le k \le p + \frac{1}{2}(q+d)$$

Since $0 < d < 1$ and k must be an integer, we obtain,

$$-p \le k \le p \tag{9.2.4}$$

Thus, the design problem is to determine an FIR filter $H(z)$ such that $H(\omega) \approx e^{-j\omega d}$, and subject to the constraints:

$$H(\omega_k) = e^{-j\omega_k d}, \quad -p \le k \le p \tag{9.2.5}$$

When $q = d = 0$, we must choose $-p \le k \le p-1$, because $k = \pm p$ both are mapped onto the Nyquist frequency $\omega = \pm\pi$ and need be counted only once. In this case, of course, we expect the design method to produce the identity filter $H(z) = 1$.

Following [168], we use a constrained least-squares design criterion with the following performance index into which the constraints have been incorporated by means of complex-valued Lagrange multipliers λ_k:

$$\mathcal{J} = \int_{-\alpha\pi}^{\alpha\pi} |H(\omega) - e^{-j\omega d}|^2 \frac{d\omega}{2\pi} + \sum_{k=-p}^{p} [e^{-j\omega_k d} - H(\omega_k)]\lambda_k^* + \text{c.c.} = \min \tag{9.2.6}$$

where "c.c." denotes the complex conjugate of the second term. The approximation $H(\omega) \approx e^{-j\omega d}$ is enforced in the least-squares sense over a portion of the Nyquist interval, $[-\alpha\pi, \alpha\pi]$, where typically, $0.9 \le \alpha \le 1$, with $\alpha = 1$ covering the full interval. Assuming an Mth order filter $\mathbf{h} = [h_0, h_1, \ldots, h_M]^T$, we can write the frequency response in terms of the $(M+1)$-dimensional vectors,

$$H(\omega) = \sum_{n=0}^{M} h_n e^{-jn\omega} = \mathbf{s}_\omega^\dagger \mathbf{h}, \quad \mathbf{s}_\omega = \begin{bmatrix} 1 \\ e^{j\omega} \\ \vdots \\ e^{jM\omega} \end{bmatrix}, \quad \mathbf{h} = \begin{bmatrix} h_0 \\ h_1 \\ \vdots \\ h_M \end{bmatrix} \tag{9.2.7}$$

Similarly, we can express the gain constraints in the vector form,

$$S^\dagger \mathbf{h} = \mathbf{g} \tag{9.2.8}$$

where S is an $(M+1)\times(2p+1)$ matrix and $\mathbf{g}$ a $(2p+1)$-dimensional column vector defined component-wise by

$$\begin{aligned} S_{nk} &= e^{jn\omega_k}, \quad 0 \le n \le M, \quad -p \le k \le p \\ g_k &= e^{-j\omega_k d}, \quad -p \le k \le p \end{aligned} \tag{9.2.9}$$

that is,

$$S = [\ldots, \mathbf{s}_{\omega_k}, \ldots], \quad \mathbf{g} = \begin{bmatrix} \vdots \\ e^{-j\omega_k d} \\ \vdots \end{bmatrix} \tag{9.2.10}$$

It follows that the performance index can be written compactly as,

$$\mathcal{J} = \int_{-\alpha\pi}^{\alpha\pi} |\mathbf{s}_\omega^\dagger \mathbf{h} - e^{-j\omega d}|^2 \frac{d\omega}{2\pi} + \boldsymbol{\lambda}^\dagger(\mathbf{g} - S^\dagger\mathbf{h}) + (\mathbf{g} - S^\dagger\mathbf{h})^\dagger \boldsymbol{\lambda} = \min \tag{9.2.11}$$

where $\boldsymbol{\lambda} = [\ldots, \lambda_k, \ldots]^T$ is the vector of Lagrange multipliers. Expanding the first term of $\mathcal{J}$, we obtain,

$$\mathcal{J} = \mathbf{h}^\dagger R\mathbf{h} - \mathbf{h}^\dagger\mathbf{r} - \mathbf{r}^\dagger\mathbf{h} + \alpha + \boldsymbol{\lambda}^\dagger(\mathbf{g} - S^\dagger\mathbf{h}) + (\mathbf{g}^\dagger - \mathbf{h}^\dagger S)\boldsymbol{\lambda} = \min \tag{9.2.12}$$

where the matrix R and vector $\mathbf{r}$ are defined by,

$$R = \frac{1}{2\pi}\int_{-\alpha\pi}^{\alpha\pi} \mathbf{s}_\omega \mathbf{s}_\omega^\dagger \, d\omega\,, \quad \mathbf{r} = \frac{1}{2\pi}\int_{-\alpha\pi}^{\alpha\pi} \mathbf{s}_\omega e^{-j\omega d}\, d\omega \tag{9.2.13}$$

and component-wise,

$$\begin{aligned} R_{nm} &= \int_{-\alpha\pi}^{\alpha\pi} e^{j\omega(n-m)}\,\frac{d\omega}{2\pi} = \frac{\sin(\alpha\pi(n-m))}{\pi(n-m)}\,, \quad n,m = 0,1,\ldots,M \\ r_n &= \int_{-\alpha\pi}^{\alpha\pi} e^{j\omega(n-d)}\,\frac{d\omega}{2\pi} = \frac{\sin(\alpha\pi(n-d))}{\pi(n-d)}\,, \quad n = 0,1,\ldots,M \end{aligned} \tag{9.2.14}$$

We note that for $\alpha = \pi$, R reduces to the identity matrix. The optimal solution for $\mathbf{h}$ is obtained by setting the gradient of $\mathcal{J}$ to zero:

$$\frac{\partial\mathcal{J}}{\partial\mathbf{h}^*} = R\mathbf{h} - \mathbf{r} - S\boldsymbol{\lambda} = 0 \quad \Rightarrow \quad \mathbf{h} = R^{-1}\mathbf{r} + R^{-1}S\boldsymbol{\lambda} = \mathbf{h}_\mathrm{u} + R^{-1}S\boldsymbol{\lambda}$$

where $\mathbf{h}_\mathrm{u} = R^{-1}\mathbf{r}$ is the unconstrained solution of the least-squares problem. The Lagrange multiplier $\boldsymbol{\lambda}$ can be determined by multiplying both sides by $S^\dagger$ and using the constraint (9.2.8):

$$\mathbf{g} = S^\dagger\mathbf{h} = S^\dagger\mathbf{h}_\mathrm{u} + S^\dagger R^{-1}S\boldsymbol{\lambda} \quad \Rightarrow \quad \boldsymbol{\lambda} = (S^\dagger R^{-1}S)^{-1}(\mathbf{g} - S^\dagger\mathbf{h}_\mathrm{u})$$

Finally, substituting $\boldsymbol{\lambda}$ into the solution for $\mathbf{h}$, we obtain,

$$\boxed{\mathbf{h} = \mathbf{h}_\mathrm{u} + R^{-1}S(S^\dagger R^{-1}S)^{-1}(\mathbf{g} - S^\dagger\mathbf{h}_\mathrm{u})} \tag{9.2.15}$$

This type of constrained least-squares problem appears in many applications. We will encounter it again in the context of designing linearly constrained minimum variance beamformers for interference suppression, and in the problem of optimum stock portfolio design.

The MATLAB function `combfd` implements the above design method. Its inputs are the fractional period D, the order M of the filter $H(z)$, the comb/notch pole parameter a, and the Nyquist factor α,

```
[bD,aD,h,zmax] = combfd(D,M,a,alpha);
```
% comb/notch filter design with fractional delay

Entering the parameter a as negative indicates the design of a notch filter. The outputs `bD,aD` are the coefficients of the numerator and denominator polynomials of the comb/notch filters (9.2.2):

$$\begin{aligned} B_D(z) &= b\left[1 \pm z^{-D_\mathrm{int}}H(z)\right] \\ A_D(z) &= 1 - a\,z^{-D_\mathrm{int}}H(z) = 1 - az^{-D_\mathrm{int}}(h_0 + h_1 z^{-1} + \cdots + h_M z^{-M}) \end{aligned} \tag{9.2.16}$$

The output `h` is the impulse response vector $\mathbf{h}$, and `zmax` is the maximum pole radius of the denominator filter $A_D(z)$, which can be used to monitor the stability of the designed comb/notch filter. The pole parameter a can be fixed using the bandwidth equation (9.1.11), which is still approximately valid.

Fig. 9.2.2 shows a design example with fractional period $D = 9.1$, so that $D_\mathrm{int} = 9$ and $d = 0.1$. The other parameters were $M = 8$, $a = R^D$ with $R = 0.95$, and $\alpha = 1$.

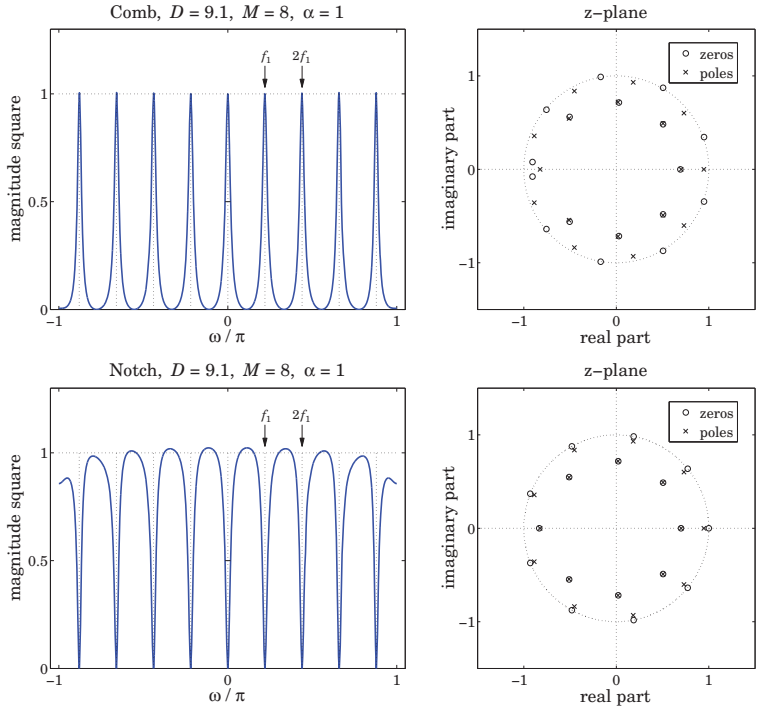


Fig. 9.2.2 Comb and notch filters with $D = 9.1$, and their pole/zero patterns.

The 3-dB widths obtained from Eq. (9.1.11) are indicated on the graphs by the two short horizontal lines at the half-power levels. The frequency plots are over the symmetric interval $-\pi \le \omega \le \pi$. The comb peaks have unity gain at the harmonics. For the notch case, the response between the notch dips is not very flat, but can be made flatter by decreasing the bandwidth, i.e., increasing the parameter a towards unity.

The right graphs depict the pole/zero patterns of the polynomials $B_D(z)$ and $A_D(z)$. These polynomials have orders $D_\mathrm{int} + M = 9 + 8 = 17$. For the comb filter, we observe how the $D_\mathrm{int} = 9$ poles arrange themselves around the unit circle at the harmonic frequencies, while the remaining 8 poles lie inside the unit circle.

The zeros of $B_D(z)$ also arrange themselves in two groups, 8 of them lying on the unit circle halfway between the comb peak poles, and the remaining 9 lying inside the

unit circle, with a group of 7 poles and 7 zeros almost falling on top of each other, almost canceling each other.

A similar pattern occurs for the notch filter, except now the notch zeros at the harmonics have poles lying almost behind them in order to sharpen the notch widths, while the remaining pole/zero pairs arrange themselves inside the unit-circle as in the comb case.

Generally, this design method tends to work well whenever D is near an odd integer, such as in the above example and in the top graphs of Fig. 9.2.4, which have $D = 9.1$ and $D = 8.9$. The method has some difficulty when D is near an even integer, such as $D = 9.9$ or $D = 8.1$, as shown in Figs. 9.2.3 and the bottom of 9.2.4.

In such cases, the method tends to place a pole or pole/zero pair on the real axis near $z = -1$ resulting in an unwanted peak or dip at the Nyquist frequency $\omega = \pi$. Such poles are evident in the pole/zero plots of Fig. 9.2.3. If D were exactly an even integer, then such pole/zero pair at Nyquist would be present, but for non-integer D, the Nyquist frequency is not one of the harmonics. Removing that pole/zero pair from the design, does not improve the problem.

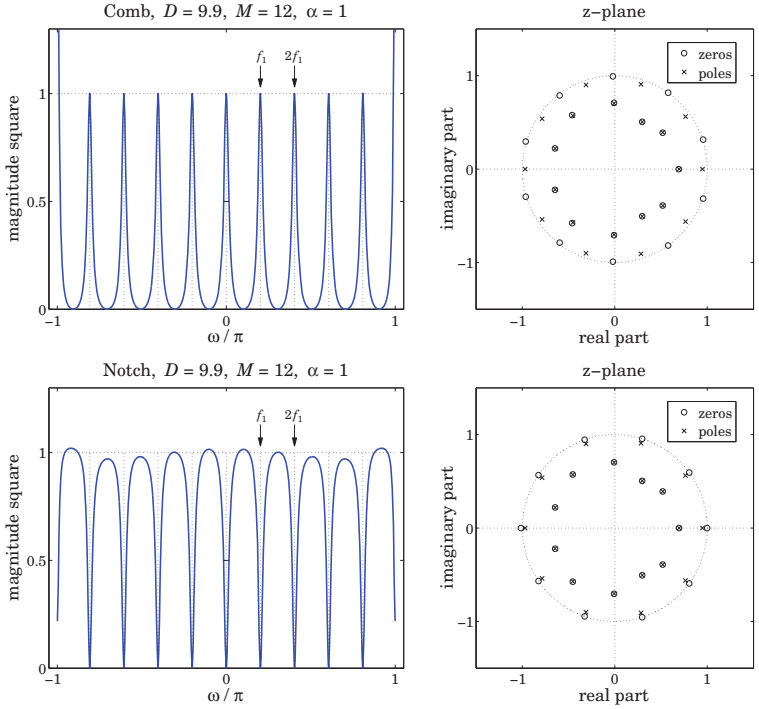


Fig. 9.2.3 Comb and notch filters with $D = 9.9$, and their pole/zero patterns.

The MATLAB code for generating the magnitude responses and pole/zero plots is the same for all three figures. In particular, Fig. 9.2.2 was generated by,

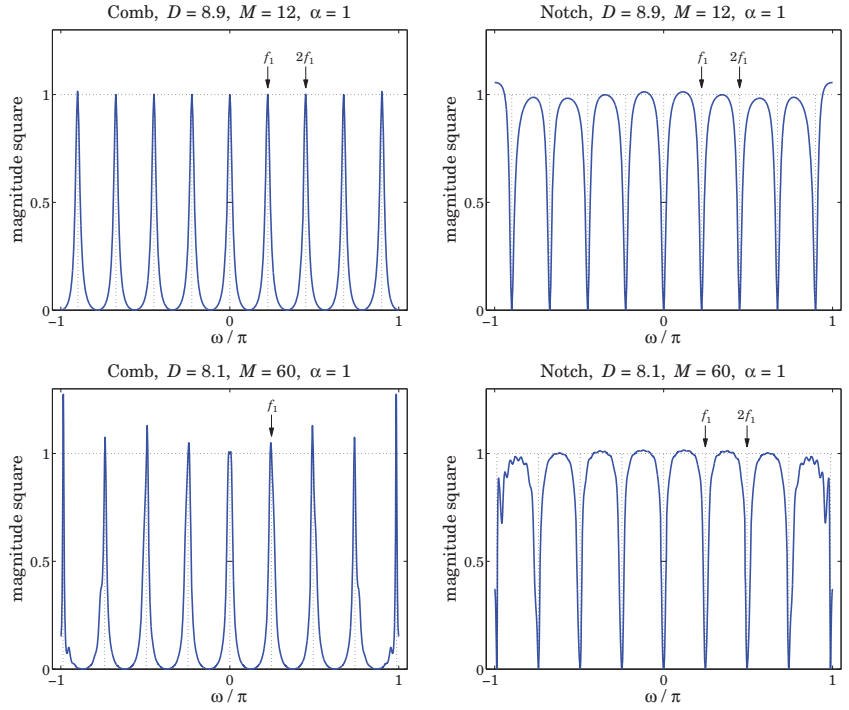


Fig. 9.2.4 Comb and notch filters with $D = 8.9$ and $D = 8.1$.

```
f = linspace(-1,1,4001); w = pi*f;                  % frequency range

D=9.1; R=0.95; a=R^D; M=8; alpha=1;                 % design parameters
beta = (1-a)/(1+a); Dw = 4/D * atan(beta);          % bandwidth calculation

[bD,aD,h,zmax] = combfd(D,M,a,alpha);               % comb, param a entered as positive
Hcomb = abs(freqz(bD,aD,w)).^2;                     % comb's magnitude rresponse

figure; plot(w/pi,Hcomb); figure; zplane(bD,aD);    % upper two graphs

[bD,aD,h,zmax] = combfd(D,M,-a,alpha);              % notch, param a entered as negative
Hnotch = abs(freqz(bD,aD,w)).^2;

figure; plot(w/pi,Hnotch); figure; zplane(bD,aD);   % lower two graphs
```

Parallel and Cascade Realizations

As we mentioned in the beginning of the previous section, an alternative approach is to design individual peak or notch filters at the harmonics and then combine the filters in parallel for the comb case, and in cascade for the notch case. Fig. 9.2.5 illustrates this

type of design for the two "difficult" cases of $D = 9.9$ and $D = 8.1$ using second-order peak/notch filters designed to have the same bandwidth as in Fig. 9.2.3.

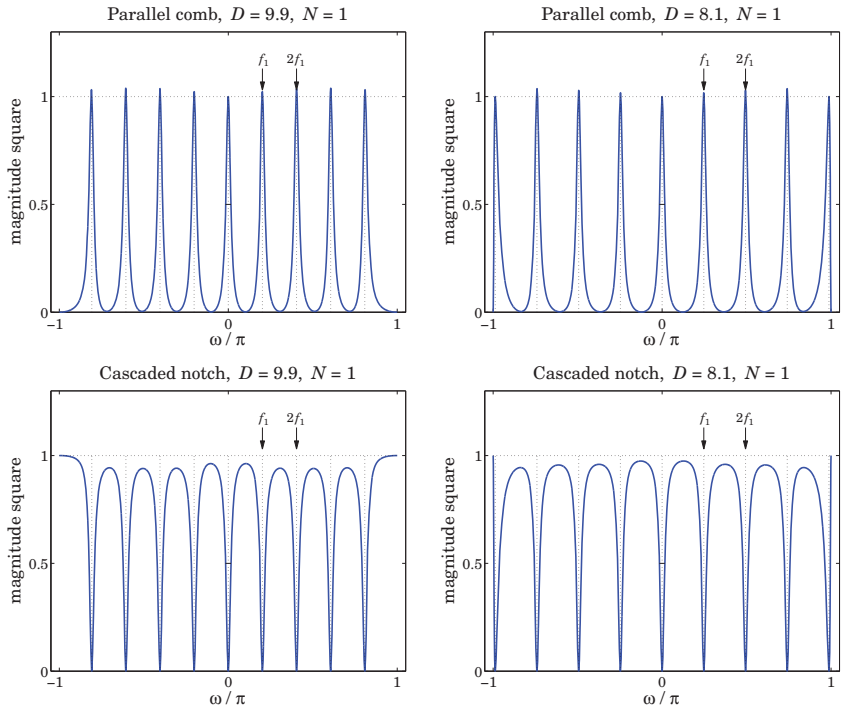


Fig. 9.2.5 Second-order parallel comb and cascaded notch filters.

Let $H_k(z)$ be the peak/notch filter for the kth harmonic $\omega_k = k\omega_1 = 2\pi k/D$, $k = 0, 1, \ldots, p$ and its negative $-\omega_k$. Then, the transfer functions of the comb and notch filters will be:

$$H_{\text{comb}}(z) = \sum_{k=0}^{p} H_k(z)\,, \qquad H_{\text{notch}}(z) = \prod_{k=0}^{p} H_k(z) \tag{9.2.17}$$

In their simplest form, the individual filters $H_k(z)$ are second-order and can be obtained from the lowpass and highpass filters (9.1.8) by the lowpass-to-bandpass z-domain transformation [30,595]:

$$z \;\to\; z' = \frac{z(\cos\omega_k - z)}{1 - z\cos\omega_k} \tag{9.2.18}$$

The resulting second-order peaking and notch filters are [30], for $k = 1, 2, \ldots, p$:

$$\begin{aligned} \text{peak:} \quad & H_k(z) = b\,\frac{1 - z^{-2}}{1 - (1+a)\cos\omega_k\, z^{-1} + a z^{-2}}\,, \quad b = \frac{1-a}{2} \\ \text{notch:} \quad & H_k(z) = b\,\frac{1 - 2\cos\omega_k\, z^{-1} + z^{-2}}{1 - (1+a)\cos\omega_k\, z^{-1} + a z^{-2}}\,, \quad b = \frac{1+a}{2} \end{aligned} \tag{9.2.19}$$

The filter parameter a is fixed in terms of the 3-dB width of the peak or the notch by,

$$\tan\left(\frac{\Delta\omega}{2}\right) = \frac{1-a}{1+a} = \beta \tag{9.2.20}$$

For $k = 0$, we may use the first-order lowpass/highpass filters of Eq. (9.1.8) without any z-domain transformation. But in order for their 3-dB frequency to match the specified 3-dB width $\Delta\omega$, their parameter a must be redefined as follows:

$$\tan\left(\frac{\Delta\omega}{4}\right) = \frac{1-a}{1+a} = \beta \tag{9.2.21}$$

To clarify the construction, we give below the MATLAB code for generating the left graphs of Fig. 9.2.5,

```
f = linspace(-1,1,4001); w = pi*f;                % frequency range -π ≤ ω ≤ π

D = 9.9; p = floor(D/2); w1 = 2*pi/D;             % design parameters
R = 0.95; a = R^2;

beta = (1-a)/(1+a); dw = 2*atan(beta);            % 3-dB width calculation
beta0 = tan(dw/4);  a0 = (1-beta0)/(1+beta0);     % bandwidth parameter for k = 0 section

A = [1, -a0, 0];                                  % denominator coefficients for k = 0
Bcomb =  [1, 1, 0] * (1-a0)/2;                    % numerator coefficients for k = 0
Bnotch = [1,-1, 0] * (1+a0)/2;

Hcomb  = freqz(Bcomb,A,w);                        % k = 0 section, H0(ω)
Hnotch = freqz(Bnotch,A,w);

for k=1:p,                                        % non-zero harmonics
   A = [1, -(1+a)*cos(k*w1), a];                  % denominator of Hk(z)
   Bcomb =  [1, 0, -1] * (1-a)/2;                 % numerator of peak Hk(z)
   Bnotch = [1, -2*cos(k*w1), 1] * (1+a)/2;       % numerator of notch Hk(z)
   Hcomb =  Hcomb + freqz(Bcomb,A,w);             % add in parallel for comb
   Hnotch = Hnotch .* freqz(Bnotch,A,w);          % cascade for notch
end

figure; plot(w/pi, abs(Hcomb).^2, '-');           left graphs
figure; plot(w/pi, abs(Hnotch).^2,'-');
```

It is evident from Fig. 9.2.5 that this design method is flexible enough to correctly handle any values of the fractional period D. However, because of the mutual interaction between the individual filters, the peaks of the comb do not quite have unity gains, and the segments between the nulls of the notch filter are not quite flat.

This behavior can be fixed by decreasing the width $\Delta\omega$. However, for a fixed value of $\Delta\omega$, the only way to improve the response is by using higher-order filters. For example,

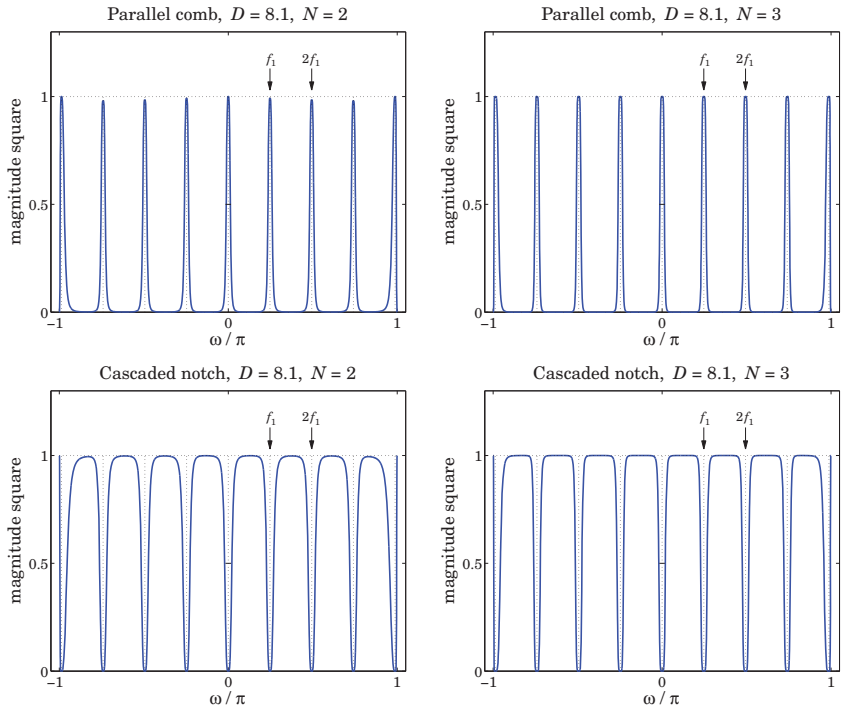


Fig. 9.2.6 High-order Butterworth parallel comb and cascaded notch filters.

Fig. 9.2.6 illustrates the cases of designing the individual filters using Butterworth filter prototypes of orders $N = 2$ and $N = 3$, whereas Fig. 9.2.5 corresponds to $N = 1$.

The following MATLAB code illustrates the generation of the left graphs in Fig. 9.2.6, and uses the functions `hpeq` and `frespc` from the high-order equalizer design toolbox in [595], which is also included in the OSP toolbox:

```
f = linspace(-1,1,4001); w = pi*f;

D = 8.1; p = floor(D/2); w1 = 2*pi/D;
R = 0.95; a = R^2;

beta = (1-a)/(1+a); dw = 2*atan(beta);          % 3-dB width

N = 2; GB = -20*log10(2);                       % Butterworth order and bandwidth gain

[B0,A0] = hpeq(N, -inf, 0, GB, 0, dw/2);        % k = 0 for comb, cutoff = half-bandwidth
Hcomb = frespc(B0,A0,w);

for k=1:p
   [B,A] = hpeq(N, -inf, 0, GB, k*w1, dw);      % non-zero harmonics
    Hcomb = Hcomb + frespc(B,A,w);              % add in parallel
end
```



```
figure; plot(w/pi,abs(Hcomb).^2,'-');           upper-left graph

[B0,A0] = hpeq(N, 0, -inf, GB, 0, dw/2);        % k = 0 for notch
Hnotch = frespc(B0,A0,w);

for k=1:p
   [B,A] = hpeq(N, 0, -inf, GB, k*w1, dw);
    Hnotch = Hnotch .* frespc(B,A,w);           % cascade in series
end

figure; plot(w/pi,abs(Hnotch).^2,'-');          % lower-left graph
```

The higher-order designs can also be based on Chebyshev or elliptic filters. In all cases, the starting point is a lowpass (or highpass) analog prototype filter $H_a(s)$, which is transformed into a peaking (or notch) filter centered at ω_k using the s-to-z domain bandpass transformation [30,595]:

$$H(z) = H_a(s)\,, \quad s = \frac{z'-1}{z'+1} = \frac{z^2 - 2\cos\omega_k\, z + 1}{z^2 - 1} \tag{9.2.22}$$

where z' is given by Eq. (9.2.18). For example, the analog Butterworth prototype filters of orders $N = 1, 2, 3$ are:

$$H_a(s) = \frac{\beta}{\beta + s}\,, \quad H_a(s) = \frac{\beta^2}{\beta^2 + \sqrt{2}\beta s + s^2}\,, \quad H_a(s) = \frac{\beta}{\beta + s} \cdot \frac{\beta^2}{\beta^2 + \beta s + s^2}$$

Similarly, for the notch filters, the analog prototypes are the highpass filters:

$$H_a(s) = \frac{s}{\beta + s}\,, \quad H_a(s) = \frac{s^2}{\beta^2 + \sqrt{2}\beta s + s^2}\,, \quad H_a(s) = \frac{s}{\beta + s} \cdot \frac{s^2}{\beta^2 + \beta s + s^2}$$

For arbitrary N, the Butterworth lowpass and highpass filters are:

$$H_a(s) = \left[\frac{\sigma}{\beta + s}\right]^r \prod_{i=1}^{L} \left[\frac{\sigma^2}{\beta^2 + 2\beta s \sin\phi_i + s^2}\right], \quad \phi_i = \frac{\pi(2i-1)}{2N} \tag{9.2.23}$$

where $N = 2L + r$, with integer L and $r = 0, 1$, and with $\sigma = \beta$ in the lowpass case, and $\sigma = s$ in the highpass one. The parameter β is related to the 3-dB width through $\beta = \tan(\Delta\omega/2)$. The filters of Eq. (9.2.19) are obtained by applying the transformation (9.2.22) to the $N = 1$ case.

Each peaking or notching filter is the cascade of L second-order sections in s or fourth-order sections in z (and possibly a second-order section in z if $r = 1$). The function `frespc` is used to calculate the corresponding frequency responses in such cascaded form. Further details on high-order designs and a description of the function `hpeq` can be found in [595].

9.3 Signal Averaging

Signal averaging is a technique for estimating a repetitive signal in noise. Evoked biological signals, GPS, and radar were some applications mentioned at the beginning of

this chapter. A variant of the method can also be used to deseasonalize business, social, and climate data—the difference being here that the non-periodic part of the measured signal is not only noise but it can also contain a trend component. The typical assumed noise model in signal averaging has the form:

$$y_n = s_n + v_n \tag{9.3.1}$$

where s_n is periodic with some period D, assumed to be an integer, and v_n is zero-mean white noise. The periodic signal s_n can be extracted by filtering y_n through any comb filter, such as the IIR filter of Eq. (9.1.9).

Signal averaging is equivalent to comb filtering derived by applying the D-fold replicating transformation $z \to z^D$ to an ordinary, length-N, lowpass FIR averaging filter:

$$H_{\text{LP}}(z) = \frac{1}{N}\left[1 + z^{-1} + z^{-2} + \cdots + z^{-(N-1)}\right] = \frac{1}{N}\,\frac{1 - z^{-N}}{1 - z^{-1}} \tag{9.3.2}$$

The definition $H(z) = H_{\text{LP}}(z^D)$, then gives the comb filter:

$$H(z) = \frac{1}{N}\left[1 + z^{-D} + z^{-2D} + \cdots + z^{-(N-1)D}\right] = \frac{1}{N}\,\frac{1 - z^{-ND}}{1 - z^{-D}} \tag{9.3.3}$$

The latter equation shows that $H(z)$ has zeros at all the (ND)-th roots of unity that are not D-th roots of unity. At the latter, the filter has unity-gain peaks.

An example is shown in Fig. 9.3.1, with period $D = 10$ and $N = 5$ and $N = 10$. The comb peaks are at the D-th roots of unity $\omega_k = 2\pi k/D = 2\pi k/10$, $k = 0, 1, \ldots, 9$. The 3-dB width of the peaks is indicated on the graphs by the short horizontal lines at the half-power level centered around the first harmonic.

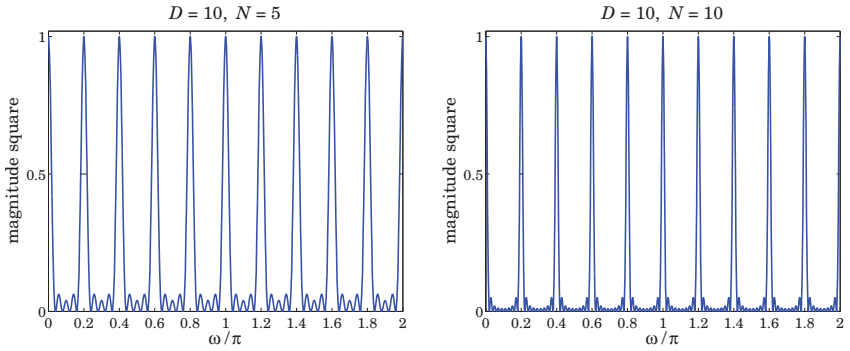


Fig. 9.3.1 Signal averaging filters with $D = 10$ and $N = 5, 10$.

The 3-dB width is given by

$$\Delta\omega = 0.886\,\frac{2\pi}{ND} \tag{9.3.4}$$

which follows from the frequency response of $H(z)$:

$$H(\omega) = \frac{1}{N}\,\frac{1 - e^{-j\omega ND}}{1 - e^{-j\omega D}} = \frac{1}{N}\,\frac{\sin(ND\omega/2)}{\sin(D\omega/2)}\,e^{-j(\omega(N-1)D/2} \tag{9.3.5}$$

Thus, the peaks get narrower with increasing number N of averaging periods. This has the effect of decreasing the noise, while letting through the periodic signal s_n.

The signal averaging interpretation can be seen from the time-domain operation of the filter. The corresponding output is the estimated periodic signal,

$$\hat{s}_n = \frac{1}{N}\left[y_n + y_{n-D} + y_{n-2D} + \cdots + y_{n-(N-1)D}\right] \tag{9.3.6}$$

Inserting $y_n = s_n + v_n$ and using the periodicity property $s_{n-D} = s_n$, we obtain,

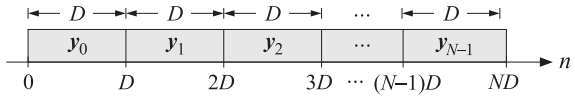
$$\hat{s}_n = s_n + \frac{1}{N}\left[v_n + v_{n-D} + \cdots + v_{n-(N-1)D}\right] \equiv s_n + \hat{v}_n \tag{9.3.7}$$

Because v_n was assumed to be stationary uncorrelated white noise, the variance of the filtered noise $\hat{v}_n$ will be reduced by a factor of N,

$$\sigma_{\hat{v}}^2 = \frac{1}{N^2}\left[\text{var}(v_n) + \text{var}(v_{n-D}) + \cdots + \text{var}(v_{n-(N-1)D})\right] = \frac{1}{N^2}\,(N\sigma_v^2) = \frac{1}{N}\,\sigma_v^2 \tag{9.3.8}$$

which implies that the NRR of the comb filter is $\mathcal{R} = 1/N$. Thus, by choosing N sufficiently large, the noise can be reduced, enabling the estimation of s_n.

Let the signal y_n be collected over N periods, that is, $0 \le n \le ND - 1$, and divide the signal into N length-D period segments as shown below,



The filtering operation (9.3.6) can be thought of as the averaging the N subblocks together. Indeed, let $y_i(n) = y_{iD+n}$, for $n = 0, 1, \ldots, D - 1$, be the samples within the i-th subblock, $i = 0, 1, \ldots, N - 1$. Then, we have

$$\frac{1}{N}\sum_{i=0}^{N-1} y_i(n) = \frac{1}{N}\sum_{i=0}^{N-1} y_{iD+n} = \frac{1}{N}\sum_{k=0}^{N-1} y_{(N-1)D+n-kD} = \hat{s}_{(N-1)D+n} \tag{9.3.9}$$

or, in words, the last D filter output samples, that is, over the period $[(N-1)D, ND-1]$, are the average of the samples over the last N periods. This can also be seen more simply by writing (9.3.6) in recursive form, which follows from Eq. (9.3.3),

$$\hat{s}_n = \hat{s}_{n-D} + \frac{1}{N}\,(y_n - y_{n-ND}) = \hat{s}_{n-D} + \frac{1}{N}\,y_n\,, \qquad 0 \le n \le ND - 1 \tag{9.3.10}$$

where the term y_{n-ND} was dropped because of the causal nature of y_n and the assumed range of n, that is, $0 \le n \le ND - 1$. Thus, Eq. (9.3.10) shows that $\hat{s}_n$ is the accumulation and averaging of the N period segments of y_n.

The MATLAB implementation of signal averaging is straightforward, for example, assuming that the array `y` has length at least ND,

```
s = 0;
for i=0:N-1,
   yi = y(i*D+1 : i*D+D);        % extract i-th period
   s = s + yi;                   % accumulate i-th period
end
s = s/N;                         % average of N periods
```

So far we have not imposed the constraint $S_0 = s_n + s_{n-1} + \cdots + s_{n-D+1} = 0$. If in addition to the noise component v_n, there is a slowly-varying background or trend present, say, t_n, so that the observation signal is $y_n = s_n + t_n + v_n$, then we may associate the constant S_0 with the trend and assume that $S_0 = 0$. To guarantee this constraint, we may subtract from each block $y_i(n)$ its local average, and compute the estimated periodic component by:

$$\hat{s}_n = \frac{1}{N}\sum_{i=0}^{N-1}[y_i(n) - \mu_i], \quad \mu_i = \frac{1}{D}\sum_{n=0}^{D-1} y_i(n) \tag{9.3.11}$$

which does satisfy $S_0 = 0$. By replicating the μ_i by D times within the i-th time period $[iD, iD + D - 1]$, and stringing the replicated values together over all the periods, we obtain a step-wise estimate of the trend component t_n. The following MATLAB code illustrates how to do that:

```
y = y(:);
L = length(y);
N = floor(L/D);                    % number of periods in y
r = mod(L,D);                      % L = ND + r

s = 0;
for i=0:N-1,
   yi = y(i*D+1 : i*D+D);          % i-th period
   m(i+1) = mean(yi);              % mean to be removed
   s = s + yi - m(i+1);            % accumulate i-th period
end
s = s / N;                         % estimated period

ys = repmat(s,N,1);                % replicate N periods
ys(end+1:end+r) = s(1:r);          % extend to length L by appending a portion of s

yt = repmat(m,D,1); yt = yt(:);    % repeat each mean D times within its period
yt(end+1:end+r) = yt(end);         % extend to length L by replicating last mean r times
```

where `ys` represents the estimated periodic signal, replicated over N periods, and `yt` is the estimated step-function trend. These above steps have been incorporated into the MATLAB function `sigav`:

```
[ys,s,yt] = sigav(y,D);            % signal averaging
```

Example 9.3.1: Fig. 9.3.2 shows a simulated signal averaging example. The period is $D = 10$ and the total number of periods $N = 100$. The graphs display only the first 10 periods to improve visibility. The periodic signal was superimposed on a slowly-varying trend and noise was added:

$$y_n = s_n + t_n + v_n, \quad s_n = 0.5\sin\left(\frac{4\pi n}{D}\right) + 0.5\sin\left(\frac{6\pi n}{D}\right), \quad t_n = \sin\left(\frac{2\pi n}{10D}\right)$$

where $n = 0, 1, \ldots, ND - 1$, and v_n is zero-mean, unit-variance, white noise. The upper row shows the noise-free case (with $v_n = 0$). The upper-right graph shows the periodic signal s_n. The estimated one resulting from the output of `sigav` is essentially identical to s_n and thus not displayed. The step-function estimated trend is shown on the upper-left.

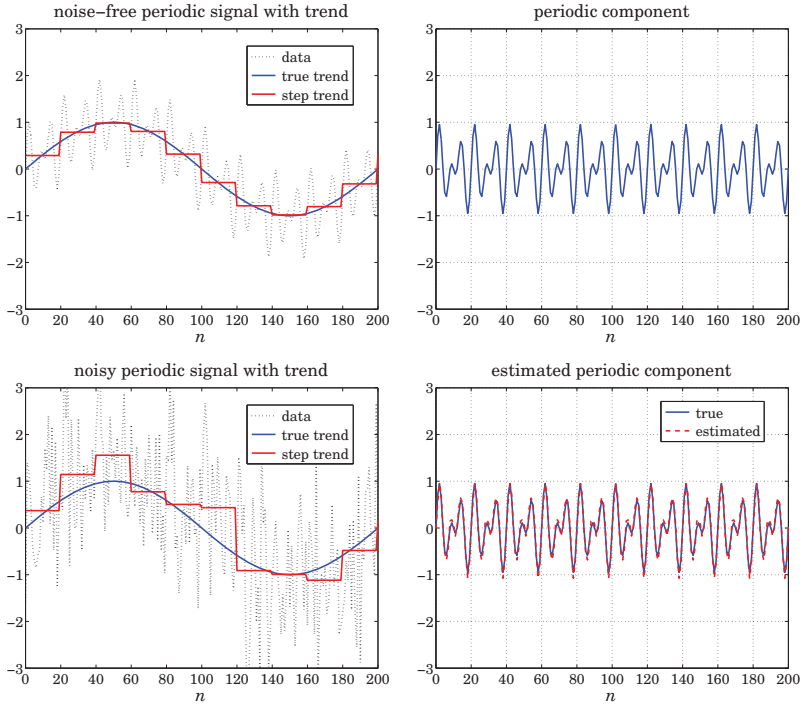


Fig. 9.3.2 Signal averaging of noisy periodic signal with slowly-varying trend.

The lower-left graph shows the noisy case, including the estimated step-trend signal. The lower-right graph shows the estimated periodic signal from the output of `sigav`. The following MATLAB code illustrates the generation of the bottom graphs:

```
D = 20; N = 100; n = 0:N*D;

s = (sin(4*pi*n/D) + sin(6*pi*n/D))/2;          % periodic component
t = sin(2*pi*n/D/10);                           % trend component

seed = 2008; randn('state',seed);
v = randn(size(n));

y = s + t + v;                                  % noisy observations

[ys,p,yt] = sigav(y,D);                         % signal averaging, p = one period

figure; plot(n,y,'--', n,t,'-.', n,yt,'-');     % yt is the estimated trend
xlim([0,200]);                                  % show only the first 10 periods
figure; plot(n,ys, '-');                        % estimated periodic component
xlim([0,200]);
```

Example 9.3.2: *Housing Starts.* Fig. 9.3.3 shows the application of signal averaging to the monthly, not seasonally adjusted, new privately-owned housing starts, for the 25 year period from January 1984 to December 2008. The data are from the US Census Bureau from the web link: `http://www.census.gov/ftp/pub/const/starts_cust.xls`.

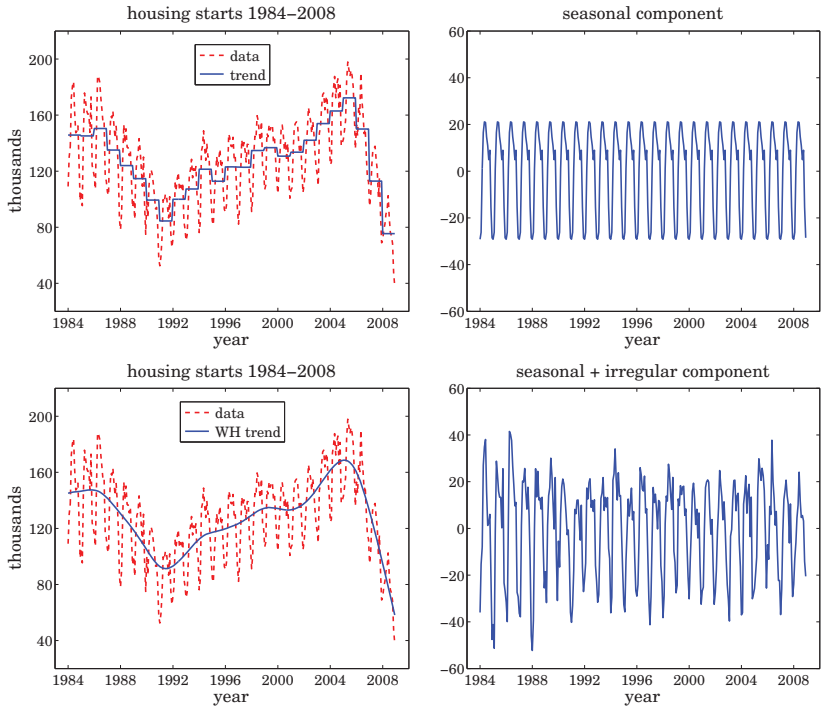


Fig. 9.3.3 Signal averaging and smoothing of monthly housing data.

The upper graphs show the estimated step-wise trend and the seasonal, periodic, component. Although there is clear annual periodicity in the data, the signal averaging method is not the best approach to this application because it does not result into a smooth trend. We consider better methods to deseasonalize such data in the next sections.

As an alternative method, the bottom graphs show the application of the Whittaker Henderson smoothing method to estimate the smooth trend. The optimal smoothing parameter was determined by the GCV criterion to be $\lambda = 6850$ and the smoothing order was $s = 2$.

The difference between the raw data and the estimated trend represents the seasonal plus irregular component and is plotted in the bottom-right graph. Further application of signal averaging to this component will generate an estimate of the seasonal component. It is not plotted because it is essentially identical to that shown in the upper-right graph.

The following MATLAB code illustrates the generation of the four graphs, including, but commented out, the computation of the seasonal part for the bottom graphs:

```
Y = loadfile('newhouse.dat');          % data file available in the OSP toolbox

i = find(Y(:,1)==109.1);               % finds the beginning of the year 1984

y = Y(i:end-4,1);                      % keep data from Jan.1984 to Dec.2008
t = taxis(y,12,1984);                  % define time axis

[ys,s,yt] = sigav(y,12);               % signal averaging with period 12

figure; plot(t,y,'--', t,yt,'-');      % upper-left graph
figure; plot(t,ys,'-');                % upper-right graph

s = 2; la = 6800:2:6900;               % smoothing order and search-range of λ's
[gcv,lopt] = whgcv(y,la,s);            % optimum smoothing parameter, λopt = 6850

yt = whsm(y,lopt,s);                   % Whittaker-Henderson smoothing
ysi = y-yt;                            % seasonal + irregular component
% ys = sigav(ysi,12);                  % seasonal component, not shown

figure; plot(t,y,'--', t,yt,'-');      % bottom-left graph
figure; plot(t,ysi,'-');               % bottom-right graph
% figure; plot(t,ys,'-');              % essentially the same as upper-right graph
```


9.4 Ideal Seasonal Decomposition Filters

A possible approach for separating the three components of the signal $y_n = s_n + t_n + v_n$ is to first estimate the trend t_n using a lowpass filter, and then extract the seasonal component s_n by applying a comb filter to the residual $r_n = y_n - t_n = s_n + v_n$, which consists of the seasonal and irregular parts.

The technique assumes of course that the trend is a slowly-varying, low-frequency, signal. Fig. 9.4.1 illustrates some typical frequency spectra for the three components and the ideal filters that might be used to extract them.

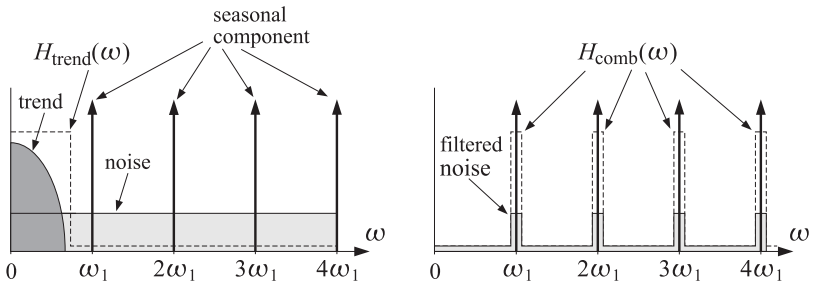


Fig. 9.4.1 Ideal filters for decomposition into trend and seasonal components.

Let $H_{\text{trend}}(z)$ be the trend-extraction filter and $H_{\text{comb}}(z)$ the comb filter with peaks at the seasonal harmonics (excluding the one at DC). Then, the filtering equations for

extracting the three components from y_n can be expressed in the z-domain as follows:

$$\begin{aligned} T(z) &= H_{\text{trend}}(z)\,Y(z) \\ R(z) &= S(z)+V(z) = Y(z)-T(z) = \big[1-H_{\text{trend}}(z)\big]Y(z) \\ S(z) &= H_{\text{comb}}(z)\,R(z) = H_{\text{comb}}(z)\big[1-H_{\text{trend}}(z)\big]Y(z) \equiv H_S(z)\,Y(z) \\ V(z) &= R(z)-S(z) = \big[1-H_{\text{comb}}(z)\big]\big[1-H_{\text{trend}}(z)\big]Y(z) \equiv H_I(z)\,Y(z) \end{aligned}$$

where $Y(z), S(z), T(z), V(z), R(z)$ are the z-transforms of y_n, s_n, t_n, v_n, r_n. Thus, the filters for extracting the three components are:

$$\begin{aligned} H_T(z) &= H_{\text{trend}}(z) && \text{(trend)} \\ H_S(z) &= H_{\text{comb}}(z)\big[1-H_{\text{trend}}(z)\big] && \text{(seasonal)} \\ H_I(z) &= \big[1-H_{\text{comb}}(z)\big]\big[1-H_{\text{trend}}(z)\big] && \text{(irregular)} \end{aligned} \tag{9.4.1}$$

The three filters satisfy the complementarity property:

$$H_T(z)+H_S(z)+H_I(z) = 1 \tag{9.4.2}$$

In Example 9.3.2, we followed exactly this approach where the trend filter was implemented as a Whittaker-Henderson smoother and the comb filter as a signal averager. Other possibilities exist for these filters and a lot of research has gone into making choices that try to balance a good filter response versus the ability to work well with short data records, including the handling of the end-point problem.

Example 9.4.1: *Housing Starts.* The housing starts signal considered in Example 9.3.2 displays the typical frequency spectra shown in Fig. 9.4.1.

The left graph in Fig. 9.4.2 shows the corresponding magnitude spectrum of the original data signal y_n, normalized to unity maximum and plotted over the symmetric Nyquist interval $[-\pi,\pi]$ in units of the fundamental harmonic $\omega_1 = 2\pi/12$. The spectrum is dominated by the low-frequency trend signal. The right graph shows the spectrum of the seasonal plus irregular component $r_n = y_n - t_n = s_n + v_n$, which displays the harmonics more clearly.

The following MATLAB code illustrates the computation of the spectra:

```
Y = loadfile('newhouse.dat');          % data file available in the OSP toolbox
i = find(Y(:,1)==109.1);               % finds the beginning of the year 1984
y = Y(i:end-4,1);                      % keep data from Jan.1984 to Dec.2008

s = 2; lopt = 6850                     % use optimum λ from Example 9.3.2
yt = whsm(y,lopt,s);                   % Whittaker-Henderson smoothing
ysi = y - yt;                          % seasonal + irregular component

k = linspace(-6,6,1201); w = 2*pi*k/12;                % frequency ω = kω1
L = length(y);
wind = 0.54 - 0.46*cos(2*pi*(0:L-1)/(L-1))';           % Hamming window

Y = abs(freqz(y.*wind,1,w));   Y = Y/max(Y);           % normalized spectrum
Ysi = abs(freqz(ysi.*wind,1,w)); Ysi = Ysi/max(Ysi);

figure; plot(k,Y); figure; plot(k,Ysi);                % left and right graphs
```

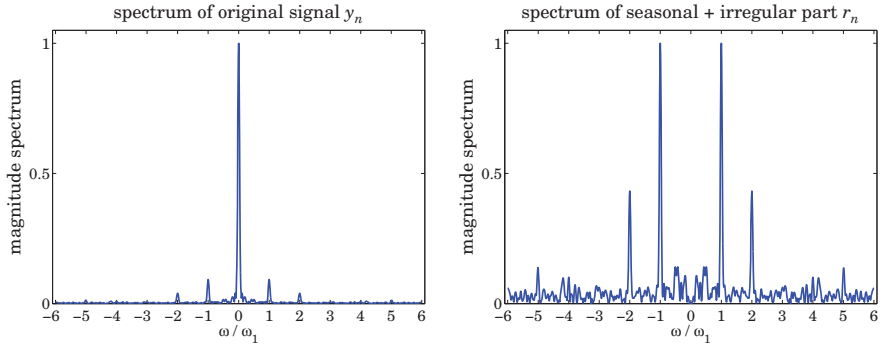



Fig. 9.4.2 Spectra of monthly housing data with and without trend.

The signals were windowed by a Hamming window prior to computing their DTFTs. □

Ideally, it does not matter if $H_{\text{comb}}(z)$ excludes or not the peak at DC because it would be canceled from $H_S(z)$ by the presence of the factor $\big[1-H_{\text{trend}}(z)\big]$. However, in practice because the filters are non-ideal, an extra step is usually taken to ensure that this peak is absent or minimized from s_n. For example, an additional de-trending step may be applied to $S(z)$, that is,

$$\begin{aligned} S_{\text{prelim}}(z) &= H_{\text{comb}}(z)\,R(z) \\ S(z) &= S_{\text{prelim}}(z) - H_{\text{trend}}(z)\,S_{\text{prelim}}(z) = H_{\text{comb}}(z)\big[1-H_{\text{trend}}(z)\big]^2 Y(z) \end{aligned} \tag{9.4.3}$$

This results in the modified extraction filters, which still satisfy (9.4.2):

$$\begin{aligned} H_T(z) &= H_{\text{trend}}(z) && \text{(trend)} \\ H_S(z) &= H_{\text{comb}}(z)\big[1-H_{\text{trend}}(z)\big]^2 && \text{(seasonal)} \\ H_I(z) &= \big[1-H_{\text{trend}}(z)\big]\big\{1-H_{\text{comb}}(z)\big[1-H_{\text{trend}}(z)\big]\big\} && \text{(irregular)} \end{aligned} \tag{9.4.4}$$

Further refinements will be discussed later on.

9.5 Classical Seasonal Decomposition

The classical seasonal decomposition method is the simplest realization of the procedure outlined in the previous section. Consider the following two possible lowpass trend-extraction filters:

$$\begin{aligned} H_{\text{trend}}(z) &= \frac{1}{D}\big[1+z^{-1}+z^{-2}+\cdots+z^{-(D-1)}\big] \\ H_{\text{trend}}(z) &= \frac{1}{D}\big[1+z^{-1}+z^{-2}+\cdots+z^{-(D-1)}\big]\cdot\frac{1}{2}(1+z^{-1}) \end{aligned} \tag{9.5.1}$$

where D is the period of the seasonal component. The first is typically used when D is odd, and the second, when D is even. They are referred to as the $1\times D$ and $2\times D$ trend

filters, the notation $N_1 \times N_2$ denoting the convolution of a length-N_1 with a length-N_2 averaging filter:

$$\frac{1}{N_1}\left[1 + z^{-1} + \cdots + z^{-(N_1-1)}\right] \cdot \frac{1}{N_2}\left[1 + z^{-1} + \cdots + z^{-(N_2-1)}\right] \tag{9.5.2}$$

The filters (9.5.1) are not perfect but are widely used. They have the desirable property of having nulls at the non-zero harmonics $\omega_k = k\omega_1 = 2\pi k/D, k = 1, 2, \ldots, D-1$. Their 3-dB cutoff frequency is about one-half the fundamental harmonic ω_1, that is,

$$\omega_c = 0.886\,\frac{\pi}{D} \tag{9.5.3}$$

Eq. (9.5.3) can easily be derived for the $1 \times D$ case and is a good approximation for the $2 \times D$ case. Fig. 9.5.1 shows the magnitude response $|H_{\text{trend}}(\omega)|$ versus ω over the symmetric Nyquist interval, $-\pi \le \omega \le \pi$. The 3-dB frequency is indicated on the graph at the $1/\sqrt{2}$ level.

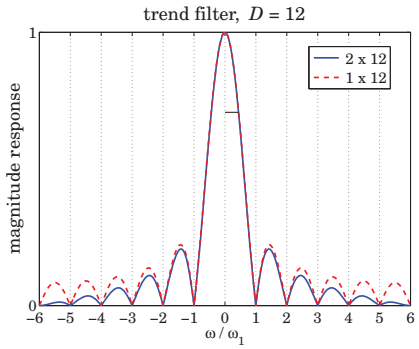


Fig. 9.5.1 Trend-extraction filters with $D = 12$.

In order to avoid delays introduced by the filters, the filters can be made symmetric with respect to the time origin. Let $D = 2p+1$ or $D = 2p$ in the even or odd case. Then, the symmetrized versions of the filters (9.5.1) are obtained by advancing them by p time units, that is, multiplying them by a factor of z^p:

$$\begin{aligned} D &= 2p+1\,, \quad H_{\text{trend}}(z) = z^p \frac{1}{D}\left[1 + z^{-1} + \cdots + z^{-(D-1)}\right] \\ D &= 2p\,, \qquad H_{\text{trend}}(z) = z^p \frac{1}{D}\left[1 + z^{-1} + \cdots + z^{-(D-1)}\right] \cdot \frac{1}{2}(1 + z^{-1}) \end{aligned} \tag{9.5.4}$$

The corresponding frequency responses are obtained by setting $z = e^{j\omega}$:

$$\begin{aligned} D &= 2p+1\,, \quad H_{\text{trend}}(\omega) = \frac{\sin(D\omega/2)}{D\sin(\omega/2)} \\ D &= 2p\,, \qquad H_{\text{trend}}(\omega) = \frac{\sin(D\omega/2)}{D\sin(\omega/2)} \cdot \cos(\omega/2) \end{aligned} \tag{9.5.5}$$

where we used the identity $1+z^{-1}+\cdots+z^{-(D-1)} = (1-z^{-D})/(1-z^{-1})$. The symmetric impulse responses are:

$$\begin{aligned} D &= 2p+1\,, \quad \mathbf{h}_{\text{trend}} = \frac{1}{D}[1, \underbrace{1, \ldots, 1}_{2p-1 \text{ ones}}, 1] \\ D &= 2p\,, \qquad \mathbf{h}_{\text{trend}} = \frac{1}{D}[0.5, \underbrace{1, \ldots, 1}_{2p-1 \text{ ones}}, 0.5] \end{aligned} \tag{9.5.6}$$

In both cases, the filter length is $2p+1$, and the time-domain operation for calculating the estimated trend is by the symmetric convolutional equation:

$$\hat{t}_n = \sum_{i=-p}^{p} h_{\text{trend}}(i)\, y_{n-i} \tag{9.5.7}$$

The issues of filtering with double-sided filters were discussed in Sec. 3.9. We recall that for a length-L input signal y_n, the steady-state filtered output is over the time range $p \le n \le L-1-p$. The first p and last p output transients can be computed using appropriate asymmetric filters, and there exist many possibilities for these. Musgrave's minimum-revision method, discussed in Sec. 9.8, constructs such asymmetric filters from a given symmetric filter such as $\mathbf{h}_{\text{trend}}$.

The calculation of the trend estimate, incorporating also the end-point asymmetric filters, can be carried out with the MATLAB functions `trendma`, `minrev`, and `lpfilt`,

```
htrend = trendma(D);          % trend filters of Eq. (9.5.6)
B = minrev(htrend,R);         % corresponding smoothing matrix
t_hat = lpfilt(B,y);          % filtering operation
```

where `y` denotes the input data vector, and R is the Musgrave parameter to be explained in Sec. 9.8. The use of asymmetric filters affects only the first p and last p outputs.

In the so-called *classical decomposition method*, we apply the above filtering procedure to calculate the trend, and then apply ordinary signal averaging on the residual $r_n = y_n - t_n$ to calculate the seasonal component. The following computational steps describe the method:

```
B = minrev(trendma(D),R);     % trend moving-average, with minimum-revision end-filters
yt = lpfilt(B,y);             % trend component
yr = y - yt;                  % seasonal + irregular components
ys = sigav(yr,D);             % seasonal component
yi = yr - ys;                 % irregular component
```

For a multiplicative decomposition, $y_n = s_n t_n \nu_n$, the last three steps are replaced by,

```
yr = y ./ yt;                 % seasonal + irregular components
ys = sigav(yr,D);             % seasonal component
yi = yr ./ ys;                % irregular component
```

The function `cldec` implements the above steps,

```
[yt,ys,yi] = cldec(y,D,R,type);     % classical decomposition method
```

where the string `type` takes on the values `'a'` or `'m'` for additive (the default) or multiplicative decomposition. The default value of R is zero, which simply omits the computation of the first and last p transients and replaces them with the corresponding samples of the input signal y_n.

Example 9.5.1: *Housing Starts.* Fig. 9.5.2 the trend and seasonal components of the housing starts data extracted by the classical decomposition method versus the methods discussed in Example 9.3.2.

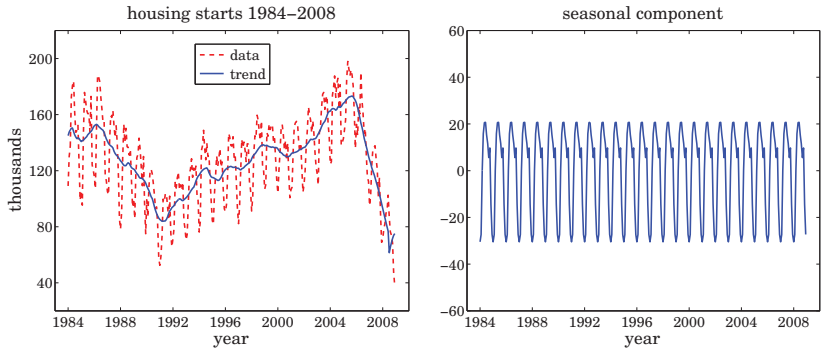


Fig. 9.5.2 Classical decomposition of monthly housing data.

The Musgrave parameter was chosen to be $R = 10$. Since $D = 12$, the value of R affects only the first and last 6 outputs. The MATLAB code for generating these graphs was,

```
Y = loadfile('newhouse.dat');
i = find(Y(:,1)==109.1);
y = Y(i:end-4,1); t = taxis(y,12,1984);

D=12; R=10;
[yt,ys,yi] = cldec(y,D,R);              % classical decomposition method

figure; plot(t,y,'--', t,yt,'-');       % left graph
figure; plot(t,ys,'-');                 % right graph
```

The estimated trend is not as smooth as that of the Whittaker-Henderson method, but the estimated seasonal component is essentially the same as that of Example 9.3.2. □

Example 9.5.2: *Global Carbon Dioxide Data.* Figure 9.5.3 shows on the upper-left the monthly global CO_2 data for the period of January 1980 to March 2009, obtained from the NOAA web site: `http://www.esrl.noaa.gov/gmd/ccgg/trends/`.

The vertical axis is in parts per million (ppm), which represents the dry air mole fraction, that is, the number of CO_2 molecules divided by the number of all air molecules, after water vapor has been removed.

The upper graphs show the application of the classical seasonal decomposition method. The upper-left graph shows the trend t_n extracted by a 2×12 moving-average filter, while the right graph shows the seasonal component s_n.

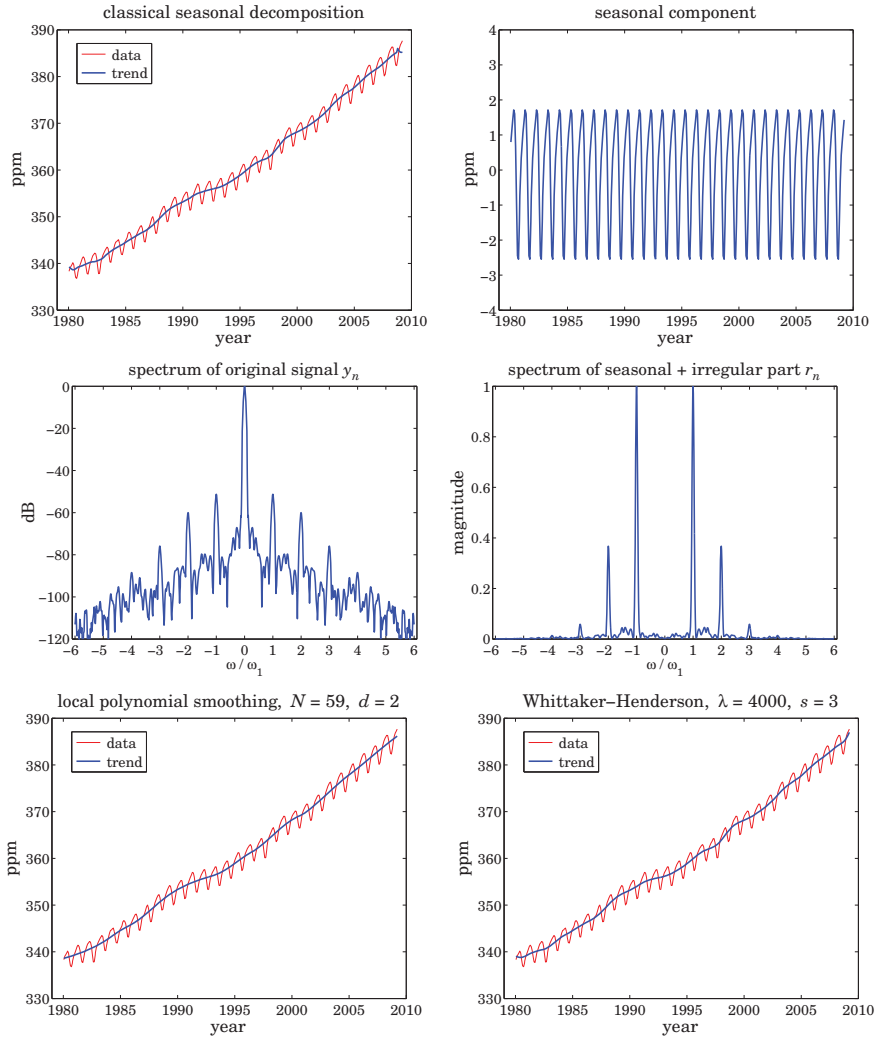


Fig. 9.5.3 Monthly global CO_2 data and spectra.

The middle graphs show the spectra of the original data on the left, and of the residual part $r_n = y_n - t_n = s_n + v_n$ on the right. The frequency axis is the symmetric Nyquist interval $[-\pi, \pi]$ in units of the fundamental harmonic $\omega_1 = 2\pi/12$. The trend dominates the spectrum of y_n and swamps the smaller harmonic peaks of the seasonal part. Indeed, the level of the seasonal component relative to the trend can be estimated in dB to be:

$$20\log_{10}\left(\frac{\text{std}(s_n)}{\text{mean}(y_n)}\right) = 20\log_{10}\left(\frac{1.46}{360}\right) = -47.8 \text{ dB}$$

Therefore, the spectrum of the seasonal component is too small to be visible if plotted in absolute units. In order to make it visible, a Kaiser window with an 80-dB sidelobe level was applied to y_n prior to computing its spectrum and then plotted in dB. On the other hand, after the trend is removed, the harmonics in the residual component r_n are quite visible if plotted in absolute units as in the middle-right graph.

The bottom two graphs show the trend component t_n extracted by a local polynomial smoothing filter on the left (with length $N = 59$ and length $d = 2$), and by a Whittaker-Henderson smoother on the right (with $\lambda = 4000$ and $s = 3$). The corresponding seasonal components obtained by signal averaging of the residual $r_n = y_n - t_n$ are not shown because they are essentially the same as that of the upper-right graph. The MATLAB code used to generate these six graphs was as follows:

```
Y = loadfile('co2_mm_gl.dat');                % data file in the OSP toolbox
t = Y(:,3); y = Y(:,4); yt0 = Y(:,5);         % extract times and signals

R = 15; [yt,ys,yi] = cldec(y,12,R);           % classical decomposition

figure; plot(t,y, t,yt);                      % upper-left graph
figure; plot(t,ys);                           upper-right graph

k = linspace(-6,6,1201); w = 2*pi*k/12;       % frequency in units of ω1

L = length(y); Rdb = 80;                      % Kaiser window parameters
wind = kwindow(L,Rdb)';                       % Kaiser window in the OSP toolbox

ysi = y - yt;                                 % seasonal + irregular component

Y =   abs(freqz(y.*wind,1,w));   Y = Y/max(Y);          % DTFT computation
Ysi = abs(freqz(ysi.*wind,1,w)); Ysi = Ysi/max(Ysi);

figure; plot(k, 20*log10(Y));                 % middle-left graph
figure; plot(k, Ysi);                         % middle-right graph

N=59; d=2; yt = lpfilt(lpsm(N,d),y);          % LPSM smoother

figure; plot(t,y, t,yt);                      % bottom-left graph
% ys = sigav(y-yt,12);                        % seasonal part, not shown
% figure; plot(t,ys);

la=4000; s=3; yt = whsm(y,la,s);              % Whittaker-Henderson smoother

figure; plot(t,y, t,yt);                      % bottom-right graph
```

The signal `yt0` extracted from the 5th column of the data file (as in the second line of code above) represents the already de-seasonalized data, and therefore, we can compare it to the trend extracted by the above three methods. It is not plotted because it is virtually identical to the above extracted trends.

The percentage error defined as `100*norm(yt-yt0)/norm(yt0)` is found to be 0.05%, 0.07%, and 0.05% for the classical, LPSM, and WH methods, respectively. □

To gain some further insight into the nature of the filtering operations for the classical decomposition method, we show in Fig. 9.5.4 the magnitude responses of the filters $H_S(\omega)$, $H_T(\omega)$, and $H_I(\omega)$ for extracting the seasonal, trend, and irregular components, as defined by Eq. (9.4.1). The trend filter $H_{\text{trend}}(\omega)$ is given by Eq. (9.5.5), and the comb filter $H_{\text{comb}}(\omega)$ by Eq. (9.3.5) with the phase factor removed to make it symmetric.

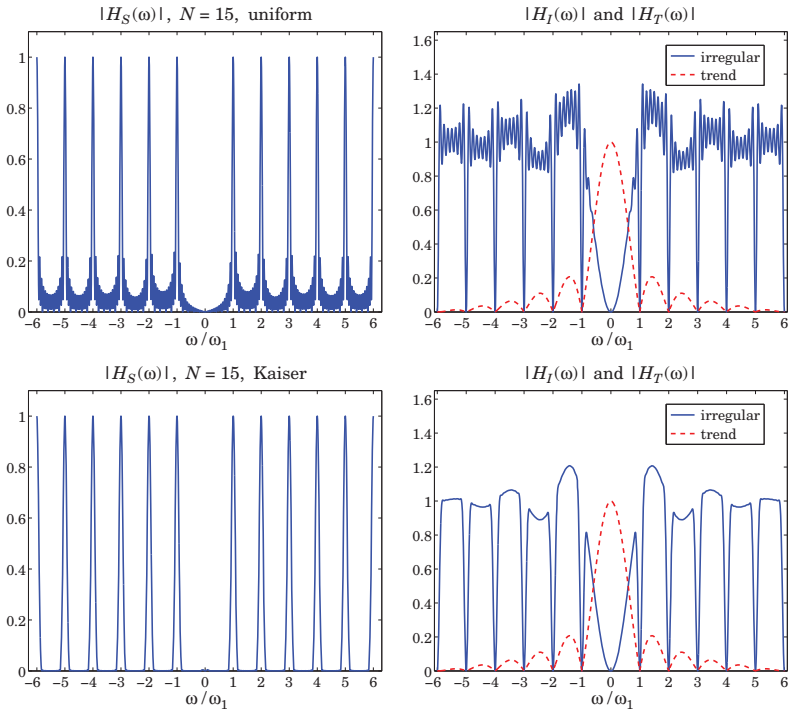


Fig. 9.5.4 Component extraction filters $H_S(\omega)$, $H_T(\omega)$, and $H_I(\omega)$.

The upper graphs in Fig. 9.5.4 show the case of $D = 12$ and $N = 15$. We observe the absence of the harmonic at DC in $H_S(\omega)$. The irregular filter does not quite extract the noise component v_n, but rather a filtered version thereof. Ideally, the irregular filter $H_I(\omega)$ should have zeros at the harmonics, be very small in the passband of $H_T(\omega)$, and be flat between the harmonics. The actual filter $H_I(\omega)$ does approximate these features.

The sidelobe behavior about the harmonics in $H_S(\omega)$, or about the nulls in $H_I(\omega)$, is due to the sidelobes introduced by the signal averaging filter $H_{\text{comb}}(\omega)$ of Eq. (9.3.5), which was obtained by applying the seasonalizing transformation $z \to z^D$ to a length-N FIR filter with uniform weights—the sidelobes being effectively the D-fold replicated versions of the sidelobes of a length-N rectangular window.

Such sidelobes are suppressed only by about 13 dB relative to the main peaks and are quire visible (at the level of $10^{-13/20} = 0.22$). The sidelobes can be suppressed further by replacing the rectangular FIR filter by a length-N windowed version thereof, using for example a Hamming or a Kaiser window. To be precise, the comb filter obtained from a

window $w(n)$, $-M \le n \le M$, where $N = 2M+1$, is defined by

$$W(z) = \sum_{n=-M}^{M} w(n) z^{-n} \quad \Rightarrow \quad H_{\text{comb}}(z) = W(z^D) = \sum_{n=-M}^{M} w(n) z^{-nD} \tag{9.5.8}$$

where $w(n)$ must be normalized to add up to unity. The two lower graphs of Fig. 9.5.4 show the filters obtained from a Kaiser window of length $N = 15$ and sidelobe level $R_{\text{dB}} = 50$ dB. The sidelobes are suppressed to the level of $10^{-50/20} = 0.003$ and are not visible if plotted in absolute scales. The price one pays for suppressing the sidelobes is, of course, the widening of the harmonic peaks. To clarify these ideas, we give below the MATLAB code for generating the graphs in Fig. 9.5.4:

```
D = 12; N = 15;
k = linspace(-6,6,1201); w = 2*pi*k/D;                          % frequency axis

ht = trendma(D);                  Ht = abs(freqz(ht,1,w));      % trend filter Htrend(ω)
hc = up(ones(1,N)/N, D);          Hc = abs(freqz(hc,1,w));      % comb filter Hcomb(ω)
hs = conv(hc, compl(ht));         Hs = abs(freqz(hs,1,w));      % seasonal filter HS(ω)
hi = conv(compl(hs), compl(ht));  Hi = abs(freqz(hi,1,w));      % irregular filter HI(ω)
ha = compl(hs);                   Ha = abs(freqz(ha,1,w));      % seasonal adjustment filter

figure; plot(k, Hs); figure; plot(k, Hi, k,Ht,'r--');           % upper graphs
figure; plot(k, Ha);                                            % left graph in Fig. 9.5.5

Rdb=50; hk = kwindow(N,Rdb); hk = hk/sum(hk);                   % Kaiser window

hc = up(hk, D);                   Hc = abs(freqz(hc,1,w));      % new Hcomb(ω)
hs = conv(hc, compl(ht));         Hs = abs(freqz(hs,1,w));      % new HS(ω)
hi = conv(compl(hs), compl(ht));  Hi = abs(freqz(hi,1,w));      % new HI(ω)
ha = compl(hs);                   Ha = abs(freqz(ha,1,w));      % seasonal adjustment filter

figure; plot(k, Hs); figure; plot(k, Hi, k,Ht,'r--');           % lower graphs
figure; plot(k, Ha);                                            % right graph in Fig. 9.5.5
```

The impulse response definitions in this code implement Eq. (9.4.1) in the time domain. The upsampling function `up` was described in Sec. 9.1. The function `compl` computes the impulse response of the complement of a double-sided symmetric filter, that is, $H(z) \to 1 - H(z)$, or $h_n \to \delta_n - h_n$. The function `kwindow` computes the Kaiser window (for spectral analysis) [604] for a given length N and sidelobe level R_{db} in dB, and it is part of the OSP toolbox.

Fig. 9.5.5 illustrates the complementarity property more clearly by showing the seasonal adjustment filter $H_A(\omega) = 1 - H_S(\omega) = H_T(\omega) + H_I(\omega)$, that is, the filter that removes the seasonal component from the data. As expected, the filter has nulls at the harmonics and is essentially flat in-between.

9.6 Seasonal Moving-Average Filters

Signal averaging can be thought of as ordinary filtering by the seasonalized FIR averager filter of Eq. (9.3.3). However, as we saw in Eq. (9.3.9), the averaged period builds up gradually at the filter output and becomes available only as the last D output points. This is so because the filter length ND is essentially the same as the signal length so

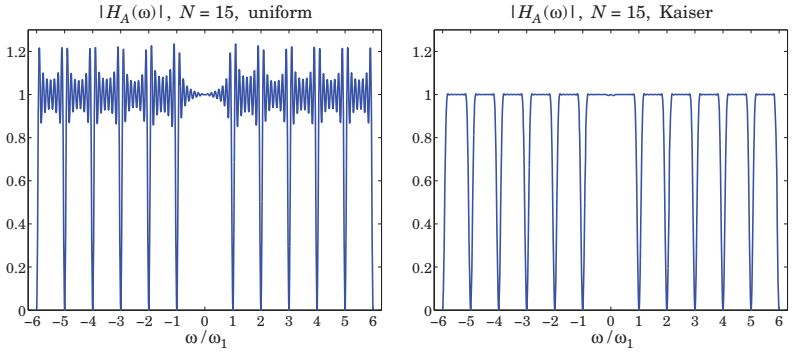


Fig. 9.5.5 Seasonal adjustment filter $H_A(\omega) = 1 - H_S(\omega) = H_T(\omega) + H_I(\omega)$.

that the filter operates mostly in its transient state. Indeed, if L is the length of the signal y_n, the number of periods is $N = \text{floor}(L/D)$ so that $L \approx ND$.

In the classical decomposition method, the final accumulated period is replicated N times to make up the seasonal component s_n. This procedure is appropriate only if s_n is truly periodic. However, in many practical applications s_n is only quasi-periodic with slowly changing periods. In order to be able to estimate s_n more accurately we must use a shorter seasonal moving-average filter that tracks the local (i.e., within the filter's moving window) periodic component.

Example 9.6.1: Fig. 9.6.1 illustrates the filtering point of view for extracting the seasonal part s_n. The same CO_2 data are used as in Example 9.5.2. The classical decomposition method is applied first to determine the trend t_n, and then the residual signal is formed $r_n = y_n - t_n$. In this example, the number of periods contained in the y_n signal is $N = 29$.

The upper-left graph shows the result of ordinary causal filtering of the residual signal r_n by the signal averaging comb filter (9.3.3) using MATLAB's built-in function `filter`. We observe that the transients eventually build up to the same final period as that obtained by signal averaging (shown as the dotted line.)

In the upper-right graph, the residual r_n was filtered by the double-sided filtering function `filtdbl` discussed in Sec. 3.9, which is ordinary causal convolution followed by advancing the result by $(N-1)D/2$ samples. Again, we observe the input-on and input-off transients and the build-up of the correct period at the middle.

The transient portions of the double-sided filter output can be adjusted by using Musgrave's minimum-revision asymmetric filters for the left and right end points. The resulting filter output is shown in the lower-left graph, in which the Musgrave parameter was chosen to be $R = \infty$ (see Sec. 9.8 for more on that.)

The lower-right graph shows the result of filtering r_n through a so-called 3×3 double-sided seasonal moving-average filter, which is discussed below. The MATLAB code for generating these graphs is as follows:

```
Y = loadfile('co2_mm_gl.dat');
t = Y(:,3); y = Y(:,4);                                         % CO2 data
```

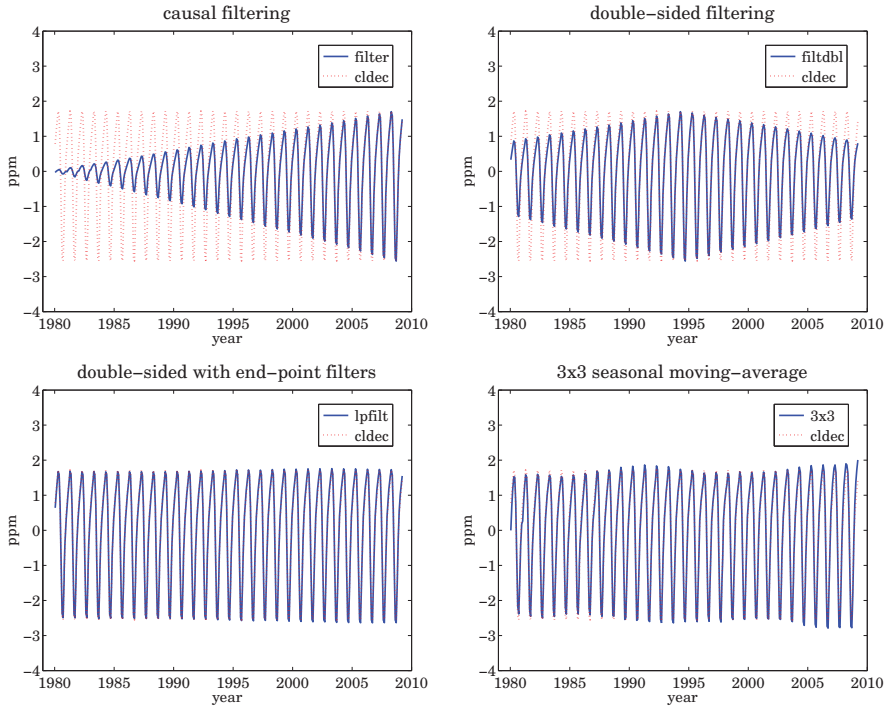


Fig. 9.6.1 Filtering versions of seasonal filter.

```
D=12; N=floor(length(y)/D); M=33; R=inf;      % filter parameters

[yt,ys,yi] = cldec(y,D,R); yr = y - yt;       % yr = residual component r_n

h = ones(1,N)/N;                              % length-N moving-average
hc = up(h, D);                                % seasonalized comb filter obtained from h
Bc = upmat(minrev(h,R), D);                   % seasonalized minimum-revision filter matrix

ys1 = filter(hc,1,yr);                        % ordinary causal filtering by the comb filter hc
ys2 = filtdbl(hc,yr);                         % double-sided filtering
ys3 = lpfilt(Bc, yr);                         % double-sided filtering and end-point filters
[yt4,ys4] = smadec(y, D, M, R);               % ys4 is the 3×3 moving-average output

figure; plot(t,ys1, t,ys,':'); figure; plot(t,ys2, t,ys,':');     % upper graphs
figure; plot(t,ys3, t,ys,':'); figure; plot(t,ys4, t,ys,':');     % lower graphs
```

The `smadec` function is a simple alternative to `cldec` and is discussed below. The function `upmat` upsamples a filter matrix by a factor of D for its use in comb filtering. It upsamples each row and then each column by D and then, it replaces each group of D columns by the corresponding convolution matrix arising from the first column in each group. It can be passed directly into the filtering function `lpfilt`,


```
Bup = upmat(B,D;          % upsampling a filtering matrix
```

For example, the asymmetric filters associated with the 3×3 seasonal moving-average filter [618] are as follows for $D = 3$, where the middle column is the 3×3 filter and the other columns, the asymmetric filters to be used at the ends of the data record, and the function `smat` is described below:

$$B = \text{smat}(1,33) = \frac{1}{27}\begin{bmatrix} 11 & 7 & 3 & 0 & 0 \\ 11 & 10 & 6 & 3 & 0 \\ 5 & 7 & 9 & 7 & 5 \\ 0 & 3 & 6 & 10 & 11 \\ 0 & 0 & 3 & 7 & 11 \end{bmatrix}$$

$$B_{\text{up}} = \text{upmat}(B,3) = \frac{1}{27}\begin{bmatrix} 11 & 0 & 0 & 7 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 11 & 0 & 0 & 7 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 11 & 0 & 0 & 7 & 0 & 0 & 3 & 0 & 0 & 0 & 0 \\ 11 & 0 & 0 & 10 & 0 & 0 & 6 & 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 11 & 0 & 0 & 10 & 0 & 0 & 7 & 0 & 0 & 5 & 0 & 0 \\ 0 & 0 & 11 & 0 & 0 & 10 & 0 & 0 & 7 & 0 & 0 & 5 & 0 \\ 5 & 0 & 0 & 7 & 0 & 0 & 9 & 0 & 0 & 7 & 0 & 0 & 5 \\ 0 & 5 & 0 & 0 & 7 & 0 & 0 & 10 & 0 & 0 & 11 & 0 & 0 \\ 0 & 0 & 5 & 0 & 0 & 7 & 0 & 0 & 10 & 0 & 0 & 11 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 & 6 & 0 & 0 & 10 & 0 & 0 & 11 \\ 0 & 0 & 0 & 0 & 3 & 0 & 0 & 7 & 0 & 0 & 11 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 7 & 0 & 0 & 11 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 7 & 0 & 0 & 11 \end{bmatrix} \tag{9.6.1}$$

The simple 3×3, 3×5, and 3×9 seasonal moving-average filters are widely used in de-seasonalizing business, government, and census data. They are obtained by symmetrizing the $N_1 \times N_2$ filters of Eq. (9.5.2) and then applying the transformation $z \to z^D$. For example, the resulting 3×3 and 3×5 comb filters are:

$$\begin{aligned} H_{33}(z) &= \frac{1}{3}(z^D + 1 + z^{-D}) \cdot \frac{1}{3}(z^D + 1 + z^{-D}) \\ H_{35}(z) &= \frac{1}{3}(z^D + 1 + z^{-D}) \cdot \frac{1}{5}(z^{2D} + z^D + 1 + z^{-D} + z^{-2D}) \end{aligned} \tag{9.6.2}$$

with symmetric impulse responses,

$$\begin{aligned} \mathbf{h}_{33} &= \frac{1}{9}[1, \underbrace{0, \ldots, 0}_{D-1 \text{ zeros}}, 2, 0, \ldots, 0, 3, 0, \ldots, 0, 2, 0, \ldots, 0, 1] \\ \mathbf{h}_{35} &= \frac{1}{15}[1, 2, 0, \ldots, 0, 3, 0, \ldots, 0, 3, 0, \ldots, 0, 3, 0, \ldots, 0, 2, 0, \ldots, 0, 1] \end{aligned} \tag{9.6.3}$$

The MATLAB function `smav` calculates such impulse responses,

```
h = smav(N1,N2,D);        % seasonal moving-average filters
```

It is simply:

```
h  = up(conv(ones(1,N1), ones(1,N2))/(N1*N2), D);
```

These filters are to be applied to the residual signal $r_n = y_n - t_n$. Their end-point effects can be handled by using Musgrave's minimum-revision filters or by any other appropriate asymmetric filters. In fact, the census X-11/X-12 methods use asymmetric filters that are specially constructed for the 3×3, 3×5, and 3×9 filters, and may be found in Ref. [618]. They have been incorporated into the `smadec` and `x11dec`. For example, Eq. (9.6.1) shows the 3×3 filter matrix before and after it is upsampled.

To summarize, the filtering approach for de-seasonalizing a signal $y_n = s_n + t_n + v_n$ with period D consists of the following two basic steps:

1. Apply a lowpass filter to extract the trend component t_n, incorporating also asymmetric end-point filters. The trend-extraction filter can be a simple $1\times D$ or $2\times D$ moving average, or, any other lowpass filter such as a local-polynomial or Whittaker-Henderson smoother.
2. Apply a comb filter to the de-trended residual signal $r_n = y_n - t_n$ to extract the seasonal part s_n, incorporating asymmetric filters for the end-points. The comb filter can be a simple seasonalized 3×3, 3×5, or 3×9 lowpass filter, or a more general seasonalized filter such as one obtained from a non-rectangular window. The de-seasonalized, or seasonally adjusted, signal is then $a_n = y_n - s_n$.

The MATLAB function `smadec` carries out this program using the simple $1\times D$ or $2\times D$ moving-average filter for de-trending and the 3×3, 3×5, or 3×9 comb filters for the seasonal part. It has usage:

```
[yt,ys,yi] = smadec(y,D,M,R,iter,type);   % seasonal moving-average decomposition
```

where `yt, ys, yi` are the estimated components t_n, s_n, v_n, and `y` is the input data vector. The integer values $M = 33, 35, 39$ select the 3×3, 3×5, or 3×9 seasonal comb filters, other values of M can also be used. The Musgrave parameter defaults to $R = \infty$, the parameter `iter` specifies the number of iterations of the filtering process, which correspond to applying the trend filter `iter` times. The string `type` takes on the values `'a', 'm'` for additive or multiplicative decomposition. To clarify the operations, we give below the essential part of the code in `smadec` for the additive case:

```
F = minrev(trendma(D),R);      % trend moving-average, with minimum-revision end-filters
B = smat(D,M,R);               % seasonal moving averages, with end-filters

yt = y;                        % initialize iteration
for i=1:iter,
  yt = lpfilt(F,yt);           % T component
  yr = y - yt;                 % S+I component
  ys = lpfilt(B,yr);           % S component
  yi = yr - ys;                % I component
end
```

The function `smat` generates the filtering matrix of the seasonalized comb filters, including the specific asymmetric filters for the 3×3, 3×5, or 3×9 cases, as well for other cases.

Example 9.6.2: *Unemployment Data 1965–1979.* The data set representing the monthly number of unemployed 16–19 year old men for the period Jan. 1965 to Dec. 1979 has served as a benchmark for comparing seasonal adjustment methods [633,637]. The data set is available from the US Bureau of Labor Statistics web site: `http://www.bls.gov/data/` (series ID: `LNU03000013`, under category: Unemployment > Labor Force Statistics > on-screen data search).

The upper graphs of Fig. 9.6.2 illustrate the application of the `smadec` function using $D = 12$, $M = 35$ (which selects the 3×5 comb), one iteration, Musgrave parameter $R = \infty$ for the $2\times D$ trend filter, and additive decomposition type. The left graph shows the trend t_n and the right, the estimated seasonal component s_n, which is not exactly periodic but exhibits quasi-periodicity. The results are comparable to those of Refs. [633,637].

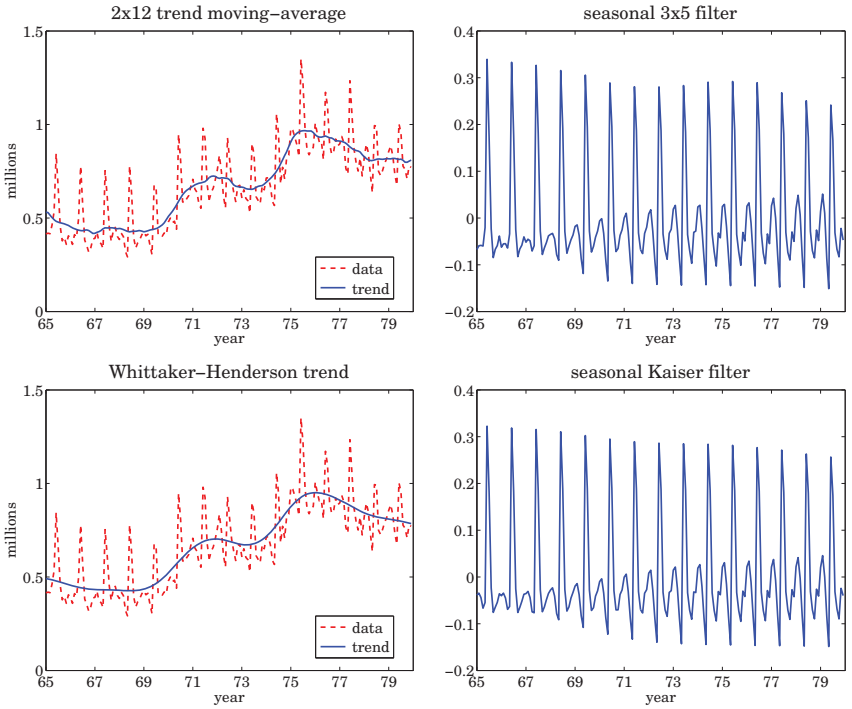


Fig. 9.6.2 Trend/seasonal decomposition of monthly unemployment data for 1965–1979.

The lower graphs show the decomposition obtained by de-trending using a Whittaker-Henderson smoother of order $s = 2$, followed by a Kaiser comb filter. The function `whgcv` was used to determine the optimum smoothing parameter, $\lambda_{\text{opt}} = 2039$. The Kaiser window had length $N = 15$ and relative sidelobe level of $R_{\text{db}} = 50$ dB. The MATLAB code for generating these graphs was as follows:

```
Y = loadfile('unemp-1619-nsa.dat');          % data file in OSP toolbox
i=find(Y==1965); Y = Y(i:i+14,2:13)';        % extract 1965-1979 data
```

```
y = Y(:)/1000; t = taxis(y,12,65);                    % y units in millions

D=12; M=35; R=inf; iter=1; type='a';                  % smadec input parameters

[yt,ys,yi] = smadec(y,D,M,R,iter,type);               % yt,ys represent t_n, s_n

figure; plot(t,y, t,yt); figure; plot(t,ys);          % upper graphs

s = 2; la = 2000:2050;                                % search range of λ's
[gcv,lopt] = whgcv(y,la,s);                           % optimum λ_opt = 2039

yt = whsm(y,lopt,s);                                  % extract t_n component
yr = y-yt;                                            % residual S+I component

Rdb=50; N=15; h = kwindow(N,Rdb); hk = h/sum(h);      % Kaiser window
B = upmat(minrev(hk,R), D);                           % Kaiser comb with end-filters

ys = lpfilt(B, yr);                                   % extract s_n component

figure; plot(t,y, t,yt); figure; plot(t,ys);          % bottom graphs
```

The Whittaker-Henderson method results in a smoother trend. However, the trend from `smadec` can be made equally smooth by increasing the number of iterations, for example, setting `iter=3`. The frequency responses of the various filters are shown in Eq. (9.6.3).

The filters $H_T(\omega), H_S(\omega)$ for extracting the trend and seasonal components, and the seasonal-adjustment filter $H_A(\omega) = 1 - H_S(\omega)$ are constructed from Eq. (9.4.1). The MATLAB code for generating these graphs was:

```
k = linspace(-6,6,1201); w = 2*pi*k/12;               % frequency range [−π,π]

ht = trendma(12);          Ht = abs(freqz(ht,1,w));   % 2×12 trend filter
hc = smav(3,5,12);                                    % upsampled 3×5 comb filter
hs = conv(hc,compl(ht));   Hs = abs(freqz(hs,1,w));   % seasonal filter, H_S(ω)
ha = compl(hs);            Ha = abs(freqz(ha,1,w));   % adjustment filter, 1 − H_S(ω)

figure; plot(k,Hs, k,Ht,'--'); figure; plot(k,Ha);    % upper graphs

Ht = 1 ./ (1 + lopt * (2*sin(w/2)).^(2*s));           % Whittaker-Henderson trend filter
hc = up(hk,12);                                       % Kaiser comb impulse response
Hc = freqz(hc,1,w) .* exp(j*(N-1)*D*w/2);             % Kaiser comb frequency response
Hs = Hc .* (1-Ht); Ha = 1 - Hs;                       % H_S(ω) and H_A(ω) = 1 − H_S(ω)
Hs = abs(Hs); Ha = abs(Ha);

figure; plot(k,Hs, k,Ht,'--'); figure; plot(k,Ha);    % bottom graphs
```

The Whittaker-Henderson trend filter was computed using Eq. (8.2.7). The frequency response of the Kaiser comb filter was multiplied by $e^{j\omega(N-1)D/2}$ to make the filter symmetric. It is evident that the WH/Kaiser filters perform better. □

The steps implementing the Whittaker-Henderson/Kaiser decomposition have been incorporated into the MATLAB function `whkdec`,

```
[yt,ys,yi] = whkdec(y,D,s,la,N,Rdb,R,type);           % WH/Kaiser decomposition
```

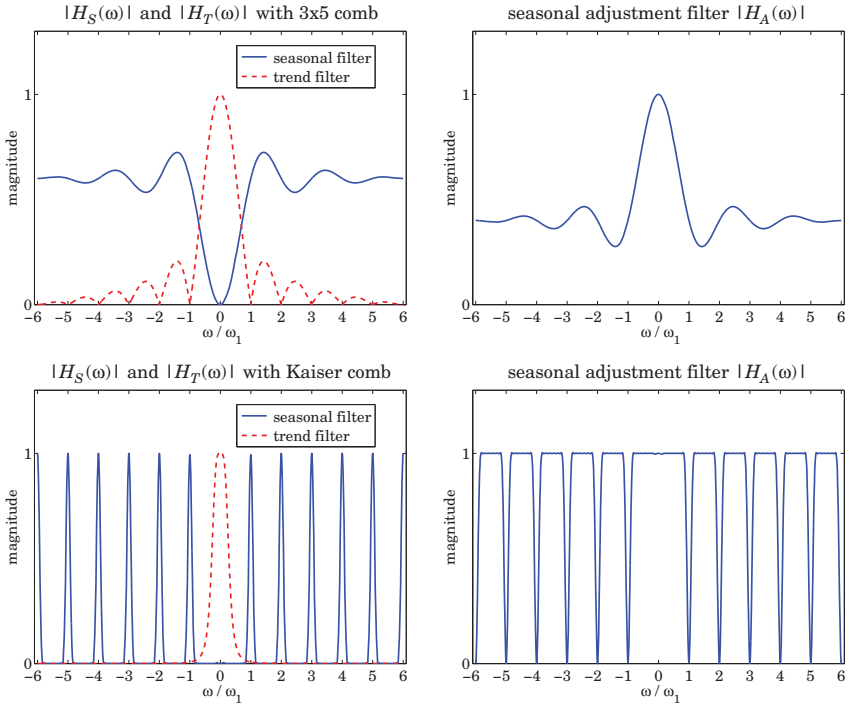



Fig. 9.6.3 Frequency responses of trend, seasonal, and seasonal-adjustment filters.

The WH parameters `s,la` must be selected in advance, for example, λ can be tentatively estimated using the GCV function `whgcv`, but it should be noted that the GCV does not always give a "good" value for λ. The Kaiser window length `N` must be odd and the sidelobe level must be restricted to the range $[13, 120]$ dB. The Musgrave parameter R affects only the Kaiser comb filter because the WH trend already takes into account the end points. The parameter `type` is as in the function `smadec`.

9.7 Census X-11 Decomposition Filters

The Census X-11/X-12 seasonal adjustment procedures have become a standard for deseasonalizing economic data [605–621]. They are based on a series of filtering operations that represent a refined version of the procedures outlined in the previous section.

Here, we only discuss the relevant filtering operations, leaving out details such as adjustments for outliers or calendar effects. The most recent version, X-12-ARIMA, is available from the web site [607]. The web pages [608,609] contain a number of papers on the development of the X-11/X-12 methods.

As outlined in [610,613], the X-11 method involves the repeated application of the 2×12 trend filter of Eq. (9.5.1), the 3×3 and 3×5 comb filters of Eq. (9.6.2), and the Henderson filters of lengths 9, 13, or 25, with polynomial and smoothing orders $d = s = 3$ given by Eq. (4.2.29) of Chap. 4. The basic X-11 filtering steps are as follows, assuming an additive model $y_n = s_n + t_n + v_n$,

1. Apply a 2×12 trend filter to y_n to get a preliminary estimate of the trend t_n.
2. Subtract t_n from y_n to get a preliminary estimate of the residual $r_n = y_n - t_n$.
3. Apply the 3×3 comb filter to r_n to get a preliminary estimate of s_n.
4. Get an improved s_n by removing its filtered version by the 2×12 trend filter.
5. Subtract s_n from y_n to get a preliminary adjusted signal $a_n = y_n - s_n = t_n + v_n$.
6. Filter a_n by a Henderson filter to get an improved estimate of the trend t_n.
7. Subtract t_n from y_n to get an improved residual $r_n = y_n - t_n$.
8. Apply the 3×5 comb filter to r_n to get an improved estimate of s_n.
9. Get the final s_n by removing its filtered version by the 2×12 trend filter.
10. Subtract s_n from y_n to get the final adjusted signal $a_n = y_n - s_n = t_n + v_n$.
11. Filter a_n by a Henderson filter to get the final estimate of the trend t_n.
12. Subtract t_n from a_n to get the final estimate of the irregular component v_n.

These steps are for monthly data. For quarterly data, replace the 2×12 trend filter by a 2×4 filter. The steps can be expressed concisely in the z-domain as follows:

$$
\begin{aligned}
&1.\quad Y_T^{\text{pre}} = FY \\
&2.\quad Y_R^{\text{pre}} = Y - Y_T^{\text{pre}} = (1-F)Y \\
&3.\quad Y_S^{\text{pre}} = H_{33}Y_R^{\text{pre}} = H_{33}(1-F)Y \\
&4.\quad Y_S^{\text{imp}} = Y_S^{\text{pre}} - FY_S^{\text{pre}} = H_{33}(1-F)^2Y \\
&5.\quad Y_A^{\text{pre}} = Y - Y_S^{\text{imp}} = \big[1 - H_{33}(1-F)^2\big]Y \\
&6.\quad Y_T^{\text{imp}} = HY_A^{\text{pre}} = H\big[1 - H_{33}(1-F)^2\big]Y \\
&7.\quad Y_R^{\text{imp}} = Y - Y_T^{\text{imp}} = \Big[1 - H\big[1 - H_{33}(1-F)^2\big]\Big]Y \\
&8.\quad Y_S^{\text{imp}} = H_{35}Y_R^{\text{imp}} = H_{35}\Big[1 - H\big[1 - H_{33}(1-F)^2\big]\Big]Y \\
&9.\quad Y_S = Y_S^{\text{imp}} - FY_S^{\text{imp}} = (1-F)H_{35}\Big[1 - H\big[1 - H_{33}(1-F)^2\big]\Big]Y \equiv H_S Y \\
&10.\quad Y_A = Y - Y_S = (1-H_S)Y \equiv H_A Y \\
&11.\quad Y_T = HY_A = H(1-H_S)Y \equiv H_T Y \\
&12.\quad Y_I = Y_A - Y_T = (1-H)(1-H_S)Y \equiv H_I Y
\end{aligned}
\tag{9.7.1}
$$

where $Y_T^{\text{pre}} = FY$ stands for $Y_Y^{\text{pre}}(z) = F(z)Y(z)$, etc., and $F(z)$, $H_{33}(z)$, $H_{35}(z)$, $H(z)$ denote the 2×12 trend filter, the 3×3 and 3×5 comb filters, and the Henderson filter, and the z-transforms of the data, trend, seasonal, adjusted, and irregular components are denoted by $Y(z)$, $Y_T(z)$, $Y_S(z)$, $Y_A(z)$, and $Y_I(z)$.

It follows that the effective filters for extracting the seasonal, seasonally-adjusted, trend, and irregular components are:

$$
\begin{aligned}
H_S &= (1-F)H_{35}\Big[1 - H\big[1 - H_{33}(1-F)^2\big]\Big] && \text{(seasonal)} \\
H_A &= 1 - H_S && \text{(seasonally-adjusted)} \\
H_T &= H(1-H_S) && \text{(trend)} \\
H_I &= (1-H)(1-H_S) && \text{(irregular)}
\end{aligned}
\tag{9.7.2}
$$

They satisfy the complementarity property $H_T(z) + H_S(z) + H_I(z) = 1$.

Example 9.7.1: *X-11 Filters.* The construction of the time-domain impulse responses of the X-11 decomposition filters (9.7.2) is straightforward. For example, the following MATLAB code evaluates the impulse responses (using a 13-term Henderson filter), as well as the corresponding frequency responses shown in Fig. 9.7.1,

```
k = linspace(-6,6,1201); w = 2*pi*k/12;      % frequency axis −π ≤ ω ≤ π

hf  = trendma(12);                           % 2×12 trend filter, F
hfc = compl(hf);                             % complement of trend filter, 1 − F
h33 = smav(3,3,12);                          % 3×3 comb filter, H33
h35 = smav(3,5,12);                          % 3×5 comb filter, H35
N=13; he = lprs2(N,3,3);                     % 13-term Henderson filter, H
g = conv(hfc,hfc);                           % G = (1 − F)^2, G is temporary variable
g = compl(conv(h33, g));                     % G = 1 − H33(1 − F)^2
g = compl(conv(he,g));                       % G = 1 − H[1 − H33(1 − F)^2]
g = conv(h35,g);                             % G = H35{1 − H[1 − H33(1 − F)^2]}
                                             % HS = (1 − F)H35{1 − H[1 − H33(1 − F)^2]}
hs = conv(hfc,g);          Hs = abs(freqz(hs,1,w));    % seasonal
ha = compl(hs);            Ha = abs(freqz(ha,1,w));    % adjustment
ht = conv(he,ha);          Ht = abs(freqz(ht,1,w));    % trend
hi = conv(compl(he),ha);   Hi = abs(freqz(hi,1,w));    % irregular

figure; plot(k, Hs); figure; plot(k, Ht);              % upper graphs
figure; plot(k, Ha); figure; plot(k, Hi);              % lower graphs
```

We note that the filters have the expected shapes. All cases described in [613] can be generated by variations of this code. □

The MATLAB function `x11dec` implements the above steps, amended by the use of asymmetric filters to handle the end-points of the time series,

```
[yt,ys,yi] = x11dec(y,D,M1,M2,N1,N2,R,type);      % X-11 decomposition method
```

where `D` is the seasonal period, `M1,M2` the sizes of the first and second comb filters (entered as 33, 35, or 39), `N1,N2` are the lengths of the first and second Henderson filters, `R` is the Musgrave minimum-revision parameter affecting both the Henderson filters and the trend filter, and `type` designates an additive or multiplicative decomposition. The Musgrave parameter R is usually assigned the following values, depending on the length

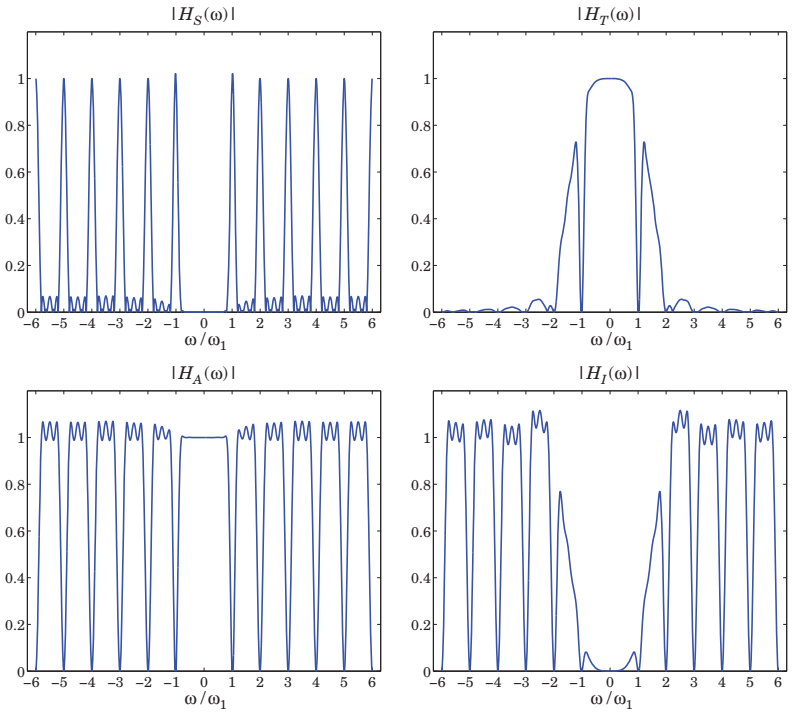


Fig. 9.7.1 X-11 decomposition filters (with 13-term Henderson).

N of the Henderson filter [618]:

N	R
5	0.001
7	4.5
9	1.0
13	3.5
23	4.5

(9.7.3)

Example 9.7.2: *Unemployment Data 1980–2008.* Fig. 9.7.2 shows the X-11 decomposition of the monthly unemployment data for 20 year and older men for the period Jan. 1980 to Dec. 2008. The data are from the US BLS web site: `http://www.bls.gov/data/`, series `LNU03000025`, under category: Unemployment > Labor Force Statistics > on-screen data search. The already seasonally adjusted data are also available as series `LNS13000025`.

The upper-left graph shows the original data and the extracted trend t_n assuming an additive model $y_n = t_n + s_n + v_n$. The lower-left graph is the extracted seasonal component s_n. The upper-right graph shows the seasonally-adjusted signal $a_n = y_n - s_n = t_n + v_n$ to be compared with that of the lower-right graph, which shows the already available adjusted signal— the two agreeing fairly well. The graphs were generated by the following code:

```
Y = loadfile('unemp-20-nsa.dat');          % not-seasonally adjusted data
```

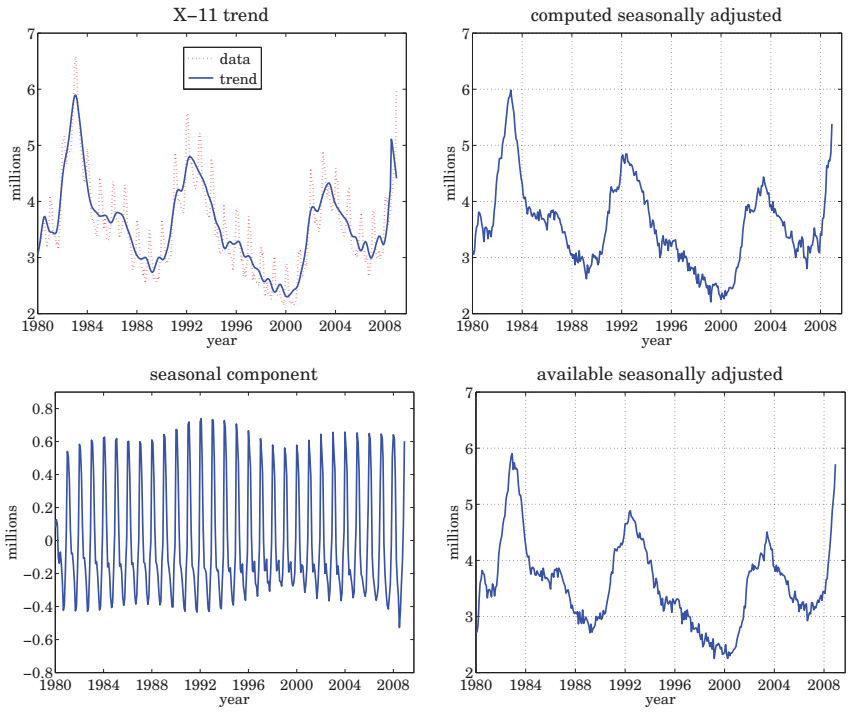



Fig. 9.7.2 X-11 decomposition of unemployment data 1980-2008.

```
i = find(Y==1980); Y = Y(i:end,2:13)';      % select years 1980-2008
y = Y(:)/1000; t = taxis(y,12,1980);       % data vector y, and time axis
                                           % data sets available in the OSP toolbox
Y = loadfile('unemp-20-sa.dat');           % seasonally adjusted data
i = find(Y==1980); Y = Y(i:end,2:13)';
yadj = Y(:)/1000;                          % yadj = already available adjusted data

D=12; M1=33; M2=35; N1=13; N2=13; R=3.5; type='a';    % X-11 parameters
[yt,ys,yi] = x11dec(y,D,M1,M2,N1,N2,R,type);          % X-11 method
ya = y-ys;                                            % seasonally adjusted

%s = 2; la = 1000; N=15; Rdb=50;                      % WH/K parameters
%[yt,ys,yi] = whkdec(y,D,s,la,N,Rdb,R,type);          % Whittaker-Henderson/Kaiser
%ya = y-ys;                                           % seasonally adjusted

figure; plot(t,y,':', t,yt,'-'); figure; plot(t,ya);  % upper graphs
figure; plot(t,ys); figure; plot(t,yadj);             % lower graphs
```

The value of R was 3.5 because a 13-term Henderson filter was used. The purpose of this example was to compare the performance of our simplified X-11 implementation with the results that are already available from the Bureau of Labor Statistics. We note that the use of the Whittaker-Henderson/Kaiser decomposition method also works comparably well,

for example with parameters $s = 2$, $\lambda = 1000$, Kaiser length $N = 15$, and $R_{\text{db}} = 50$ dB. The code for that is included above but it is commented out. □

9.8 Musgrave Asymmetric Filters

The handling of the end-point problem by the use of asymmetric filters was discussed in Sec. 3.9. We saw that the output y_n of filtering a length-L signal x_n, $0 \le n \le L-1$, by a double-sided filter h_m, $-M \le m \le M$, using for example the function `filtdbl`, consists of M initial and M final transient output samples, and $L - 2M$ steady-state samples, the latter being computed by the steady-state version of the convolutional equation:

$$y_n = \sum_{m=-M}^{M} h_m x_{n-m}\,, \quad M \le n \le L-1-M \tag{9.8.1}$$

The overall operation can be cast in convolution matrix form. For example, for $L = 8$ and $M = 2$, we have:

$$\begin{bmatrix} y_0 \\ y_1 \\ \dots \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ \dots \\ y_6 \\ y_7 \end{bmatrix} = \begin{bmatrix} h_0 & h_{-1} & h_{-2} & 0 & 0 & 0 & 0 & 0 \\ h_1 & h_0 & h_{-1} & h_{-2} & 0 & 0 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ h_2 & h_1 & h_0 & h_{-1} & h_{-2} & 0 & 0 & 0 \\ 0 & h_2 & h_1 & h_0 & h_{-1} & h_{-2} & 0 & 0 \\ 0 & 0 & h_2 & h_1 & h_0 & h_{-1} & h_{-2} & 0 \\ 0 & 0 & 0 & h_2 & h_1 & h_0 & h_{-1} & h_{-2} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 0 & h_2 & h_1 & h_0 & h_{-1} \\ 0 & 0 & 0 & 0 & 0 & h_2 & h_1 & h_0 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \dots \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ \dots \\ x_6 \\ x_7 \end{bmatrix} \tag{9.8.2}$$

The middle $L - 2M = 4$ output samples are steady, while the first and last $M = 2$ are transient and are computed by using fewer filter weights that the steady ones. The transient and steady filters can be arranged into a matrix B, which is for the above example,

$$B = \begin{bmatrix} h_0 & h_1 & h_2 & 0 & 0 \\ h_{-1} & h_0 & h_1 & h_2 & 0 \\ h_{-2} & h_{-1} & h_0 & h_1 & h_2 \\ 0 & h_{-2} & h_{-1} & h_0 & h_1 \\ 0 & 0 & h_{-2} & h_{-1} & h_0 \end{bmatrix} \tag{9.8.3}$$

The convolution matrix H can be built from the knowledge of B as described in Sec. 3.9. The matrix B conveniently summarizes the relevant filters and can be used as an input to the filtering function `lpfilt`.

As discussed in Sec. 3.9, local polynomial smoothing filters, including Henderson filters, generate their own matrix B to handle the series end-points, with the non-central columns of B consisting of the corresponding prediction filters.

However, when one does not have available such prediction filters, but only the central filter h_m, $-M \le m \le M$, one must use appropriately designed end-point filters.

For example, Eq. (9.8.2) would change to:

$$\begin{bmatrix} y_0 \\ y_1 \\ \dots \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ \dots \\ y_6 \\ y_7 \end{bmatrix} = \begin{bmatrix} f_0^0 & f_{-1}^0 & f_{-2}^0 & 0 & 0 & 0 & 0 & 0 \\ f_1^1 & f_0^1 & f_{-1}^1 & f_{-2}^1 & 0 & 0 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ h_2 & h_1 & h_0 & h_{-1} & h_{-2} & 0 & 0 & 0 \\ 0 & h_2 & h_1 & h_0 & h_{-1} & h_{-2} & 0 & 0 \\ 0 & 0 & h_2 & h_1 & h_0 & h_{-1} & h_{-2} & 0 \\ 0 & 0 & 0 & h_2 & h_1 & h_0 & h_{-1} & h_{-2} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 0 & g_2^1 & g_1^1 & g_0^1 & g_{-1}^1 \\ 0 & 0 & 0 & 0 & 0 & g_2^0 & g_1^0 & g_0^0 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \dots \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ \dots \\ x_6 \\ x_7 \end{bmatrix} \tag{9.8.4}$$

where the filters f_m^0 and f_m^1 are used for computing the first two transient outputs y_0, y_1, and the filters g_m^0 and g_m^1 are for the last two outputs y_7, y_6. The corresponding B matrix would be in this case:

$$B = \begin{bmatrix} f_0^0 & f_1^1 & h_2 & 0 & 0 \\ f_{-1}^0 & f_0^1 & h_1 & g_2^1 & 0 \\ f_{-2}^0 & f_{-1}^1 & h_0 & g_1^1 & g_2^0 \\ 0 & f_{-2}^1 & h_{-1} & g_0^1 & g_1^0 \\ 0 & 0 & h_{-2} & g_{-1}^1 & g_0^0 \end{bmatrix} \tag{9.8.5}$$

More generally, the filters f_m^i, g_m^i, $i = 0, 1, \dots, M-1$, compute the first M and last M output samples y_i, y_{L-1-i}, respectively, through the convolutional equations:

$$\begin{aligned} y_i &= \sum_{m=-M}^{i} f_m^i x_{i-m} = \sum_{m=-i}^{M} f_{-m}^i x_{i+m} = f_i^i x_0 + \cdots + f_0^i x_i + \cdots + f_{-M}^i x_{i+M} \\ y_{L-1-i} &= \sum_{m=-i}^{M} g_m^i x_{L-1-i-m} = g_M^i x_{L-1-i-M} + \cdots + g_0^i x_{L-1-i} + \cdots + g_{-i}^i x_{L-1} \end{aligned} \tag{9.8.6}$$

for $i = 0, 1, \dots, M-1$, where the limits of summations follow by the requirement that only available x_n samples appear in the sums.

Musgrave's method [615,616] constructs such asymmetric filters from the knowledge only of the central filter h_m. The construction applies to filters h_m that are symmetric, $h_m = h_{-m}$, and are normalized to unity gain at DC, such as lowpass trend filters,

$$\sum_{m=-M}^{M} h_m = 1 \tag{9.8.7}$$

The asymmetric filters f_m^i, g_m^i are required to satisfy similar moment constraints:

$$\sum_{m=-M}^{i} f_m^i = 1\,, \qquad \sum_{m=-i}^{M} g_m^i = 1 \tag{9.8.8}$$

The design is based on a minimum-revision criterion. When the data record has length L, the ith output from the end, y_{L-1-i}, is computed with the filter g_m^i. If or

when the series is extended to length $L-1+M$, then the same output can actually be computed with the symmetric filter h_m resulting in a revised output y^{rev}_{L-1-i}, that is,

$$y_{L-1-i} = \sum_{m=-i}^{M} g^i_m x_{L-1-i-m}\,, \quad y^{\text{rev}}_{L-1-i} = \sum_{m=-M}^{M} h_m x_{L-1-i-m}$$

Musgrave's criterion selects g^i_m to minimize the mean-square revision error $E[e^2_{L-1-i}]$, where $e_{L-1-i} = y_{L-1-i} - y^{\text{rev}}_{L-1-i}$, under the assumption that locally the input series is linear, that is, $x_{L-1-i-m} = a + bm + v_m$, with a, b constant parameters, and v_m zero-mean white noise with variance σ^2. The mean-square error becomes then,

$$\begin{aligned} E[e^2_{L-1-i}] &= E\Big[\Big[\sum_{m=-i}^{M} g^i_m(a+bm+v_m) - \sum_{m=-M}^{M} h_m(a+bm+v_m)\Big]^2\Big] \\ &= E\Big[\Big[b\sum_{m=-i}^{M} m g^i_m + \sum_{m=-i}^{M}(g^i_m - h_m)v_m - \sum_{m=-M}^{-i-1} h_m v_m\Big]^2\Big] \\ &= \sigma^2 \sum_{m=-i}^{M}(g^i_m - h_m)^2 + b^2\Big(\sum_{m=-i}^{M} m g^i_m\Big)^2 + \text{const.} \end{aligned} \tag{9.8.9}$$

where "const." is a positive term independent of g^i_m. In deriving this, we used the moment constraints (9.8.7) and (9.8.8), and the property $\sum_{m=-M}^{M} m h_m = 0$, which follows from the assumed symmetry of h_m. Defining the constant $\beta^2 = b^2/\sigma^2$, it follows that the optimum filter g^i_m will be the solution of the following optimization criterion, which incorporates the constraint (9.8.8) by means of a Lagrange multiplier λ:

$$\mathcal{J} = \sum_{m=-i}^{M}(g_m - h_m)^2 + \beta^2\Big(\sum_{m=-i}^{M} m g_m\Big)^2 + \lambda\Big(1 - \sum_{m=-i}^{M} g_m\Big) = \min \tag{9.8.10}$$

In a similar fashion, we can show that the filters f^i_m are the solutions of

$$\mathcal{J} = \sum_{m=-M}^{i}(f_m - h_m)^2 + \beta^2\Big(\sum_{m=-M}^{i} m f_m\Big)^2 + \lambda\Big(1 - \sum_{m=-M}^{i} f_m\Big) = \min \tag{9.8.11}$$

Because h_m is even in m it follows (by changing variables $m \to -m$ in the sums) that $f^i_m = g^i_{-m}$, that is, the beginning filters are the reverse of the end filters. Thus, only g^i_m need be determined and is found to be [616]:

$$g^i_m = h_m + \frac{A_i}{M+i+1} + \frac{\beta^2 B_i}{D_i}(m - \mu_i)\,, \quad -i \le m \le M \tag{9.8.12}$$

for $i = 0, 1, \ldots, M-1$, with the constants A_i, B_i, D_i, μ_i defined by,

$$\begin{aligned} A_i &= \sum_{m=-M}^{-i-1} h_m\,, \quad B_i = \sum_{m=-M}^{-i-1}(m - \mu_i)h_m\,, \quad i = 0, 1, \ldots, M-1 \\ \mu_i &= \frac{M-i}{2}\,, \quad D_i = 1 + \frac{\beta^2}{12}(M+i)(M+i+1)(M+i+2) \end{aligned} \tag{9.8.13}$$

To show Eq. (9.8.12), we set the gradient of $\mathcal{J}$ in (9.8.10) to zero, $\partial\mathcal{J}/\partial g_m = 0$, to get,

$$g_m = h_m + \lambda - \beta^2 G m\,, \quad G = \sum_{m=-i}^{M} m g_m \tag{9.8.14}$$

Summing up over m and using the constraint (9.8.8), and then, multiplying by m and summing up over m, results in two equations for the two unknowns λ, G:

$$\begin{aligned} 1 &= \sum_{m=-i}^{M} h_m + \Big(\sum_{m=-i}^{M} 1\Big)\lambda - \Big(\sum_{m=-i}^{M} m\Big)\beta^2 G \\ G &= \sum_{m=-i}^{M} m h_m + \Big(\sum_{m=-i}^{M} m\Big)\lambda - \Big(\sum_{m=-i}^{M} m^2\Big)\beta^2 G \end{aligned} \tag{9.8.15}$$

Using the properties,

$$1 - \sum_{m=-i}^{M} h_m = \sum_{m=-M}^{-i-1} h_m\,, \quad \sum_{m=-i}^{M} m h_m = -\sum_{m=-M}^{-i-1} m h_m$$

and the identities,

$$\begin{aligned} \sum_{m=-i}^{M} 1 &= M + i + 1 \\ \sum_{m=-i}^{M} m &= \frac{1}{2}(M+i+1)(M-i) = (M+i+1)\mu_i \\ \sum_{m=-i}^{M} m^2 &= (M+i+1)\mu_i^2 + \frac{1}{12}(M+i)(M+i+1)(M+i+2) \end{aligned} \tag{9.8.16}$$

and solving Eqs. (9.8.15) for the constants λ, G and substituting them in (9.8.14), gives the solution (9.8.12). The parameter β is usually computed in terms of the Musgrave parameter R, the two being related by

$$R^2 = \frac{4}{\pi\beta^2} \quad \Rightarrow \quad \beta^2 = \frac{4}{\pi R^2} \tag{9.8.17}$$

The MATLAB function `minrev` implements Eq. (9.8.12) and arranges the asymmetric filters into a filtering matrix B, which can be passed into the filtering function `lpfilt`,

```
B = minrev(h,R);      % minimum-revision asymmetric filters
```

The input is any odd-length symmetric filter h_m and the parameter R. Typical values of R are given in Eq. (9.7.3). The value $R = \infty$ corresponds to slope $\beta = 0$. For $R = 0$ or $\beta = \infty$, the limit of the solution (9.8.12) is ignored and, instead, the function `minrev` generates the usual convolutional transients for the filter h_m, resulting in a matrix B such that in Eqs. (9.8.3).

We have made extensive use of this function since Chap. 3. As a further example, we compare the filtering matrix B for a 7-term Henderson filter resulting from `minrev`

with the standard value $R = 4.5$ to that resulting from the function `lprs` using the corresponding prediction filters for the same Henderson filter:

$$h = \text{lprs2}(7,3,3) = [-0.0587,\ 0.0587,\ 0.2937,\ 0.4126,\ 0.2937,\ 0.0587,\ -0.0587]$$

$$B = \text{minrev}(h, 4.5) = \begin{bmatrix} 0.5345 & 0.2892 & 0.0336 & -0.0587 & 0 & 0 & 0 \\ 0.3833 & 0.4103 & 0.2747 & 0.0587 & -0.0531 & 0 & 0 \\ 0.1160 & 0.2937 & 0.3997 & 0.2937 & 0.0582 & -0.0542 & 0 \\ -0.0338 & 0.0610 & 0.2870 & 0.4126 & 0.2870 & 0.0610 & -0.0338 \\ 0 & -0.0542 & 0.0582 & 0.2937 & 0.3997 & 0.2937 & 0.1160 \\ 0 & 0 & -0.0531 & 0.0587 & 0.2747 & 0.4103 & 0.3833 \\ 0 & 0 & 0 & -0.0587 & 0.0336 & 0.2892 & 0.5345 \end{bmatrix}$$

$$B = \text{lprs}(7,3,3) = \begin{bmatrix} 0.8182 & 0.1836 & -0.0587 & -0.0587 & 0.0336 & 0.0682 & -0.1049 \\ 0.4895 & 0.4510 & 0.2741 & 0.0587 & -0.0951 & -0.0874 & 0.1818 \\ -0.2448 & 0.4283 & 0.5245 & 0.2937 & -0.0140 & -0.1486 & 0.1399 \\ -0.2797 & 0.1049 & 0.3357 & 0.4126 & 0.3357 & 0.1049 & -0.2797 \\ 0.1399 & -0.1486 & -0.0140 & 0.2937 & 0.5245 & 0.4283 & -0.2448 \\ 0.1818 & -0.0874 & -0.0951 & 0.0587 & 0.2741 & 0.4510 & 0.4895 \\ -0.1049 & 0.0682 & 0.0336 & -0.0587 & -0.0587 & 0.1836 & 0.8182 \end{bmatrix}$$

The mean-square revision error (9.8.9) was calculated assuming a local linear function of time for the input. The criterion can be generalized to higher-order polynomials. For example, for a second-order polynomial,

$$x_{L-1-i-m} = a + bm + cm^2 + v_m$$

the mean-square revision error will be:

$$\begin{aligned} E[e_{L-1-i}^2] &= E\Big[\big[\sum_{m=-i}^{M} g_m(a + bm + cm^2 + v_m) - \sum_{m=-M}^{M} h_m(a + bm + cm^2 + v_m)\big]^2\Big] \\ &= \sigma^2 \sum_{m=-i}^{M} (g_m - h_m)^2 + c^2\Big(\sum_{m=-i}^{M} m^2 g_m\Big)^2 + \text{const.} \end{aligned} \tag{9.8.18}$$

where we assumed that h_m is symmetric, has unity gain at DC, and zero second moment (i.e., it reproduces second-order polynomials). Similarly, we assumed that g_m has unity gain at DC and zero first moment (so that it reproduces first-order polynomials). Thus, the expression (9.8.18) was obtained under the constraints:

$$h_m = h_{-m}\,, \quad \sum_{m=-M}^{M} \begin{bmatrix} 1 \\ m \\ m^2 \end{bmatrix} h_m = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad \sum_{m=-i}^{M} \begin{bmatrix} 1 \\ m \end{bmatrix} g_m = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \tag{9.8.19}$$

Defining $\gamma^2 = c^2/\sigma^2$, we obtain the following optimization criterion, which incorporates the above constraints on g_m with two Lagrange multipliers λ_1, λ_2:

$$\mathcal{J} = \sum_{m=-i}^{M} (g_m - h_m)^2 + \gamma^2\Big(\sum_{m=-i}^{M} m^2 g_m\Big)^2 + \lambda_1\Big(1 - \sum_{m=-i}^{M} g_m\Big) - \lambda_2\Big(\sum_{m=-i}^{M} m g_m\Big) \tag{9.8.20}$$

The vanishing of the gradient gives:

$$g_m = h_m + \lambda_1 + \lambda_2 m - \gamma^2 G m^2\,, \quad G = \sum_{m=-i}^{M} m^2 g_m \tag{9.8.21}$$

By multiplying by m^0, m^1, m^2 and summing up over m, we obtain three equations for the three unknowns λ_1, λ_2, G,

$$\begin{aligned} 1 &= \sum_{m=-i}^{M} h_m + \Big(\sum_{m=-i}^{M} 1\Big)\lambda_1 + \Big(\sum_{m=-i}^{M} m\Big)\lambda_2 - \Big(\sum_{m=-i}^{M} m^2\Big)\gamma^2 G \\ 0 &= \sum_{m=-i}^{M} m h_m + \Big(\sum_{m=-i}^{M} m\Big)\lambda_1 + \Big(\sum_{m=-i}^{M} m^2\Big)\lambda_2 - \Big(\sum_{m=-i}^{M} m^3\Big)\gamma^2 G \\ G &= \sum_{m=-i}^{M} m^2 h_m + \Big(\sum_{m=-i}^{M} m^2\Big)\lambda_1 + \Big(\sum_{m=-i}^{M} m^3\Big)\lambda_2 - \Big(\sum_{m=-i}^{M} m^4\Big)\gamma^2 G \end{aligned} \tag{9.8.22}$$

Substituting the solutions for λ_1, λ_2, G into (9.8.21) gives the solution:

$$g_m^i = h_m + \frac{A_i}{M+i+1} + \frac{B_i}{\Sigma_i}(m - \mu_i) + \frac{\gamma^2 C_i}{\Delta_i}\big[(m-\mu_i)^2 - \nu_i^2\big]\,, \quad -i \le m \le M \tag{9.8.23}$$

for $i = 0, 1, \ldots, M-1$, with the same constants A_i, B_i, μ_i as in Eq. (9.8.13), and with $\Sigma_i, \Delta_i, \nu_i, C_i$ are defined by

$$\begin{aligned} \Sigma_i &= \frac{1}{12}(M+i)(M+i+1)(M+i+2)\,, \quad \nu_i^2 = \frac{1}{12}(M+i)(M+i+2) \\ \Delta_i &= 1 + \frac{\gamma^2}{180}(M+i-1)(M+i)(M+i+1)(M+i+2) \\ C_i &= \sum_{m=-M}^{-i-1} \big[(m-\mu_i)^2 - \nu_i^2\big] h_m \end{aligned} \tag{9.8.24}$$

9.9 Seasonal Whittaker-Henderson Decomposition

There are several other seasonal decomposition methods. The Holt-Winters exponential smoothing method [239–241], which was briefly discussed in Eq. (6.13.7), is a simple, effective, method of simultaneously tracking trend and seasonal components.

Another method is based on a seasonal generalization of the Whittaker-Henderson method [622–625] and we discuss it a more detail in this section.

Model-based methods of seasonal adjustment [626–642] are widely used and are often preferred over the X-11/X-12 methods. They are based on making ARIMA-type models for the trend and seasonal components and then estimating the components using optimum Wiener filters, or their more practical implementation as Kalman filters [643–664]. We encountered some examples in making signal models of exponential-smoothing, spline, and Whittaker-Henderson filters. We will be discussing the state-space approach in a later chapter.

The seasonal generalization of the Whittaker-Henderson method, which was originally introduced by Leser, Akaike, and Schlicht [622–624], differs from the Whittaker-Henderson/Kaiser method that we discussed earlier in that the latter determines the trend t_n using ordinary Whittaker-Henderson smoothing, and then applies a Kaiser-window comb filter to the residual $r_n = y_n - t_n$ to extract the seasonal part s_n. By contrast, in the seasonalized version, t_n and s_n are determined simultaneously from a single optimization criterion. We recall that the ordinary Whittaker-Henderson performance index for estimating the trend t_n is,

$$\mathcal{J} = \sum_{n=0}^{N-1} (y_n - t_n)^2 + \lambda \sum_{n=s}^{N-1} (\nabla^s t_n)^2 = \min \tag{9.9.1}$$

where s is the smoothing order and N, the length of y_n. The seasonalized version with period D replaces this by,

$$\mathcal{J} = \sum_{n=0}^{N-1} (y_n - t_n - s_n)^2 + \lambda \sum_{n=s}^{N-1} (\nabla^s t_n)^2 + \alpha \sum_{n=D-1}^{N-1} (s_n + s_{n-1} + \cdots + s_{n-D+1})^2 = \min \tag{9.9.2}$$

A fourth term, $\beta \sum_{n=D}^{N-1} (s_n - s_{n-D})^2$, may be added [623,624], but it is generally not necessary for the following reason. The minimization of $\mathcal{J}$ forces the sum

$$S_n = s_n + s_{n-1} + \cdots + s_{n-D+1} \tag{9.9.3}$$

to become small, ideally zero, and as a consequence the quantity $s_n - s_{n-D} = S_n - S_{n-1}$ will also be made small. Nevertheless, such a term has been implemented as an option in the function `swhdec` below. Eq. (9.9.2) can be written in a compact vectorial form as,

$$\mathcal{J} = (\mathbf{y} - \mathbf{t} - \mathbf{s})^T (\mathbf{y} - \mathbf{t} - \mathbf{s}) + \lambda\, \mathbf{t}^T (D_s^T D_s)\, \mathbf{t} + \alpha\, \mathbf{s}^T (A^T A)\, \mathbf{s} = \min \tag{9.9.4}$$

where, as discussed in general terms in Sec. 8.1, the matrices D_s, A have dimensions $(N-s)\times N$ and $(N-D+1)\times N$, respectively, and are the steady-state versions of the convolution matrices of the corresponding filters, that is,

$$\begin{aligned} &D_s(z) = (1 - z^{-1})^s, \quad \mathbf{d}_s = \text{binom}(s), \qquad D_s = \text{convmat}\big(\text{flip}(\mathbf{d}_s),\, N - s\big)^T \\ &A(z) = \sum_{k=0}^{D-1} z^{-k}, \qquad \mathbf{a} = [\underbrace{1,\, 1,\, \ldots,\, 1}_{D \text{ ones}}], \quad A = \text{convmat}\big(\text{flip}(\mathbf{a}),\, N-D+1\big)^T \end{aligned} \tag{9.9.5}$$

For example, we have for $N = 7$, $s = 2$, and $D = 4$:

$$D_s = \begin{bmatrix} 1 & -2 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & -2 & 1 \end{bmatrix}, \quad A = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{9.9.6}$$

The solution of the minimization problem (9.9.4) is obtained from the vanishing of the gradient of $\mathcal{J}$ with respect to $\mathbf{t}$ and $\mathbf{s}$, which results in the system and its solution:

$$\begin{aligned} (I + P)\mathbf{t} + \mathbf{s} &= \mathbf{y} \\ \mathbf{t} + (I + Q)\mathbf{s} &= \mathbf{y} \end{aligned} \quad \Rightarrow \quad \boxed{\begin{aligned} \mathbf{t} &= (Q + P + QP)^{-1} Q\mathbf{y} \\ \mathbf{s} &= \mathbf{y} - (I + P)\mathbf{t} \end{aligned}} \tag{9.9.7}$$

where we defined $P = \lambda (D_s^T D_s)$ and $Q = \alpha (A^T A)$. The matrices P, Q and $(Q + P + QP)$ are *banded sparse* matrices and therefore the indicated inverse[†] in (9.9.7) can be computed very efficiently with $O(N)$ operations (provided it is implemented by the backslash operator in MATLAB.)

From the above system we also have, $\mathbf{t} = (I + P)^{-1}(\mathbf{y} - \mathbf{s})$, which has the appealing interpretation that the trend is obtained by an ordinary Whittaker-Henderson smoother, i.e., the operator $(I + P)^{-1}$, applied to the seasonally-adjusted signal $(\mathbf{y} - \mathbf{s})$, which is similar to how the X-11 method obtains the final trend by applying a Henderson filter.

The function `swhdec` implements this method. It has an optional argument for the fourth β-term mentioned above:

```
[yt,ys,yi] = swhdec(y,D,s,lambda,alpha,beta);     % seasonal Whittaker-Henderson
```

The larger the parameters α, β, the closer to zero the quantity (9.9.3), and the "more periodic" the seasonal component. Thus, if one wants to extract a slowly evolving periodic component, one should choose smaller values for these parameters, relative to λ. The latter, can be estimated using the GCV criterion. The simultaneous estimation of λ, α, β can be accomplished by maximizing an appropriate likelihood function in a Bayesian formulation of this method [623,636,638].

The ℓ_1-regularized version can be obtained by replacing the ℓ_2 norms of the regularizing parts by their ℓ_1 norms, that is,

$$\mathcal{J} = \sum_{n=0}^{N-1} (y_n - t_n - s_n)^2 + \lambda \sum_{n=s}^{N-1} \big|\nabla^s t_n\big| + \alpha \sum_{n=D-1}^{N-1} \big|s_n + s_{n-1} + \cdots + s_{n-D+1}\big| = \min$$

$$\boxed{\mathcal{J} = \big\|(\mathbf{y} - \mathbf{t} - \mathbf{s})\big\|_2^2 + \lambda \big\|D_s\, \mathbf{t}\big\|_1 + \alpha \big\|A\, \mathbf{s}\big\|_1 = \min} \tag{9.9.8}$$

and can be solved easily with the CVX package.[‡]

Example 9.9.1: We revisit the unemployment data for 16–19 year old men for the 1965–79 period, which we encountered in Example 9.6.2. Fig. 9.9.1 compares the trend/seasonal decomposition obtained by the X-11 method (top graphs) and by the seasonal Whittaker-Henderson (middle graphs), as well as the corresponding L_1 version (bottom graphs). The input parameters were as follows, where λ was determined in Example 9.6.2 by the GCV criterion,

$$D = 12, \quad s = 2, \quad \lambda = 2039, \quad \alpha = 10, \quad \beta = 0$$

The MATLAB code used to generate the six graphs was as follows:

```
Y = loadfile('unemp-1619-nsa.dat'); i=find(Y==1965);        % read data
Y = Y(i:i+14,2:13)'; y = Y(:)/1000; t = taxis(y,12,65);     % extract 1965-79 range

D=12; M1=33; M2=35; N1=13; N2=13; R=3.5; type='a';          % X-11 input parameters
[yt,ys,yi] = x11dec(y,D,M1,M2,N1,N2,R,type);                % X-11 decomposition

figure; plot(t,y, t,yt); figure; plot(t,ys);                % upper graphs
```

[†] It can be shown [624] that the inverse exists for all positive values of λ, α.

[‡] `http://cvxr.com/cvx`

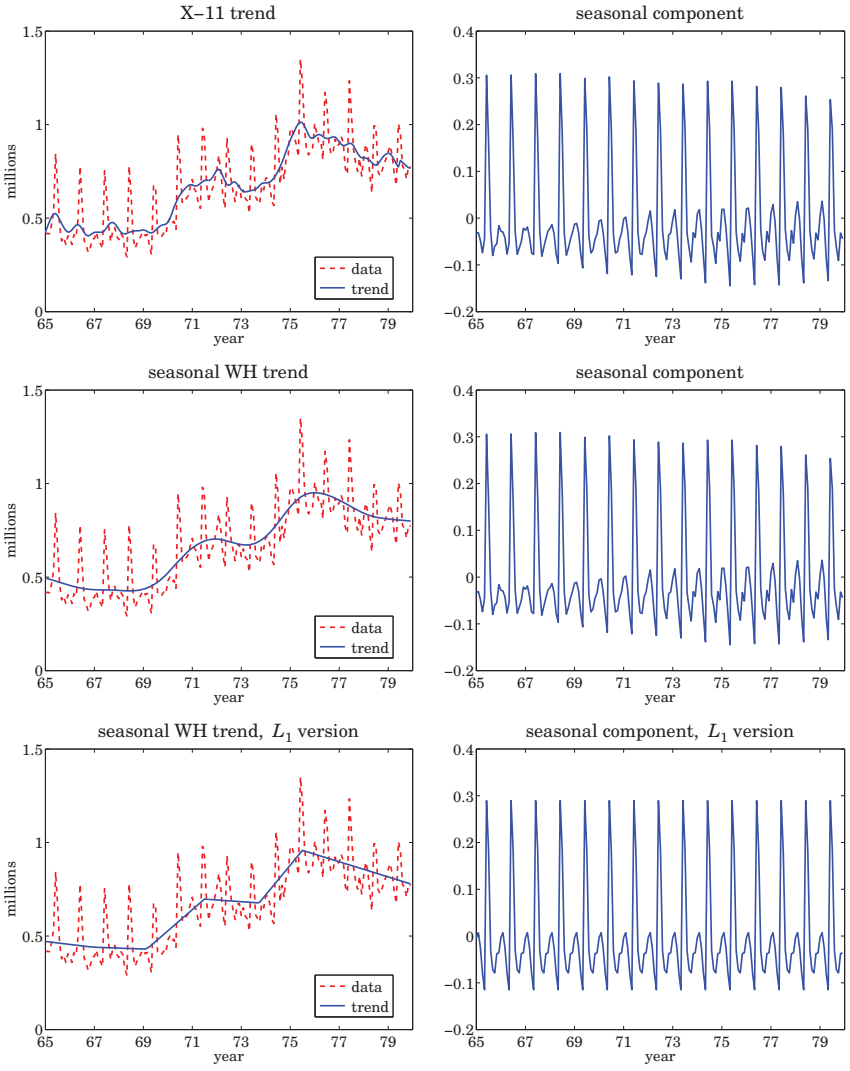


Fig. 9.9.1 X-11 and seasonal Whittaker-Henderson decomposition methods.

```
D=12; s=2; la=2039; alpha=10;                    % input parameters
[yt,ys,yi] = swhdec(y,D,s,la,alpha);             % seasonal WH decomposition

figure; plot(t,y, t,yt); figure; plot(t,ys);     % middle graphs


la=5; alpha=10;                                  % L1 version
```



```
N = length(y); s=2; Ds = diff(eye(N),s);         % construct matrices Ds and A
A = convmat(ones(1,D), N-D+1)';

cvx_quiet(true);                                 % CVX package
cvx_begin
    variable X(2*N)                              % pack trend and seasonal into X
    T = X(1:N); S = X(N+1:2*N);
    minimize( sum_square(y-T-S) + la * norm(Ds*T,1) + alpha * norm(A*S,1) );
cvx_end

yt = X(1:N); ys = X(N+1:2*N);                    % extract trend and seasonal parts

figure; plot(t,y, t,yt); figure; plot(t,ys);     % lower graphs
```

The seasonal components extracted by the methods are comparable, as are the outputs of this method and the Whittaker-Henderson/Kaiser method plotted in Fig. 9.6.2. □

In Sec. 8.2 we obtained the equivalent Whittaker-Henderson trend-extraction filter and showed that it could be thought of as the optimum unrealizable Wiener filter of a particular state-space model. The optimum filter had frequency response:

$$H(\omega)=\frac{1}{1+\lambda\,|\,D_s(\omega)\,|^2}\,,\quad \text{where}\quad D_s(\omega)=\big(1-e^{-j\omega}\big)^s \tag{9.9.9}$$

and the state-space model was defined by

$$y_n=t_n+v_n\,,\quad \nabla^s t_n=w_n \tag{9.9.10}$$

where v_n, w_n were zero-mean, mutually-uncorrelated, white-noise signals of variances σ_v^2, σ_w^2, and the smoothing parameter was identified as $\lambda=\sigma_v^2/\sigma_w^2$.

All of these results carry over to the seasonal case. First, we obtain the effective trend and seasonal filters $H_T(\omega), H_S(\omega)$ for extracting t_n, s_n. Then, we show that they are optimal in the Wiener sense. As we did in Sec. 8.2, we consider a double-sided infinitely-long signal y_n and using Parseval's identity, we may write the performance index (9.9.2) in the frequency domain, as follows:

$$\mathcal{J}=\int_{-\pi}^{\pi}\Big[\,|\,Y(\omega)-T(\omega)-S(\omega)\,|^2+\lambda\,|\,D_s(\omega)T(\omega)\,|^2+\alpha\,|\,A(\omega)S(\omega)\,|^2\Big]\frac{d\omega}{2\pi} \tag{9.9.11}$$

where $D_s(\omega)$ and $A(\omega)$ are the frequency responses of the filters in Eq. (9.9.5). From the vanishing of the gradients $\partial\mathcal{J}/\partial T^*$ and $\partial\mathcal{J}/\partial S^*$, we obtain the equations:

$$\begin{aligned} T(\omega)+\lambda\,|\,D_s(\omega)\,|^2T(\omega)+S(\omega)\;&=\;Y(\omega)\\ S(\omega)+\alpha\,|\,A(\omega)\,|^2S(\omega)+T(\omega)\;&=\;Y(\omega)\end{aligned} \tag{9.9.12}$$

which may be solved for the transfer functions $H_T(\omega)=T(\omega)/Y(\omega)$ and $H_S(\omega)=S(\omega)/Y(\omega)$, resulting in,

$$\begin{aligned} H_T(\omega)\;&=\;\frac{\alpha\,|\,A(\omega)\,|^2}{\lambda\,|\,D_s(\omega)\,|^2+\alpha\,|\,A(\omega)\,|^2+\lambda\alpha\,|\,D_s(\omega)\,|^2\,|\,A(\omega)\,|^2}\\ H_S(\omega)\;&=\;\frac{\lambda\,|\,D_s(\omega)\,|^2}{\lambda\,|\,D_s(\omega)\,|^2+\alpha\,|\,A(\omega)\,|^2+\lambda\alpha\,|\,D_s(\omega)\,|^2\,|\,A(\omega)\,|^2}\end{aligned} \tag{9.9.13}$$

with $|D_s(\omega)|^2$ and $|A(\omega)|^2$ given by,

$$\begin{aligned} |D_s(\omega)|^2 &= |1-e^{-j\omega}|^{2s} = |2\sin(\omega/2)|^{2s} \\ |A(\omega)|^2 &= |1+e^{-j\omega}+\cdots+e^{-j(D-1)\omega}|^2 = \left|\frac{\sin(\omega D/2)}{\sin(\omega/2)}\right|^2 \end{aligned} \tag{9.9.14}$$

The filters (9.9.13) generalize the Whittaker-Henderson, or Hodrick-Prescott filter (9.9.9) to the seasonal case. The filters may be identified as the optimum Wiener filters for the following signal model:

$$y_n = t_n + s_n + v_n\,, \quad \nabla^s t_n = w_n\,, \quad s_n + s_{n-1} + \cdots + s_{n-D+1} = u_n \tag{9.9.15}$$

where v_n, w_n, u_n are mutually-uncorrelated, zero-mean, white noises. The model can be written symbolically in operator form:

$$y_n = t_n + s_n + v_n\,, \quad D_s(z)t_n = w_n\,, \quad A(z)s_n = u_n \tag{9.9.16}$$

The signals t_n, s_n are not stationary, but nevertheless the optimum Wiener filters can be derived as though the signals were stationary [643–649]. Alternatively, multiplication by $D_s(z)A(z)$ acts as a stationarity-inducing transformation, resulting in the stationary signal model,

$$\begin{aligned} \bar{y}_n &= D_s(z)A(z)y_n = \bar{t}_n + \bar{s}_n + \bar{v}_n = A(z)w_n + D_s(z)u_n + D_s(z)A(z)v_n \\ \bar{t}_n &= D_s(z)A(z)t_n = A(z)w_n \\ \bar{s}_n &= D_s(z)A(z)s_n = D_s(z)u_n \\ \bar{v}_n &= D_s(z)A(z)v_n \end{aligned} \tag{9.9.17}$$

with spectral densities:

$$\begin{aligned} S_{\bar{t}\bar{y}}(\omega) &= S_{\bar{t}\bar{t}}(\omega) = \sigma_w^2|A(\omega)|^2 \\ S_{\bar{s}\bar{y}}(\omega) &= S_{\bar{s}\bar{s}}(\omega) = \sigma_u^2|D_s(\omega)|^2 \\ S_{\bar{y}\bar{y}}(\omega) &= \sigma_u^2|D_s(\omega)|^2 + \sigma_w^2|A(\omega)|^2 + \sigma_v^2|D_s(\omega)A(\omega)|^2 \end{aligned}$$

It follows from [643–649] that the optimum Wiener filters for estimating t_n, s_n will be:

$$\begin{aligned} H_T(\omega) &= \frac{S_{\bar{t}\bar{y}}(\omega)}{S_{\bar{y}\bar{y}}(\omega)} = \frac{\sigma_w^2|A(\omega)|^2}{\sigma_u^2|D_s(\omega)|^2 + \sigma_w^2|A(\omega)|^2 + \sigma_v^2|D_s(\omega)A(\omega)|^2} \\ H_S(\omega) &= \frac{S_{\bar{s}\bar{y}}(\omega)}{S_{\bar{y}\bar{y}}(\omega)} = \frac{\sigma_u^2|D_s(\omega)|^2}{\sigma_u^2|D_s(\omega)|^2 + \sigma_w^2|A(\omega)|^2 + \sigma_v^2|D_s(\omega)A(\omega)|^2} \end{aligned} \tag{9.9.18}$$

It is evident that these are identical to (9.9.13) with the identifications $\lambda = \sigma_v^2/\sigma_w^2$ and $\alpha = \sigma_v^2/\sigma_u^2$. For a finite, length-$N$, signal y_n, the model (9.9.15) has been used to derive Kalman smoothing algorithms for estimating t_n, s_n with $O(N)$ operations, and for efficiently evaluating the model's likelihood function [636,638]. We note, however, that the matrix solutions (9.9.7) are equally efficient.

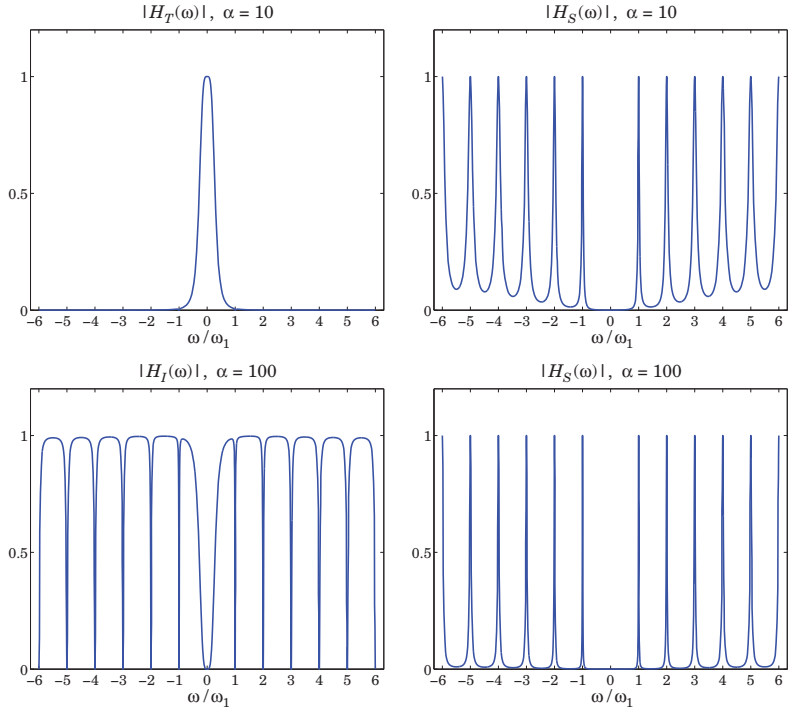


Fig. 9.9.2 Frequency responses of seasonal Whittaker-Henderson filters.

Example 9.9.2: Fig. 9.9.2 plots the frequency responses $H_T(\omega)$ and $H_S(\omega)$ of Eq. (9.9.13). For the upper graphs, the parameter values were the same as those of Example 9.9.1, that is, $D = 12$, $s = 2$, $\lambda = 2039$, $\alpha = 10$. We note that the responses have the expected shapes.

In the lower graphs, we increased the parameter α to 100 in order to sharpen the comb peaks. The lower-left graph depicts the filter $H_I(\omega) = 1 - H_T(\omega) - H_S(\omega)$ for extracting the irregular component, and the right graph depicts $H_S(\omega)$. The trend filter is not shown since it is virtually identical to that of the upper-left graph. The MATLAB code used to generate the upper graphs was as follows:

```
k = linspace(-6,6,2401); w = 2*pi*k/12;          % frequencies −π ≤ ω ≤ π

D = 12; s = 2; la = 2039; alpha = 10;

a = ones(D,1); A = freqz(a,1,w);                 % calculate A(ω)

P = la * abs(1 - exp(-j*w)).^(2*s);              % evaluate P(ω) = λ |Ds(ω)|^2
Q = alpha * abs(A).^2;                           % evaluate Q(ω) = α |A(ω)|^2
R = Q + P + Q.*P;

HT = Q./R; HS = P./R; HI = 1-HS-HT;
```

```
figure; plot(k,HT);  figure; plot(k,HS);          % upper graphs
```

9.10 Problems

9.1 First prove Eq. (9.1.2) for all n. Then, using the DFT/IDFT pair in Eq. (9.1.1), show that a more general form of (9.1.2) is,

$$\sum_{m=0}^{D-1} s_{n-m} e^{j\omega_k m} = e^{j\omega_k n} S_k\,, \quad k = 0, 1, \ldots, D-1, \quad -\infty < n < \infty$$

9.2 Consider the analog signal $s(t) = \cos(2\pi f_1 t)$ and its sampled version $s_n = \cos(2\pi f_1 nT)$, where T is the sampling interval related to the sampling rate by $f_s = 1/T$. It is required that s_n be periodic in n with period of D samples, that is, $\cos\big(2\pi f_1 (n+D)T\big) = \cos(2\pi f_1 nT)$, for all n. How does this requirement constrain f_s and f_1?

9.3 Show that the IIR comb and notch filters defined in Eq. (9.1.9) are complementary and power complementary in the sense that they satisfy Eqs. (9.1.7).

Working with the magnitude response $|H_{\text{comb}}(\omega)|^2$ show that the 3-dB width of the comb peaks is given by Eq. (9.1.11).

9.4 Show that the solution of the system (9.9.7) can be written in the more symmetric, but computationally less efficient, form:

$$\mathbf{t} = (Q + P + QP)^{-1} Q\mathbf{y}$$

$$\mathbf{s} = (P + Q + PQ)^{-1} P\mathbf{y}$$

10

Wavelets

Over the past two decades, wavelets have become useful signal processing tools for signal representation, compression, and denoising [665–833]. There exist several books on the subject [665–686], and several tutorial reviews [687–708]. The theory of wavelets and multiresolution analysis is by now very mature [709–761] and has been applied to a remarkably diverse range of applications, such as image compression and coding, JPEG2000 standard, FBI fingerprint compression, audio signals, numerical analysis and solution of integral equations, electromagnetics, biomedical engineering, astrophysics, turbulence, chemistry, infrared spectroscopy, power engineering, economics and finance, bioinformatics, characterization of long-memory and fractional processes, and statistics with regression and denoising applications [762–833].

In this chapter, we present a short review of wavelet concepts, such as multiresolution analysis, dilation equations, scaling and wavelet filters, filter banks, discrete wavelet transforms in matrix and convolutional forms, wavelet denoising, and undecimated wavelet transforms. Our discussion emphasizes computational aspects.

10.1 Multiresolution Analysis

Wavelet multiresolution analysis expands a time signal into components representing different scales—from a coarser to a finer resolution. Each term in the expansion captures the signal details at a particular scale level. The expansion is defined in terms of a sequence of nested closed subspaces V_j of the space $L^2(\mathbb{R})$ of square integrable functions on the real line $\mathbb{R}$:

$$\cdots \subset V_{-2} \subset V_{-1} \subset V_0 \subset V_1 \subset V_2 \subset \cdots \subset L^2(\mathbb{R}) \tag{10.1.1}$$

The space V_j approximates a signal at a scale j with a resolution of 2^{-j} time units. Roughly speaking, if T_0 is the sampling time interval in subspace V_0, then the sampling interval in V_j will be $T_j = 2^{-j} T_0$, which is coarser if $j < 0$, and finer if $j > 0$. The union of the V_j subspaces is the entire $L^2(\mathbb{R})$ space, and their intersection, the zero function:

$$\lim_{j\to\infty} V_j = \bigcup_{j=-\infty}^{\infty} V_j = L^2(\mathbb{R})\,, \qquad \lim_{j\to-\infty} V_j = \bigcap_{j=-\infty}^{\infty} V_j = \{0\} \tag{10.1.2}$$