(a) Show that these matrix differentiation formulas are valid.

(b) Derive an expression for the derivative

$$\frac{d}{d\mathbf{K}}\text{tr}(\mathbf{A}\mathbf{K}^H)$$

(c) Use these matrix differentiation formulas to derive the expression for the Kalman gain given in Eq. (7.113).

7.17. Consider a system consisting of two sensors, each making a single measurement of an unknown constant x. Each measurement is noisy and may be modeled as follows

$$y(1) = x + v(1)$$
$$y(2) = x + v(2)$$

where $v(1)$ and $v(2)$ are zero mean uncorrelated random variables with variance σ_1^2 and σ_2^2, respectively.

(a) In the absence of any other information, we seek the best linear estimate of x of the form

$$\hat{x} = k_1 y(1) + k_2 y(2)$$

Find the values for k_1 and k_2 that produce an unbiased estimate of x that minimizes the mean-square error, $E\{[x - \hat{x}]^2\}$.

(b) Repeat part (a) for the case where the measurement errors are correlated,

$$E\{y(1)y(2)\} = \rho\sigma_1\sigma_2$$

where ρ is the correlation coefficient.

(c) Repeat part (a) within the framework of Kalman filtering, treating the measurements $y(1)$ and $y(2)$ sequentially.

7.18. An autoregressive process of order 1 is described by the difference equation

$$x(n) = 0.5x(n-1) + w(n)$$

where $w(n)$ is zero-mean white noise with a variance $\sigma_w^2 = 0.64$. The observed process $y(n)$ is given by

$$y(n) = x(n) + v(n)$$

where $v(n)$ is zero-mean white noise with a variance $\sigma_v^2 = 1$.

(a) Write the Kalman filtering equations to find the minimum mean-square estimate, $\hat{x}(n|n)$, of $x(n)$ given the observations $y(i)$, $i = 1, \ldots, n$. The initial conditions are $\hat{x}(0|0) = 0$ and $E\{\epsilon^2(0|0)\} = 1$ where $\epsilon(0|0) = x(0) - \hat{x}(0|0)$.

(b) Assuming that the filter reaches a steady state solution, find the steady state Kalman gain and the limiting form of the estimation equation for $\hat{x}(n|n)$.

7.19. In many cases, the error covariance matrix $\mathbf{P}(n|n)$ will converge to a steady-state value $\mathbf{P}$ as $n \to \infty$. Assume that $\mathbf{C}$, $\mathbf{Q}_w$, and $\mathbf{Q}_v$ are the limiting values of $\mathbf{C}(n)$, $\mathbf{Q}_w(n)$, and $\mathbf{Q}_v(n)$, respectively.

(a) For $\mathbf{A}(n) = \mathbf{I}$, show that if $\mathbf{P}(n|n)$ converges to a steady state value $\mathbf{P}$, then the limiting value satisfies the *algebraic Ricatti equation*

$$\mathbf{P}\mathbf{C}^H\left(\mathbf{C}\mathbf{P}\mathbf{C}^H + \mathbf{Q}_v\right)^{-1}\mathbf{P} - \mathbf{Q}_w = \mathbf{0}$$

(b) Derive the Ricatti equation for a general state transition matrix $\mathbf{A}(n)$ that has a limiting value of $\mathbf{A}$.

7.20. In Example 7.4.1 we derived the Kalman filter for estimating an unknown constant from noisy measurements. The estimate at time n was shown to be

$$\hat{x}(n) = \hat{x}(n-1) + \frac{P(0)}{nP(0) + \sigma_v^2}\left[y(n) - \hat{x}(n-1)\right]$$

(a) Solve this difference equation and find a closed-form expression for $\hat{x}(n)$ in terms of $\hat{x}(0)$ and the measurements $y(0), y(1), \ldots, y(n)$.

(b) What does $\hat{x}(n)$ converge to as $n \to \infty$?

7.21. In this problem we will derive the following expression for the Kalman gain,

$$\mathbf{K}(n) = \mathbf{P}(n|n)\mathbf{C}^H(n)\mathbf{Q}_v^{-1}(n) \qquad \text{(P7.21-1)}$$

(a) By substituting Eq. (7.113) for the Kalman gain into Eq. (7.114), show that the error covariance matrix $\mathbf{P}(n|n)$ may be written as

$$\mathbf{P}(n|n) = \mathbf{P}(n|n-1) - \mathbf{P}(n|n-1)\mathbf{C}^H(n) \times \left[\mathbf{C}(n)\mathbf{P}(n|n-1)\mathbf{C}^H(n) + \mathbf{Q}_v(n)\right]^{-1}\mathbf{C}(n)\mathbf{P}(n|n-1)$$

(b) Using the matrix inversion lemma given in Eq. (2.28) on p. 29, show that the inverse covariance matrix may be written as

$$\mathbf{P}^{-1}(n|n) = \mathbf{P}^{-1}(n|n-1) + \mathbf{C}^H(n)\mathbf{Q}_v^{-1}(n)\mathbf{C}(n)$$

(c) By multiplying the expression for the Kalman gain given in Eq. (7.113) on the left by $\mathbf{P}(n|n)\mathbf{P}^{-1}(n|n)$, use your results in part (b) to derive the expression for $\mathbf{K}(n)$ given in Eq. (P7.21-1).

7.22. Consider the ARMA(1,1) process $y(n)$ given by[5]

$$y(n) + ay(n-1) = v(n) + bv(n-1)$$

where $v(n)$ is a zero-mean white-noise process with a variance σ_v^2.

(a) Show that the state-space representation for this process may be written as

$$\mathbf{x}(n) = \begin{bmatrix} -a & 1 \\ 0 & 0 \end{bmatrix}\mathbf{x}(n-1) + \begin{bmatrix} 1 \\ b \end{bmatrix}v(n)$$

$$y(n) = \begin{bmatrix} 1 & 0 \end{bmatrix}\mathbf{x}(n)$$

where $\mathbf{x}(n)$ is a two-dimensional state vector.

(b) Assuming that the error covariance $\mathbf{P}(n|n)$ converges to a steady state value of $\mathbf{P}$ and is a solution of the Ricatti equation given in Prob. 7.19, show that

$$\mathbf{P} = \sigma_v^2\begin{bmatrix} 1+c & b \\ b & b^2 \end{bmatrix}$$

[5]This problem is adapted from Haykin [5]

where c is a scalar that satisfies the second-order equation

$$c = (b-a)^2 + a^2 c - \frac{(b-a-ac)^2}{1+c}$$

and find the two values of c that satisfy this equation. For each of these values, find the corresponding values for $\mathbf{P}$.

(c) Show that the steady-state Kalman gain is

$$\mathbf{K} = \frac{b-a-ac}{1+c}\begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

and determine the values for $\mathbf{K}$ that correspond to the solutions for c found in part (b).

Computer Exercises

C7.1. Let $d(n)$ be an AR(1) process with an autocorrelation

$$r_d(k) = \alpha^{|k|}$$

with $0 < \alpha < 1$. Suppose that $d(n)$ is observed in the presence of zero-mean uncorrelated white noise $v(n)$ with a variance σ_v^2,

$$x(n) = d(n) + v(n)$$

In Example 7.2.1 we considered the design of a first-order FIR Wiener filter for estimating $d(n)$ from $x(n)$ and in Example 7.3.2 we looked at the design of a causal Wiener filter.

(a) Write a MATLAB program to design a pth-order FIR Wiener filter to estimate $d(n)$ and evaluate the mean-square error.

(b) With $\alpha = 0.8$, evaluate the error for filters of order $p = 1, 2, \ldots, 20$ and compare these errors to the causal Wiener filter. Explain your results.

(c) With $p = 10$, plot the mean-square error versus α for $\alpha = 0.1, 0.2, \ldots, 0.9$. Explain your results.

C7.2. In this exercise we look at the noise cancellation problem considered in Example 7.2.6. Let

$$x(n) = d(n) + g(n)$$

where $d(n)$ is the harmonic process

$$d(n) = \sin(n\omega_0 + \phi)$$

with $\omega_0 = 0.05\pi$ and ϕ is a random variable that is uniformly distributed between $-\pi$ and π. Assume that $g(n)$ is unit variance white noise. Suppose that a noise process $v_2(n)$ that is correlated with $g(n)$ is measured by a secondary sensor. The noise $v_2(n)$ is related to $g(n)$ by a filtering operation,

$$v_2(n) = 0.8v_2(n) + g(n)$$

(a) Using MATLAB, generate 500 samples of the processes $x(n)$ and $v_2(n)$.

(b) Derive the Wiener-Hopf equations that define the optimum pth-order FIR filter for estimating $g(n)$ from $v_2(n)$.

(c) Using filters of order $p = 2, 4$, and 6, design and implement the Wiener noise cancellation filters. Make plots of the estimated process $\hat{g}(n)$ and compare the average squared errors for each filter.

(d) In some situations, the desired signal may leak into the secondary sensor. In this case, the performance of the Wiener filter may be severely compromised. To see what effect this has, suppose the input to the Wiener filter is

$$v_0(n) = v_2(n) + \alpha d(n)$$

where $v_2(n)$ is the filtered noise defined above. Evaluate the performance of the Wiener noise canceller for several different values of α for filter orders of $p = 2$, 4, and 6. Comment on your observations.

C7.3. The state estimation equation in the discrete Kalman filter is

$$\hat{\mathbf{x}}(n|n) = \mathbf{A}(n-1)\hat{\mathbf{x}}(n-1|n-1) + \mathbf{K}(n)\Big[\mathbf{y}(n) - \mathbf{C}(n)\mathbf{A}(n-1)\hat{\mathbf{x}}(n-1|n-1)\Big]$$

Thus, given the state transition matrix $\mathbf{A}(n)$ and the observation matrix $\mathbf{C}(n)$, all that is required is the Kalman gain $\mathbf{K}(n)$. Since the Kalman gain does not depend upon the state $\mathbf{x}(n)$ or the observations $\mathbf{y}(n)$, the Kalman gain may be computed off-line prior to filtering.

(a) Write a MATLAB program `gain.m` to compute the Kalman gain $\mathbf{K}(n)$ for a stationary process with

$$\begin{aligned}\mathbf{x}(n) &= \mathbf{A}\mathbf{x}(n-1) + \mathbf{w}(n)\\ \mathbf{y}(n) &= \mathbf{C}\mathbf{x}(n) + \mathbf{v}(n)\end{aligned}$$

(b) Suppose that $x(n)$ is a third-order autoregressive process

$$x(n) = -0.1x(n-1) - 0.09x(n-2) + 0.648x(n-3) + w(n)$$

where $w(n)$ is unit variance white noise, and that the observations are

$$y(n) = x(n) + v(n)$$

where $v(n)$ is white noise with a variance $\sigma_v^2 = 0.64$. What initialization should you use for $P(0|0)$? Using this initialization, find the Kalman gain $\mathbf{K}(n)$ for $n = 0$ to $n = 10$.

(c) What is the steady-state value for the Kalman gain? How is it affected by the initialization $P(0|0)$?

(d) Generate the processes $x(n)$ and $y(n)$ in part (b) and use your Kalman filter to estimate $x(n)$ from $y(n)$. Plot your estimate and compare it to $x(n)$.

(e) Repeat parts (b) and (d) for the process

$$x(n) = -0.95x(n-1) - 0.9025x(n-2) + w(n) - w(n-1)$$

where $w(n)$ is unit variance white noise, and the observations are

$$y(n) = x(n) + v(n)$$

where $v(n)$ is white noise with a variance $\sigma_v^2 = 0.8$.

SPECTRUM ESTIMATION

8

8.1 INTRODUCTION

In this chapter we consider the problem of estimating the power spectral density of a wide-sense stationary random process. As discussed in Chapter 3, the power spectrum is the Fourier transform of the autocorrelation sequence. Therefore, estimating the power spectrum is equivalent to estimating the autocorrelation. For an autocorrelation ergodic process, recall that

$$\lim_{N\to\infty}\left\{\frac{1}{2N+1}\sum_{n=-N}^{N}x(n+k)x^*(n)\right\}=r_x(k) \tag{8.1}$$

Thus, if $x(n)$ is known for all n, estimating the power spectrum is straightforward, in theory, since all that must be done is to determine the autocorrelation sequence $r_x(k)$ using Eq. (8.1), and then compute its Fourier transform. However, there are two difficulties with this approach that make spectrum estimation both an interesting and a challenging problem. First, the amount of data that one has to work with is never unlimited and, in many cases, it may be very small. Such a limitation may be an inherent characteristic of the data collection process as is the case, for example, in the analysis of seismic data from an earthquake in which the signal persists for only a short period of time. It is also possible, however, that a limited data set is imposed by the requirement that the spectral characteristics of the process remain constant over the duration of the data record. In speech, for example, a stationarity requirement will restrict the length of time over which the signal may be assumed to be approximately stationary to a few milliseconds or less. The second difficulty is that the data is often corrupted by noise or contaminated with an interfering signal. Thus, spectrum estimation is a problem that involves estimating $P_x(e^{j\omega})$ from a finite number of noisy measurements of $x(n)$. In some applications, however, estimating the power spectrum may be facilitated by having prior knowledge about how the process is generated. It may be known, for example, that $x(n)$ is an autoregressive process or that it consists of one or more sinusoids in noise. This type of information may then allow one to parametrically estimate the power spectrum or, perhaps, to extrapolate the data or its autocorrelation in order to improve the performance of a spectrum estimation algorithm.

Spectrum estimation is a problem that is important in a variety of different fields and applications. It was shown in Chapter 7, for example, that the frequency response of a

noncausal Wiener smoothing filter is

$$H(e^{j\omega}) = \frac{P_d(e^{j\omega})}{P_d(e^{j\omega}) + P_v(e^{j\omega})}$$

where $P_d(e^{j\omega})$ is the power spectrum of $d(n)$, the desired output of the Wiener filter, and $P_v(e^{j\omega})$ is the power spectrum of the noise, $v(n)$. Therefore, before a Wiener smoothing filter can be designed and implemented, the power spectrum of both $d(n)$ and $v(n)$ must be determined. Since these power spectral densities are not generally known a priori, one is faced with the problem of estimating them from measurements. Another application in which spectrum estimation plays an important role is signal detection and tracking. Suppose, for example, that a sonar array is placed on the ocean floor to listen for the narrow-band acoustic signals that are generated by the rotating machinery or propellers of a ship. Once a narrow-band signal is detected, the problem of interest is to estimate its center frequency in order to determine the ships direction or velocity. Since these narrow-band signals are typically recorded in a very noisy environment, signal detection and frequency estimation are nontrivial problems that require robust, high-resolution spectrum estimation techniques. Other applications of spectrum estimation include harmonic analysis and prediction, time series extrapolation and interpolation, spectral smoothing, bandwidth compression, beamforming and direction finding [25,34].

The approaches for spectrum estimation may be generally categorized into one of two classes. The first includes the *classical* or *nonparametric* methods that begin by estimating the autocorrelation sequence $r_x(k)$ from a given set of data. The power spectrum is then estimated by Fourier transforming the estimated autocorrelation sequence. The second class includes the nonclassical or parametric approaches, which are based on using a model for the process in order to estimate the power spectrum. For example, if it is known that $x(n)$ is a pth-order autoregressive process, then measured values of $x(n)$ may be used to estimate the parameters of the all-pole model, $a_p(k)$, and these estimated model parameters, $\hat{a}_p(k)$, may then, in turn, be used to estimate the power spectrum as follows:

$$\hat{P}_x(e^{j\omega}) = \frac{1}{\left|\sum_{k=0}^{p} \hat{a}_p(k)e^{-jk\omega}\right|^2}$$

We begin this chapter with the classical or nonparametric spectrum estimation techniques. These methods, which are described in Section 8.2, include the periodogram, the modified periodogram, Bartlett's method, Welch's method, and the Blackman-Tukey method. The minimum variance method is then considered in Section 8.3. This technique involves the design of a narrow-band filter bank to generate a set of narrow-band random processes. The power spectrum at the center frequency of each bandpass filter is then estimated by measuring the power in the narrow-band process and dividing by the filter bandwidth. Next, in Section 8.4, the maximum entropy method (MEM) is presented, and it is shown that MEM is equivalent to spectrum estimation using an all-pole model. Then, in Section 8.5, we look at power spectrum estimation techniques that are based on a parametric model for the data. These models include moving average (MA), autoregressive (AR), and autoregressive moving average (ARMA). Next, in Section 8.6, we consider frequency estimation algorithms for harmonic processes that consist of a sum of sinusoids or complex exponentials in noise. These methods, sometimes referred to as noise sub-

space methods, include the Pisarenko harmonic decomposition, MUSIC, the eigenvector method, and the minimum norm algorithm. Finally, in Section 8.7, we look at principal components frequency estimation. This approach, which also assumes that the process is harmonic, forms a low-rank approximation to the autocorrelation matrix, which is then incorporated into a spectrum estimation algorithm such as the minimum variance method or MEM.

8.2 NONPARAMETRIC METHODS

In this section, we consider nonparametric techniques of spectrum estimation. These methods are based on the idea of estimating the autocorrelation sequence of a random process from a set of measured data, and then taking the Fourier transform to obtain an estimate of the power spectrum. We begin with the *periodogram*, a nonparametric method first introduced by Schuster in 1898 in his study of periodicities in sunspot numbers [47,49]. As we will see, although the periodogram is easy to compute, it is limited in its ability to produce an accurate estimate of the power spectrum, particularly for short data records. We will then examine a number of modifications to the periodogram that have been proposed to improve its statistical properties. These include the modified periodogram, Bartlett's method, Welch's method, and the Blackman-Tukey method.

8.2.1 The Periodogram

The power spectrum of a wide-sense stationary random process is the Fourier transform of the autocorrelation sequence,

$$P_x(e^{j\omega}) = \sum_{k=-\infty}^{\infty} r_x(k)e^{-jk\omega}$$

Therefore, spectrum estimation is, in some sense, an autocorrelation estimation problem. For an *autocorrelation ergodic* process and an unlimited amount of data, the autocorrelation sequence may, in theory, be determined using the time-average

$$r_x(k) = \lim_{N\to\infty} \frac{1}{2N+1} \sum_{n=-N}^{N} x(n+k)x^*(n) \tag{8.2}$$

However, if $x(n)$ is only measured over a finite interval, say $n = 0, 1, \ldots, N-1$, then the autocorrelation sequence must be estimated using, for example, Eq. (8.2) with a finite sum,

$$\hat{r}_x(k) = \frac{1}{N} \sum_{n=0}^{N-1} x(n+k)x^*(n) \tag{8.3}$$

In order to ensure that the values of $x(n)$ that fall outside the interval $[0, N-1]$ are excluded from the sum, Eq. (8.3) will be rewritten as follows:

$$\hat{r}_x(k) = \frac{1}{N} \sum_{n=0}^{N-1-k} x(n+k)x^*(n) \quad ; \quad k = 0, 1, \ldots, N-1 \tag{8.4}$$

with the values of $\hat{r}_x(k)$ for $k < 0$ defined using conjugate symmetry, $\hat{r}_x(-k) = \hat{r}_x^*(k)$, and with $\hat{r}_x(k)$ set equal to zero for $|k| \geq N$. Taking the discrete-time Fourier transform of

$\hat{r}_x(k)$ leads to an estimate of the power spectrum known as the *periodogram*,

$$\hat{P}_{per}(e^{j\omega}) = \sum_{k=-N+1}^{N-1} \hat{r}_x(k)e^{-jk\omega} \tag{8.5}$$

Although defined in terms of the estimated autocorrelation sequence $\hat{r}_x(k)$, it will be more convenient to express the periodogram directly in terms of the process $x(n)$. This may be done as follows. Let $x_N(n)$ be the finite length signal of length N that is equal to $x(n)$ over the interval $[0, N-1]$, and is zero otherwise,

$$x_N(n) = \begin{cases} x(n) & ; \quad 0 \le n < N \\ 0 & ; \quad \text{otherwise} \end{cases} \tag{8.6}$$

Thus, $x_N(n)$ is the product of $x(n)$ with a rectangular window $w_R(n)$,

$$x_N(n) = w_R(n)x(n) \tag{8.7}$$

In terms of $x_N(n)$, the estimated autocorrelation sequence may be written as follows:

$$\hat{r}_x(k) = \frac{1}{N}\sum_{n=-\infty}^{\infty} x_N(n+k)x_N^*(n) = \frac{1}{N}x_N(k) * x_N^*(-k) \tag{8.8}$$

Taking the Fourier transform and using the convolution theorem, the periodogram becomes

$$\boxed{\hat{P}_{per}(e^{j\omega}) = \frac{1}{N}X_N(e^{j\omega})X_N^*(e^{j\omega}) = \frac{1}{N}\left|X_N(e^{j\omega})\right|^2} \tag{8.9}$$

where $X_N(e^{j\omega})$ is the discrete-time Fourier transform of the N-point data sequence $x_N(n)$,

$$X_N(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x_N(n)e^{-jn\omega} = \sum_{n=0}^{N-1} x(n)e^{-jn\omega} \tag{8.10}$$

Thus, the periodogram is proportional to the squared magnitude of the DTFT of $x_N(n)$, and may be easily computed using a DFT as follows:

$$x_N(n) \xrightarrow{\text{DFT}} X_N(k) \longrightarrow \frac{1}{N}|X_N(k)|^2 = \hat{P}_{per}(e^{j2\pi k/N})$$

A MATLAB program for the periodogram is given in Fig. 8.1.

The Periodogram

```
function Px = periodogram(x,n1,n2)
%
   x   = x(:);
   if nargin == 1
      n1 = 1;  n2 = length(x);  end;
   Px = abs(fft(x(n1:n2),1024)).^2/(n2-n1+1);
   Px(1)=Px(2);
   end;
```

Figure 8.1 *A* MATLAB *program for computing the periodogram of* $x(n)$.

Example 8.2.1 *Periodogram of White Noise*

If $x(n)$ is white noise with a variance of σ_x^2, then $r_x(k) = \sigma_x^2\delta(k)$ and the power spectrum is a constant,

$$P_x(e^{j\omega}) = \sigma_x^2$$

Shown in Fig. 8.2*a* is a sample realization of unit variance white noise of length $N = 32$. The autocorrelation sequence that is estimated using Eq. (8.3) is shown in Fig. 8.2*b*. Note that

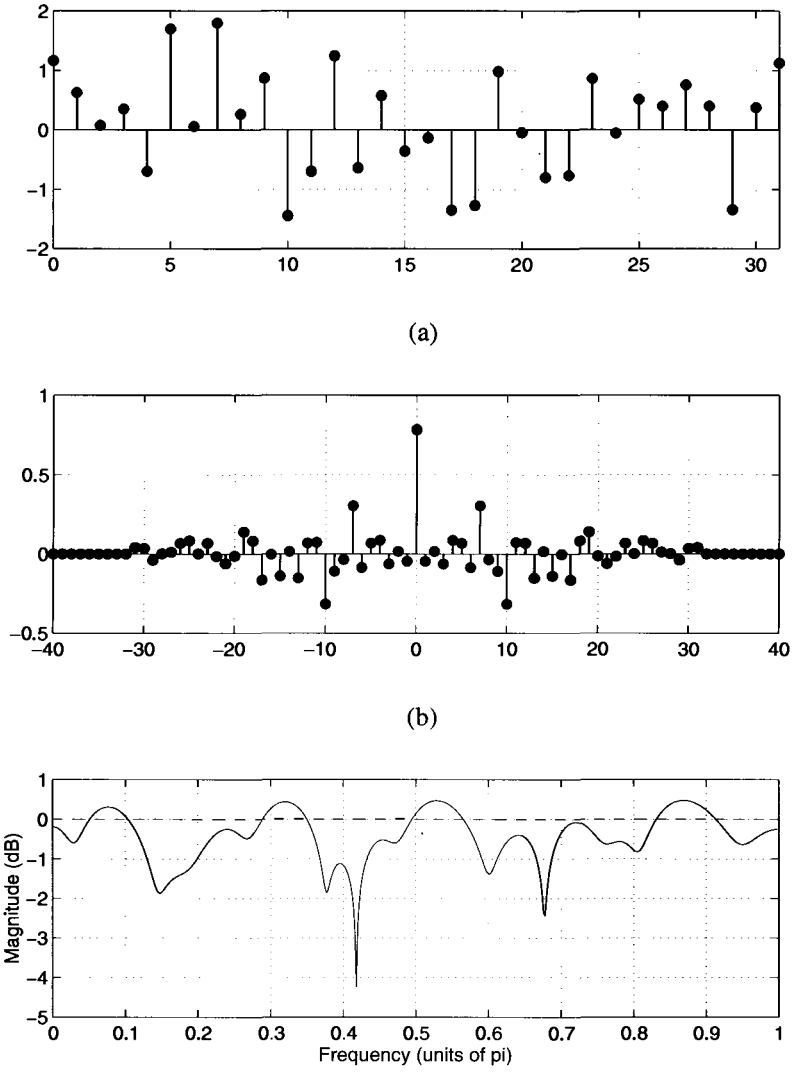


Figure 8.2 *(a) A sample realization of unit variance white noise of length $N = 32$. (b) The estimated autocorrelation sequence. (c) The periodogram along with the true power spectrum, $P_x(e^{j\omega}) = 1$, which is indicated by the dotted line.*

although $\hat{r}_x(k)$ is zero for $|k| \geq 32$, it is nonzero for all other values of k. The periodogram, which is the Fourier transform of $\hat{r}_x(k)$, is shown in Fig. 8.2*c* along with the true power spectrum. Note that although the periodogram is approximately equal to $P_x(e^{j\omega})$, on the average, we see that there is a considerable amount of variation in $\hat{P}_{per}(e^{j\omega})$ as ω varies. As we will see, such variations are not uncommon with the periodogram.

As we now show, the periodogram has an interesting interpretation in terms of filter banks. Let $h_i(n)$ be an FIR filter of length N that is defined as follows:

$$h_i(n) = \frac{1}{N} e^{jn\omega_i} w_R(n) = \begin{cases} \frac{1}{N} e^{jn\omega_i} & ; \quad 0 \leq n < N \\ 0 & ; \quad \text{otherwise} \end{cases} \tag{8.11}$$

The frequency response of this filter is

$$H_i(e^{j\omega}) = \sum_{n=0}^{N-1} h_i(n) e^{-jn\omega} = e^{-j(\omega-\omega_i)(N-1)/2} \frac{\sin[N(\omega - \omega_i)/2]}{N \sin[(\omega - \omega_i)/2]} \tag{8.12}$$

which, as illustrated in Fig. 8.3, is a bandpass filter with a center frequency ω_i, and a bandwidth that is approximately equal to $\Delta\omega = 2\pi/N$. If a WSS random process $x(n)$ is filtered with $h_i(n)$, then the output process is

$$y_i(n) = x(n) * h_i(n) = \sum_{k=n-N+1}^{n} x(k) h_i(n-k) = \frac{1}{N} \sum_{k=n-N+1}^{n} x(k) e^{j(n-k)\omega_i} \tag{8.13}$$

Since $|H_i(e^{j\omega})|_{\omega=\omega_i} = 1$, then the power spectrum of $x(n)$ and $y(n)$ are equal at frequency ω_i,

$$P_x(e^{j\omega_i}) = P_y(e^{j\omega_i})$$

Furthermore, if the bandwidth of the filter is small enough so that the power spectrum of $x(n)$ may be assumed to be approximately constant over the passband of the filter, then the

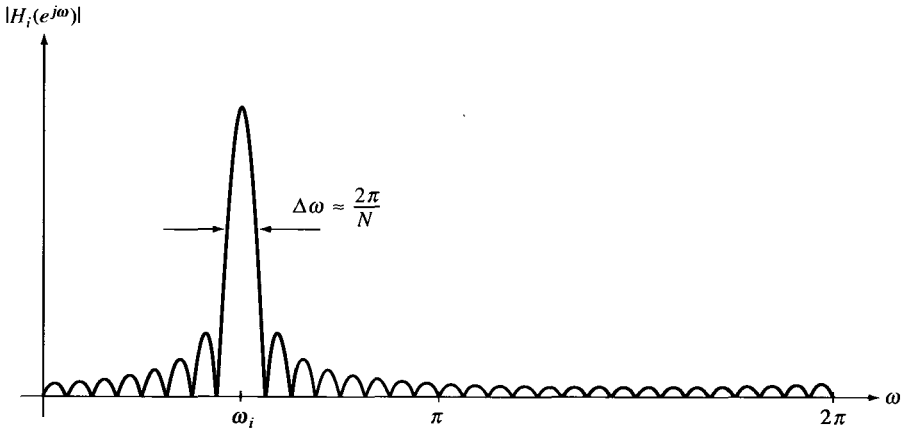


Figure 8.3 *The magnitude of the frequency response of the bandpass filter used in the filter bank interpretation of the periodogram.*

power in $y_i(n)$ will be approximately[1]

$$E\{|y_i(n)|^2\} = \frac{1}{2\pi}\int_{-\pi}^{\pi} P_x(e^{j\omega})|H_i(e^{j\omega})|^2 d\omega \approx \frac{\Delta\omega}{2\pi} P_x(e^{j\omega_i}) = \frac{1}{N} P_x(e^{j\omega_i})$$

and, therefore,

$$P_x(e^{j\omega_i}) \approx NE\{|y_i(n)|^2\} \tag{8.14}$$

Thus, if we are able to estimate the power in $y_i(n)$, then the power spectrum at frequency ω_i may be estimated as follows:

$$\hat{P}_x(e^{j\omega_i}) = N\hat{E}\{|y_i(n)|^2\} \tag{8.15}$$

One simple yet very crude way to estimate the power is to use a one-point sample average,

$$\hat{E}\{|y_i(n)|^2\} = |y_i(N-1)|^2$$

From Eq. (8.13) we see that this is equivalent to

$$|y_i(N-1)|^2 = \frac{1}{N^2}\left|\sum_{k=0}^{N-1} x(k)e^{-jk\omega_i}\right|^2 \tag{8.16}$$

Therefore,

$$\hat{P}_x(e^{j\omega_i}) = N|y_i(N-1)|^2 = \frac{1}{N}\left|\sum_{k=0}^{N-1} x(k)e^{-jk\omega_i}\right|^2 \tag{8.17}$$

which is equivalent to the periodogram. Thus, the periodogram may be viewed as the estimate of the power spectrum that is formed using a filter bank of bandpass filters as illustrated in Fig. 8.4, with $\hat{P}_{per}(e^{j\omega_i})$ being derived from a one-point sample average of the power in the filtered process $y_i(n)$. Of course, the periodogram has such a filter bank "built into it" so that it is not necessary to implement the filter bank. However, in Section 8.3

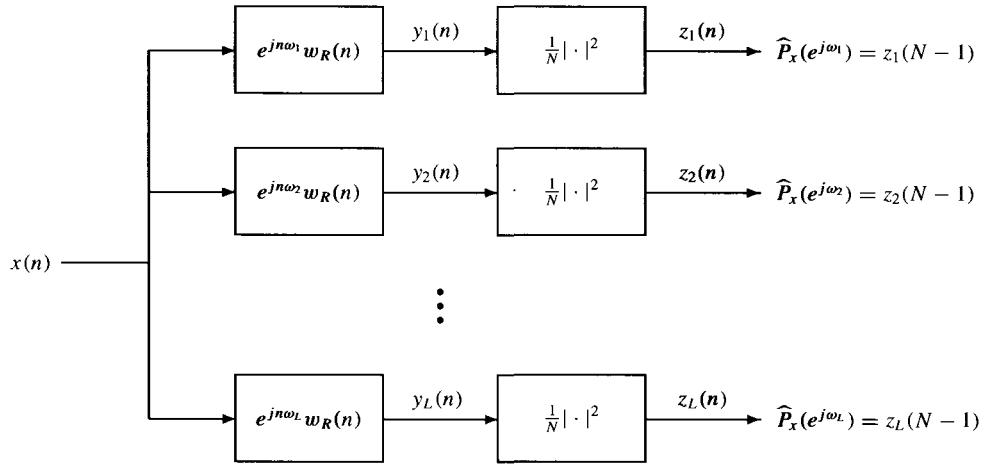


Figure 8.4 *A filter bank interpretation of the periodogram.*

[1] See the discussion leading up to Eq. (3.96) on p. 103.

we will consider a generalization of this filter bank idea that allows the filters to be *data dependent* and to have a frequency response that varies with the center frequency ω_i.

8.2.2 Performance of the Periodogram

In the previous section, it was shown that the periodogram is proportional to the squared magnitude of the DTFT of the finite length sequence $x_N(n)$. Therefore, from a computational point of view, the periodogram is simple to evaluate. In this section, we look at the performance of the periodogram. Ideally, as the length of the data record increases, the periodogram should converge to the power spectrum of the process, $P_x(e^{j\omega})$. However, we must be careful when discussing the convergence of the periodogram to $P_x(e^{j\omega})$. Specifically, since $\hat{P}_{per}(e^{j\omega})$ is a function of the random variables $x(0), \ldots, x(N)$, it is necessary to consider convergence in a statistical sense. Therefore, in this section, we will look at mean-square convergence of the periodogram [33,40], i.e., we will be interested in whether or not

$$\lim_{N\to\infty} E\left\{\left[\hat{P}_{per}(e^{j\omega}) - P_x(e^{j\omega})\right]^2\right\} = 0$$

In order for the periodogram to be mean-square convergent, it is necessary that it be asymptotically unbiased

$$\lim_{N\to\infty} E\left\{\hat{P}_{per}(e^{j\omega})\right\} = P_x(e^{j\omega}) \tag{8.18}$$

and have a variance that goes to zero as the data record length N goes to infinity,

$$\lim_{N\to\infty} \text{Var}\left\{\hat{P}_{per}(e^{j\omega})\right\} = 0 \tag{8.19}$$

In other words, $\hat{P}_{per}(e^{j\omega})$ must be a *consistent* estimate of the power spectrum (see Section 3.2.8). First, we consider the bias of the periodogram.

Periodogram Bias. To compute the bias of the periodogram, we begin by finding the expected value of $\hat{r}_x(k)$. From Eq. (8.4) it follows that the expected value of $\hat{r}_x(k)$ for $k = 0, 1, \ldots N-1$ is

$$E\{\hat{r}_x(k)\} = \frac{1}{N}\sum_{n=0}^{N-1-k} E\{x(n+k)x^*(n)\} = \frac{1}{N}\sum_{n=0}^{N-1-k} r_x(k) = \frac{N-k}{N} r_x(k)$$

and, for $k \geq N$, the expected value is zero. Using the conjugate symmetry of $\hat{r}_x(k)$ we have

$$E\{\hat{r}_x(k)\} = w_B(k) r_x(k) \tag{8.20}$$

where

$$w_B(k) = \begin{cases} \dfrac{N-|k|}{N} & ; \quad |k| \leq N \\ 0 & ; \quad |k| > N \end{cases} \tag{8.21}$$

is a Bartlett (triangular) window.[2] Therefore, $\hat{r}_x(k)$ is a biased estimate of the autocorrelation.

[2]Since the Bartlett window is applied to the autocorrelation sequence, $w_B(k)$ is referred to as a *lag window*. This is in contrast to a *data window* that is applied to $x(n)$.

Using Eq. (8.20) it follows that the expected value of the periodogram is

$$E\left\{\hat{P}_{per}(e^{j\omega})\right\} = E\left\{\sum_{k=-N+1}^{N-1} \hat{r}_x(k)e^{-jk\omega}\right\} = \sum_{k=-N+1}^{N-1} E\left\{\hat{r}_x(k)\right\}e^{-jk\omega}$$

$$= \sum_{k=-\infty}^{\infty} r_x(k)w_B(k)e^{-jk\omega} \tag{8.22}$$

Since $E\left\{\hat{P}_{per}(e^{j\omega})\right\}$ is the Fourier transform of the product $r_x(k)w_B(k)$, using the frequency convolution theorem we have

$$\boxed{E\left\{\hat{P}_{per}(e^{j\omega})\right\} = \frac{1}{2\pi} P_x(e^{j\omega}) * W_B(e^{j\omega})} \tag{8.23}$$

where $W_B(e^{j\omega})$ is the Fourier transform of the Bartlett window, $w_B(k)$,

$$W_B(e^{j\omega}) = \frac{1}{N}\left[\frac{\sin(N\omega/2)}{\sin(\omega/2)}\right]^2 \tag{8.24}$$

Thus, *the expected value of the periodogram is the convolution of the power spectrum* $P_x(e^{j\omega})$ *with the Fourier transform of a Bartlett window* and, therefore, the periodogram is a biased estimate. However, since $W_B(e^{j\omega})$ converges to an impulse as N goes to infinity, the periodogram is *asymptotically unbiased*

$$\boxed{\lim_{N\to\infty} E\left\{\hat{P}_{per}(e^{j\omega})\right\} = P_x(e^{j\omega})} \tag{8.25}$$

To illustrate the effect of the lag window, $w_B(k)$, on the expected value of the periodogram, consider a random process consisting of a random phase sinusoid in white noise

$$x(n) = A\sin(n\omega + \phi) + v(n)$$

where ϕ is a random variable that is uniformly distributed over the interval $[-\pi, \pi]$, and $v(n)$ is white noise with a variance σ_v^2. The power spectrum of $x(n)$ is

$$P_x(e^{j\omega}) = \sigma_v^2 + \frac{1}{2}\pi A^2\left[u_0(\omega - \omega_0) + u_0(\omega + \omega_0)\right]$$

Therefore, it follows from Eq. (8.23) that the expected value of the periodogram is

$$E\left\{\hat{P}_{per}(e^{j\omega})\right\} = \frac{1}{2\pi} P_x(e^{j\omega}) * W_B(e^{j\omega})$$

$$= \sigma_v^2 + \frac{1}{4}A^2\left[W_B(e^{j(\omega-\omega_0)}) + W_B(e^{j(\omega+\omega_0)})\right] \tag{8.26}$$

The power spectrum $P_x(e^{j\omega})$ and the expected value of the periodogram are shown in Fig. 8.5 for $N = 64$. There are two effects that should be noted in this example. First, is the spectral smoothing that is produced by $W_B(e^{j\omega})$, which leads to a spreading of the power in the sinusoid over a band of frequencies that has a bandwidth of approximately $4\pi/N$. The second effect is the power leakage through the sidelobes of the window, which creates secondary spectral peaks at frequencies $\omega_k \approx \omega_0 \pm \frac{2\pi}{N}k$. As we will see in Section 8.2.3, it is possible for these sidelobes to mask low-level narrowband components.

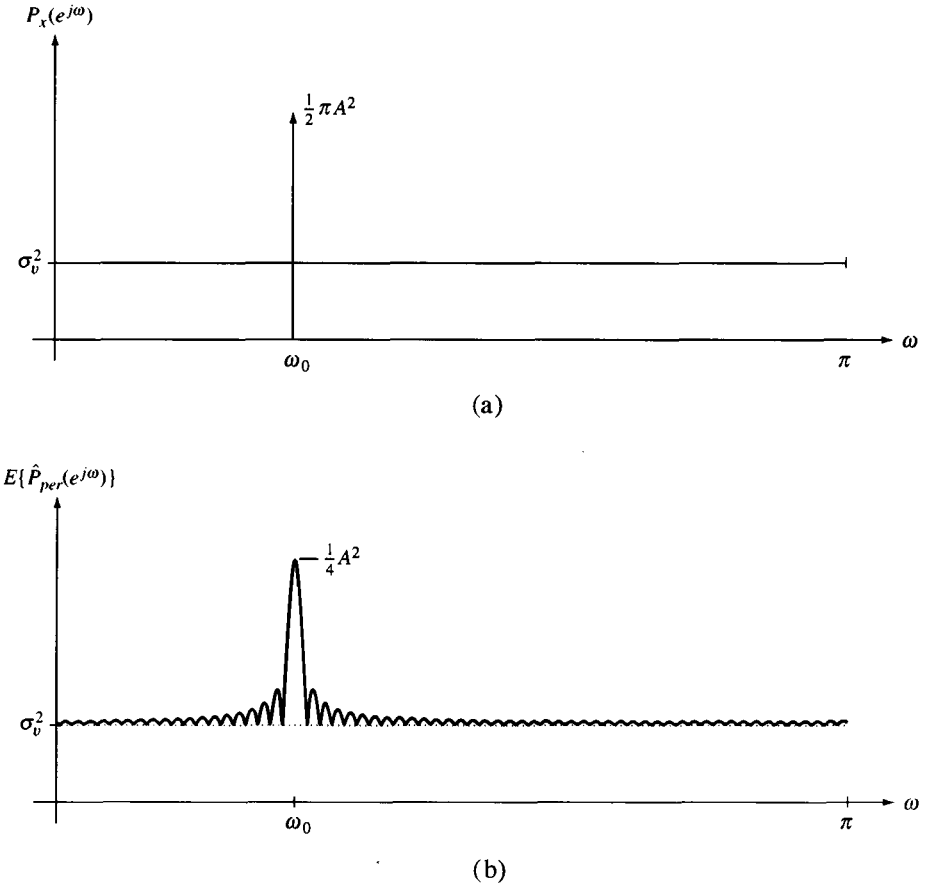


Figure 8.5 *(a) The power spectrum of a single sinusoid in white noise and (b) the expected value of the periodogram.*

Example 8.2.2 ***Periodogram of a Sinusoid in Noise***

Let $x(n)$ be a wide-sense stationary process consisting of a random phase sinusoid in unit variance white noise

$$x(n) = A\sin(n\omega_0 + \phi) + v(n)$$

With $A = 5$, $\omega_0 = 0.4\pi$, and $N = 64$, fifty different realizations of this process were generated and the periodogram of each was computed. Shown in Fig. 8.6*a* is an overlay of the 50 periodograms. Note that although each periodogram has a peak at approximately $\omega = 0.4\pi$, there is a considerable amount of variation from one periodogram to the next. In Fig. 8.6*b* the average of all 50 periodograms is shown. This average is approximately equal to the expected value given in Eq. (8.26). By increasing the number of data values to $N = 256$, we obtain the periodograms shown in Fig. 8.6*c* and *d*. Note that with the additional data, the power in the sinusoid is spread out over a much narrower band of frequencies.

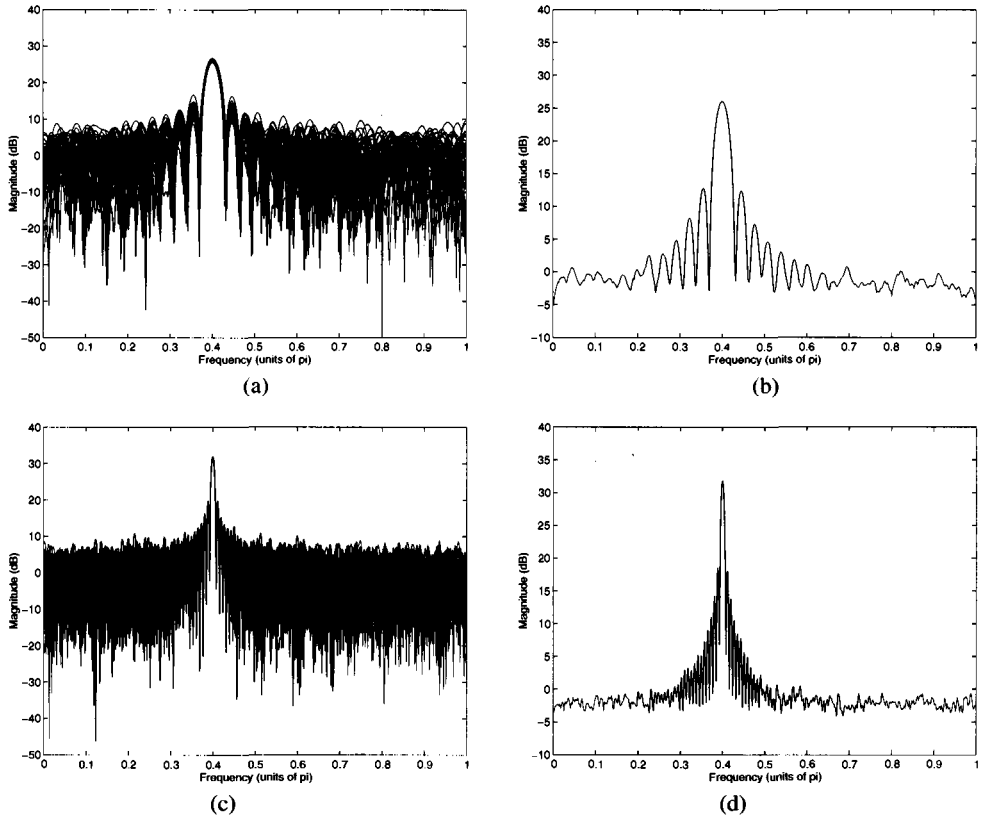


Figure 8.6 *The periodogram of a sinusoid in white noise. (a) Overlay plot of 50 periodograms using* $N = 64$ *data values and (b) the periodogram average. (c) Overlay plot of 50 periodograms using* $N = 256$ *data values and (d) the periodogram average.*

In addition to biasing the periodogram, the smoothing that is introduced by the Bartlett window also limits the ability of the periodogram to resolve closely-spaced narrowband components in $x(n)$. Consider, for example, a random process consisting of two sinusoids in white noise

$$x(n) = A_1 \sin(n\omega_1 + \phi_1) + A_2 \sin(n\omega_2 + \phi_2) + v(n)$$

where ϕ_1 and ϕ_2 are uncorrelated uniformly distributed random variables and where $v(n)$ is white noise with a variance of σ_v^2. The power spectrum of $x(n)$ is

$$P_x(e^{j\omega}) = \sigma_v^2 + \frac{1}{2}\pi A_1^2\Big[u_0(\omega - \omega_1) + u_0(\omega + \omega_1)\Big] + \frac{1}{2}\pi A_2^2\Big[u_0(\omega - \omega_2) + u_0(\omega + \omega_2)\Big]$$

and the expected value of the periodogram is

$$\begin{aligned} E\left\{\hat{P}_{per}(e^{j\omega})\right\} &= \frac{1}{2\pi} P_x(e^{j\omega}) * W_B(e^{j\omega}) \\ &= \sigma_v^2 + \frac{1}{4}A_1^2\left[W_B(e^{j(\omega-\omega_1)}) + W_B(e^{j(\omega+\omega_1)})\right] \\ &\quad + \frac{1}{4}A_2^2\left[W_B(e^{j(\omega-\omega_2)}) + W_B(e^{j(\omega+\omega_2)})\right] \end{aligned} \tag{8.27}$$

which is shown in Fig. 8.7 for $A_1 = A_2$ and $N = 64$. Since the width of the main lobe of $W_B(e^{j\omega})$ increases as the data record length decreases, for a given data record length, N, there is a limit on how closely two sinusoids or two narrowband processes may be located before they can no longer be resolved. One way to define this resolution limit is to set $\Delta\omega$ equal to the width of the main lobe of the spectral window, $W_B(e^{j\omega})$, at its "half-power" or 6 dB point. For the Bartlett window, $\Delta\omega = 0.89(2\pi/N)$, which means that the resolution of the periodogram is

$$\boxed{\text{Res}\left[\hat{P}_{per}(e^{j\omega})\right] = 0.89\frac{2\pi}{N}} \tag{8.28}$$

It is important to note, however, that Eq. (8.28) is nothing more than a *rule of thumb* that should only be used as a guideline in determining the amount of data that is necessary for a given resolution. There is nothing sacred, for example, with the proportionality constant $0.89(2\pi)$. On the other hand, what is important is the fact that the resolution is inversely proportional to the amount of data, N. Nevertheless, although the definition of $\Delta\omega$ given in Eq. (8.28) is somewhat arbitrary, one generally finds that it is difficult to resolve details in the spectrum that are much finer than this.

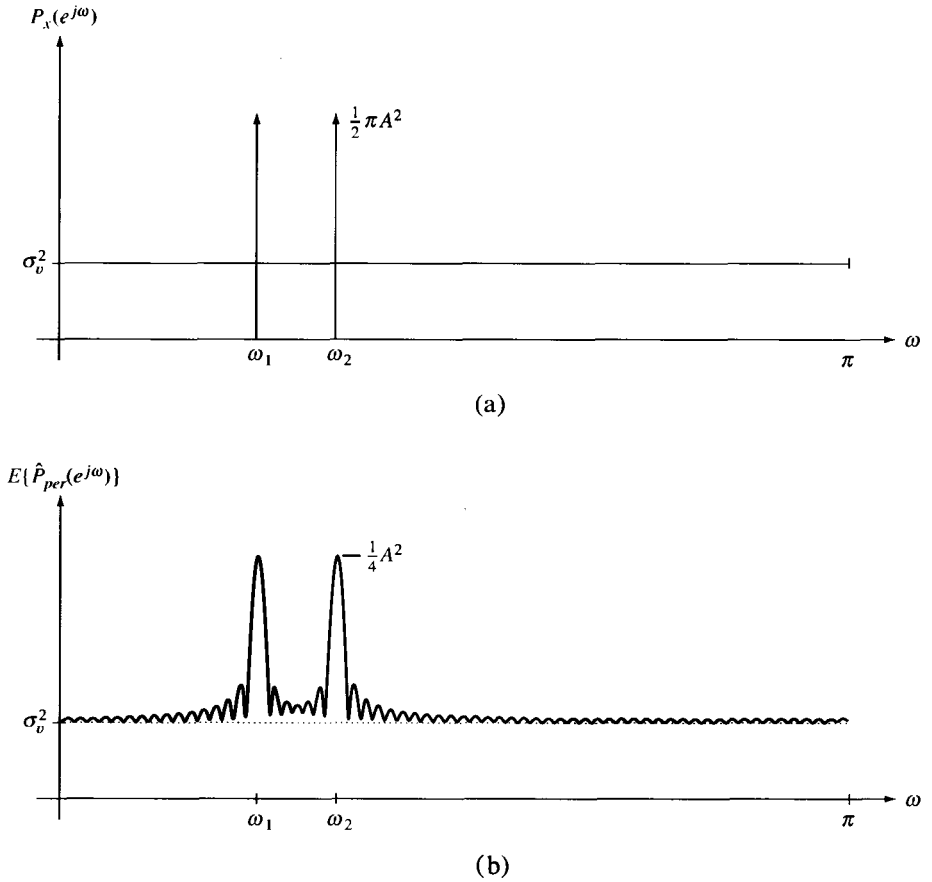


Figure 8.7 *(a) The power spectrum of two sinusoids in white noise and (b) the expected value of the periodogram.*

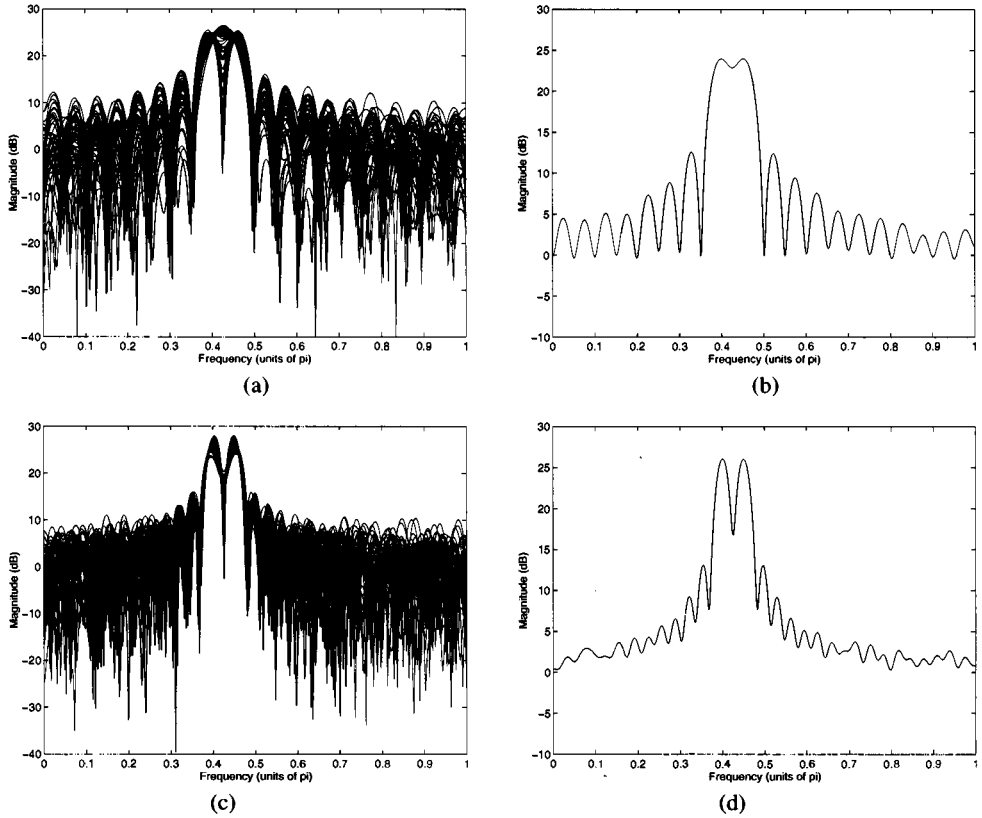


Figure 8.8 *The periodogram of two sinusoids in white noise with* $\omega_1 = 0.4\pi$ *and* $\omega_2 = 0.45\pi$. *(a) Overlay plot of 50 periodograms using* $N = 40$ *data values and (b) the ensemble average. (c) Overlay plot of 50 periodograms using* $N = 64$ *data values and (d) the ensemble average.*

Example 8.2.3 *Periodogram Resolution*

Let $x(n)$ be a random process consisting of two equal amplitude sinusoids in unit variance white noise

$$x(n) = A\sin(n\omega_1 + \phi_1) + A\sin(n\omega_2 + \phi_2) + v(n)$$

where $\omega_1 = 0.4\pi$, $\omega_2 = 0.45\pi$, and $A = 5$. Using Eq. (8.28), with $\Delta\omega = 0.05\pi$ we see that in order to resolve the two narrowband components, a data record length on the order of $N = 36$ is required. Using $N = 40$, an overlay of 50 periodograms is shown in Fig. 8.8*a*. It is clear from this figure that it is not always possible to resolve the two sinusoidal components when $N = 40$. The average of the 50 periodograms, shown in 8.8*b*, illustrates the overlap of the two Bartlett windows. However, as shown in Figures 8.8*c* and *d*, if $N = 64$ then the two sinusoids are clearly resolved.

Variance of the Periodogram. We have seen that the periodogram is an asymptotically unbiased estimate of the power spectrum. In order for it to be a consistent estimate, it is necessary that the variance go to zero as $N \to \infty$. Unfortunately, it is difficult to evaluate the variance of the periodogram for an arbitrary process $x(n)$ since the variance depends

on the fourth-order moments of the process. However, as we show next, the variance may be evaluated in the special case of white Gaussian noise.

Let $x(n)$ be a Gaussian white noise process with variance σ_x^2. Using Eq. (8.9) the periodogram may be expressed as follows:

$$\hat{P}_{per}(e^{j\omega}) = \frac{1}{N}\left|\sum_{k=0}^{N-1} x(k)e^{-jk\omega}\right|^2 = \frac{1}{N}\left\{\sum_{k=0}^{N-1} x(k)e^{-jk\omega}\right\}\left\{\sum_{l=0}^{N-1} x^*(l)e^{jl\omega}\right\}$$
$$= \frac{1}{N}\sum_{k=0}^{N-1}\sum_{l=0}^{N-1} x(k)x^*(l)e^{-j(k-l)\omega} \tag{8.29}$$

Therefore, the second-order moment of the periodogram is

$$E\left\{\hat{P}_{per}(e^{j\omega_1})\hat{P}_{per}(e^{j\omega_2})\right\}$$
$$= \frac{1}{N^2}\sum_{k=0}^{N-1}\sum_{l=0}^{N-1}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1} E\left\{x(k)x^*(l)x(m)x^*(n)\right\}e^{-j(k-l)\omega_1}e^{-j(m-n)\omega_2} \tag{8.30}$$

which depends on the fourth-order moments of $x(n)$. Since $x(n)$ is Gaussian, we may use the moment factoring theorem to simplify these moments [17,44]. For complex Gaussian random variables, the moment factoring theorem is[3]

$$E\left\{x(k)x^*(l)x(m)x^*(n)\right\} = E\left\{x(k)x^*(l)\right\}E\left\{x(m)x^*(n)\right\}$$
$$+ E\left\{x(k)x^*(n)\right\}E\left\{x(m)x^*(l)\right\} \tag{8.31}$$

Substituting Eq. (8.31) into Eq. (8.30), the second-order moment of the periodogram becomes a sum of two terms. The first term contains products of $E\left\{x(k)x^*(l)\right\}$ with $E\left\{x(m)x^*(n)\right\}$. For white noise, these terms are equal to σ_x^4 when $k = l$ and $m = n$, and they are equal to zero otherwise. Thus, the first term simplifies to

$$\frac{1}{N^2}\sum_{k=0}^{N-1}\sum_{m=0}^{N-1}\sigma_x^4 = \sigma_x^4 \tag{8.32}$$

The second term, on the other hand, contains products of $E\left\{x(k)x^*(n)\right\}$ with $E\left\{x(m)x^*(l)\right\}$. Again, for white noise, these terms are equal to σ_x^4 when $k = n$ and $l = m$, and they are equal to zero otherwise. Therefore, the second term becomes

$$\frac{1}{N^2}\sum_{k=0}^{N-1}\sum_{l=0}^{N-1}\sigma_x^4 e^{-j(k-l)\omega_1}e^{j(k-l)\omega_2} = \frac{\sigma_x^4}{N^2}\sum_{k=0}^{N-1} e^{-jk(\omega_1-\omega_2)}\sum_{l=0}^{N-1} e^{jl(\omega_1-\omega_2)}$$
$$= \frac{\sigma_x^4}{N^2}\left[\frac{1-e^{-jN(\omega_1-\omega_2)}}{1-e^{-j(\omega_1-\omega_2)}}\right]\left[\frac{1-e^{jN(\omega_1-\omega_2)}}{1-e^{j(\omega_1-\omega_2)}}\right]$$
$$= \sigma_x^4\left[\frac{\sin N(\omega_1-\omega_2)/2}{N\sin(\omega_1-\omega_2)/2}\right]^2 \tag{8.33}$$

[3]Note that this is different from the moment factoring theorem for real Gaussian random variables, which contains three terms instead of two.

Combining Eq. (8.32) and Eq. (8.33) it follows that

$$E\left\{\hat{P}_{per}(e^{j\omega_1})\hat{P}_{per}(e^{j\omega_2})\right\} = \sigma_x^4\left\{1 + \left[\frac{\sin N(\omega_1 - \omega_2)/2}{N\sin(\omega_1 - \omega_2)/2}\right]^2\right\} \tag{8.34}$$

Since

$$\begin{aligned}\text{Cov}\left\{\hat{P}_{per}(e^{j\omega_1})\hat{P}_{per}(e^{j\omega_2})\right\} &= E\left\{\hat{P}_{per}(e^{j\omega_1})\hat{P}_{per}(e^{j\omega_2})\right\} \\ &\quad - E\left\{\hat{P}_{per}(e^{j\omega_1})\right\}E\left\{\hat{P}_{per}(e^{j\omega_2})\right\}\end{aligned}$$

and $E\left\{\hat{P}_{per}(e^{j\omega})\right\} = \sigma_x^2$, then the covariance of the periodogram is

$$\text{Cov}\left\{\hat{P}_{per}(e^{j\omega_1})\hat{P}_{per}(e^{j\omega_2})\right\} = \sigma_x^4\left[\frac{\sin N(\omega_1 - \omega_2)/2}{N\sin(\omega_1 - \omega_2)/2}\right]^2 \tag{8.35}$$

Finally, setting $\omega_1 = \omega_2$ we have, for the variance,

$$\text{Var}\left\{\hat{P}_{per}(e^{j\omega})\right\} = \sigma_x^4 \tag{8.36}$$

Thus, the variance does not go to zero as $N \to \infty$, and the periodogram is *not a consistent estimate* of the power spectrum. In fact, since $P_x(e^{j\omega}) = \sigma_x^2$ then the variance of the periodogram of white Gaussian noise is proportional to the square of the power spectrum,

$$\text{Var}\left\{\hat{P}_{per}(e^{j\omega})\right\} = P_x^2(e^{j\omega}) \tag{8.37}$$

Example 8.2.4 ***Periodogram of White Noise***

Let $x(n)$ be white Gaussian noise with

$$P_x(e^{j\omega}) = 1$$

From Eq. (8.23) it follows that the expected value of the periodogram is equal to one,

$$E\{\hat{P}_{per}(e^{j\omega})\} = 1$$

and from Eq. (8.36) it follows that the variance is also equal to one,

$$\text{Var}\left\{\hat{P}_{per}(e^{j\omega})\right\} = 1$$

Thus, although the periodogram is unbiased, the variance is equal to a constant that is independent of the data record length, N. In Fig. 8.9*a*, *c*, and *e* are overlay plots of 50 periodograms of white noise that were generated using data records of length $N = 64$, 128, and 256, respectively. In Fig. 8.9*b*, *d*, and *f*, on the other hand, are the periodogram averages. What we observe is that, although the average value of the periodogram is approximately equal to $\sigma_x^2 = 1$, the variance does not decrease as the amount of data increases.

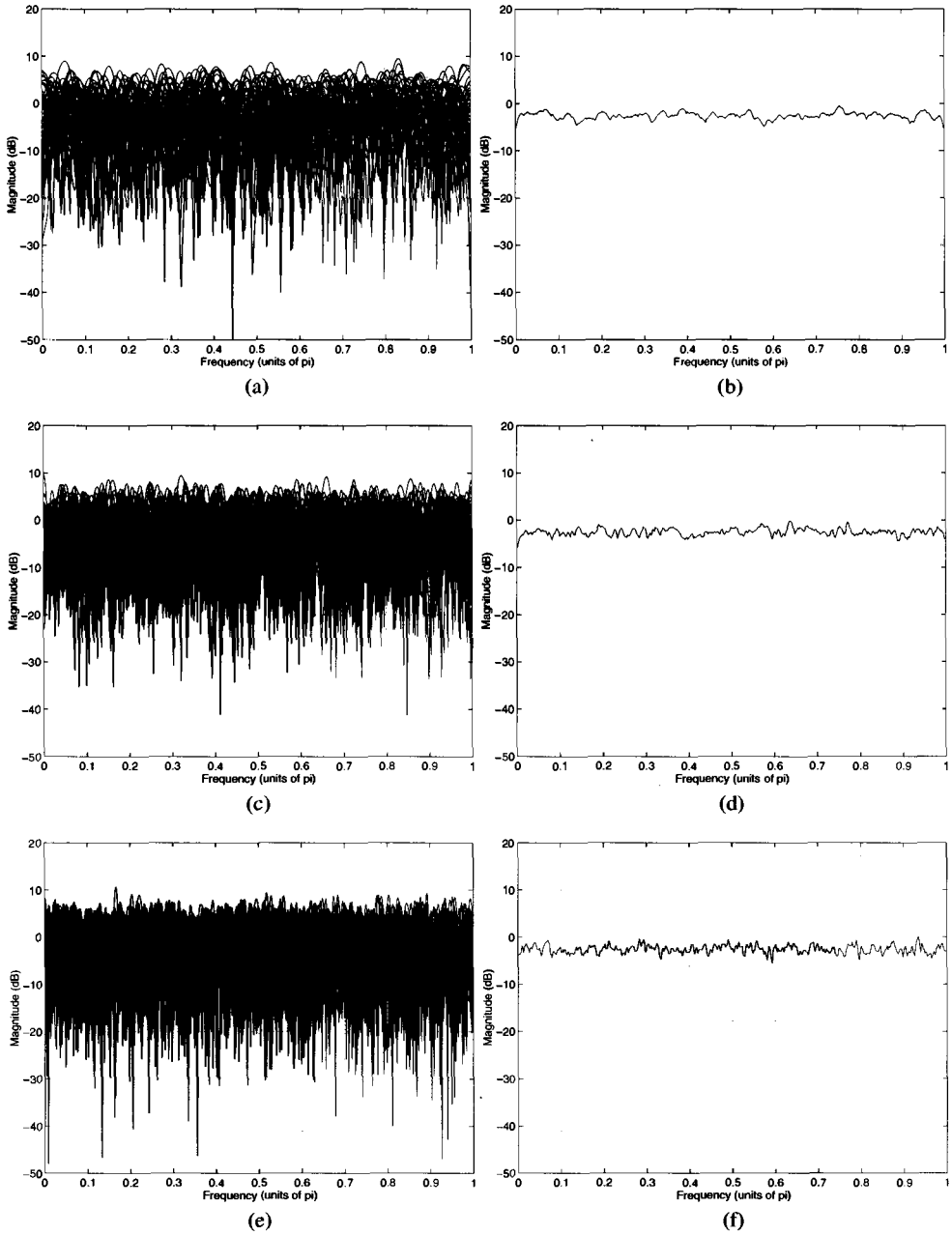


Figure 8.9 *The periodogram of unit variance white Gaussian noise. (a) Overlay plot of 50 periodograms with $N = 64$ data values and (b) the periodogram average. (c) Overlay plot of 50 periodograms with $N = 128$ data values and (d) the periodogram average. (e) Overlay plot of 50 periodograms with $N = 256$ data values and (f) the periodogram average.*

The analysis given above for the variance of the periodogram assumes that $x(n)$ is white Gaussian noise. Although the statistical analysis of a nonwhite Gaussian process is much

Table 8.1 *Properties of the Periodogram*

$$\hat{P}_{per}(e^{j\omega}) = \frac{1}{N}\left|\sum_{n=0}^{N-1} x(n)e^{-jn\omega}\right|^2$$

Bias

$$E\left\{\hat{P}_{per}(e^{j\omega})\right\} = \frac{1}{2\pi}P_x(e^{j\omega}) * W_B(e^{j\omega})$$

Resolution

$$\Delta\omega = 0.89\frac{2\pi}{N}$$

Variance

$$\text{Var}\left\{\hat{P}_{per}(e^{j\omega})\right\} \approx P_x^2(e^{j\omega})$$

8.2.3 The Modified Periodogram

In Section 8.2.1 we saw that the periodogram is proportional to the squared magnitude of the Fourier transform of the windowed signal $x_N(n) = x(n)w_R(n)$,

$$\hat{P}_{per}(e^{j\omega}) = \frac{1}{N}|X_N(e^{j\omega})|^2 = \frac{1}{N}\left|\sum_{n=-\infty}^{\infty} x(n)w_R(n)e^{-jn\omega}\right|^2 \tag{8.45}$$

Instead of applying a rectangular window to $x(n)$, Eq. (8.45) suggests the possibility of using other data windows. Would there be any benefit, for example, in replacing the rectangular window with a triangular (Bartlett) window? To answer this question, let us examine the effect of the data window on the bias of the periodogram.

Using Eq. (8.45), the expected value of the periodogram is

$$\begin{aligned} E\left\{\hat{P}_{per}(e^{j\omega})\right\} &= \frac{1}{N}E\left\{\left[\sum_{n=-\infty}^{\infty} x(n)w_R(n)e^{-jn\omega}\right]\left[\sum_{m=-\infty}^{\infty} x(m)w_R(m)e^{-jm\omega}\right]^*\right\} \\ &= \frac{1}{N}E\left\{\sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} x(n)x^*(m)w_R(m)w_R(n)e^{-j(n-m)\omega}\right\} \\ &= \frac{1}{N}\sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} r_x(n-m)w_R(m)w_R(n)e^{-j(n-m)\omega} \end{aligned} \tag{8.46}$$

With the change of variables, $k = n - m$, Eq. (8.46) becomes

$$\begin{aligned} E\left\{\hat{P}_{per}(e^{j\omega})\right\} &= \frac{1}{N}\sum_{k=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} r_x(k)w_R(n)w_R(n-k)e^{-jk\omega} \\ &= \frac{1}{N}\sum_{k=-\infty}^{\infty} r_x(k)\left[\sum_{n=-\infty}^{\infty} w_R(n)w_R(n-k)\right]e^{-jk\omega} \\ &= \frac{1}{N}\sum_{k=-\infty}^{\infty} r_x(k)w_B(k)e^{-jk\omega} \end{aligned} \tag{8.47}$$

more difficult, we may derive an approximate expression for the variance as follows. Recall that a random process $x(n)$ with power spectrum $P_x(e^{j\omega})$ may be generated by filtering unit variance white noise $v(n)$ with a linear shift-invariant filter $h(n)$ that has a frequency response $H(e^{j\omega})$ with

$$|H(e^{j\omega})|^2 = P_x(e^{j\omega}) \tag{8.38}$$

As defined in Eq. (8.7), if $x_N(n)$ and $v_N(n)$ are the sequences of length N that are formed by windowing $x(n)$ and $v(n)$, respectively, then the periodograms of these processes are

$$\hat{P}_{per}^{(x)}(e^{j\omega}) = \frac{1}{N}\left|X_N(e^{j\omega})\right|^2 \tag{8.39}$$

$$\hat{P}_{per}^{(v)}(e^{j\omega}) = \frac{1}{N}\left|V_N(e^{j\omega})\right|^2 \tag{8.40}$$

Although $x_N(n)$ is not equal to the convolution of $v_N(n)$ with $h(n)$, if N is large compared to the length of $h(n)$ so that the transient effects are small, then

$$x_N(n) \approx h(n) * v_N(n)$$

Since

$$\left|X_N(e^{j\omega})\right|^2 \approx \left|H(e^{j\omega})\right|^2 \left|V_N(e^{j\omega})\right|^2 = P_x(e^{j\omega})\left|V_N(e^{j\omega})\right|^2 \tag{8.41}$$

substituting Eqs. (8.39) and (8.40) into Eq. (8.41) we have

$$\hat{P}_{per}^{(x)}(e^{j\omega}) \approx P_x(e^{j\omega})\hat{P}_{per}^{(v)}(e^{j\omega})$$

Therefore,

$$\text{Var}\left\{\hat{P}_{per}^{(x)}(e^{j\omega})\right\} \approx P_x^2(e^{j\omega})\text{Var}\left\{\hat{P}_{per}^{(v)}(e^{j\omega})\right\}$$

and, since the variance of the periodogram of $v(n)$ is equal to one, then we have

$$\text{Var}\left\{\hat{P}_{per}^{(x)}(e^{j\omega})\right\} \approx P_x^2(e^{j\omega}) \tag{8.42}$$

Thus, assuming that N is large, the variance of the periodogram of a Gaussian random process is proportional to the square of its power spectrum.

The second-order moment and covariance of the periodogram may be similarly generalized for nonwhite Gaussian noise. For the second-order moment, Eq. (8.34) becomes

$$E\left\{\hat{P}_{per}(e^{j\omega_1})\hat{P}_{per}(e^{j\omega_2})\right\} \approx P_x(e^{j\omega_1})P_x(e^{j\omega_2})\left\{1 + \left[\frac{\sin N(\omega_1-\omega_2)/2}{N\sin(\omega_1-\omega_2)/2}\right]^2\right\} \tag{8.43}$$

and, for the covariance, Eq. (8.35) becomes

$$\text{Cov}\left\{\hat{P}_{per}(e^{j\omega_1})\hat{P}_{per}(e^{j\omega_2})\right\} \approx P_x(e^{j\omega_1})P_x(e^{j\omega_2})\left[\frac{\sin N(\omega_1-\omega_2)/2}{N\sin(\omega_1-\omega_2)/2}\right]^2 \tag{8.44}$$

Note that for large N, the term in brackets is approximately equal to zero provided $\omega_1 - \omega_2 \gg 2\pi/N$, which implies that there is little correlation between one frequency and another. The properties of the periodogram are summarized in Table 8.1.

where

$$w_B(k) = w_R(k) * w_R(-k) = \sum_{n=-\infty}^{\infty} w_R(n) w_R(n-k)$$

is a Bartlett window. Using the frequency convolution theorem, it follows that the expected value of the periodogram is

$$E\left\{\hat{P}_{per}(e^{j\omega})\right\} = \frac{1}{2\pi N} P_x(e^{j\omega}) * |W_R(e^{j\omega})|^2 \tag{8.48}$$

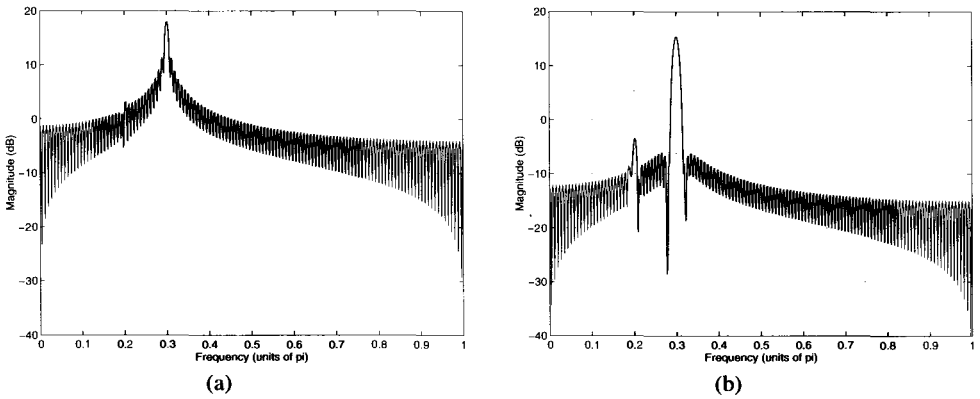
where

$$W_R(e^{j\omega}) = \frac{\sin(N\omega/2)}{\sin(\omega/2)} e^{-j(N-1)\omega/2}$$

is the Fourier transform of the rectangular data window, $w_R(n)$.[4] Therefore, the amount of smoothing in the periodogram is determined by the window that is applied to the data. Although a rectangular window has a narrow main lobe compared to other windows and, therefore, produces the least amount of spectral smoothing, it has relatively large sidelobes that may lead to masking of weak narrowband components. Consider, for example, a random process consisting of two sinusoids in white noise

$$x(n) = 0.1\sin(n\omega_1 + \phi_1) + \sin(n\omega_2 + \phi_2) + v(n)$$

With $\omega_1 = 0.2\pi$, $\omega_2 = 0.3\pi$, and $N = 128$, the expected value of the periodogram is shown in Fig. 8.10*a*. What we observe is that the sinusoid at frequency ω_1 is almost completely masked by the sidelobes of the window at frequency ω_2. However. if the rectangular window is replaced with a Hamming window, then the sinusoid at frequency ω_1 is clearly visible as illustrated in Fig. 8.10*b*. This is due to the smaller sidelobes of a Hamming window, which are down about 30 dB compared to the sidelobes of a rectangular window. On the



(a) (b)

Figure 8.10 *Spectral analysis of two sinusoids in white noise with sinusoidal frequencies of* $\omega_1 = .2\pi$ *and* $\omega_2 = .3\pi$ *and a data record length of* $N = 128$ *points. (a) The expected value of the periodogram. (b) The expected value of the modified periodogram using a Hamming data window.*

[4]Note the equivalence of Eq. (8.48) and Eq. (8.23).

other hand, this reduction in the sidelobe amplitude comes at the expense of an increase in the width of the mainlobe which, in turn, affects the resolution. The periodogram of a process that is windowed with a general window $w(n)$ is called a *modified periodogram* and is given by

$$\hat{P}_M(e^{j\omega}) = \frac{1}{NU}\left|\sum_{n=-\infty}^{\infty} x(n)w(n)e^{-jn\omega}\right|^2 \tag{8.49}$$

where N is the length of the window and

$$U = \frac{1}{N}\sum_{n=0}^{N-1}|w(n)|^2 \tag{8.50}$$

is a constant that, as we will see, is defined so that $\hat{P}_M(e^{j\omega})$ will be asymptotically unbiased. A MATLAB program for the modified periodogram is given in Fig. 8.11.

Let us now evaluate the performance of the modified periodogram. It follows from the derivation of Eq. (8.48) that the expected value of $\hat{P}_M(e^{j\omega})$ is

$$E\left\{\hat{P}_M(e^{j\omega})\right\} = \frac{1}{2\pi NU}P_x(e^{j\omega}) * |W(e^{j\omega})|^2 \tag{8.51}$$

where $W(e^{j\omega})$ is the Fourier transform of the data window. Note that with

$$U = \frac{1}{N}\sum_{n=0}^{N-1}|w(n)|^2 = \frac{1}{2\pi N}\int_{-\pi}^{\pi}|W(e^{j\omega})|^2 d\omega \tag{8.52}$$

The Modified Periodogram

```
function Px = mper(x,win,n1,n2)
%
   x  = x(:);
   if nargin == 2
      n1 = 1;  n2 = length(x);  end;
   N  = n2 - n1 + 1;
   w  = ones(N,1);
   if (win == 2) w = hamming(N);
      elseif (win == 3) w = hanning(N);
      elseif (win == 4) w = bartlett(N);
      elseif (win == 5) w = blackman(N);
      end;
   xw = x(n1:n2).*w/norm(w);
   Px = N*periodogram(xw);
   end;
```

Figure 8.11 *A* MATLAB *program for computing the modified periodogram of* $x(n)$.

then

$$\frac{1}{2\pi NU}\int_{-\pi}^{\pi}|W(e^{j\omega})|^2 d\omega = 1$$

and, with an appropriate window, $|W(e^{j\omega})|^2/NU$ will converge to an impulse of unit area as $N \rightarrow \infty$, and the modified periodogram will be asymptotically unbiased. (Note that if $w(n)$ is a rectangular window, then $U = 1$ and the modified periodogram reduces to the periodogram).

Since the modified periodogram is simply the periodogram of a windowed data sequence, the variance of $\hat{P}_M(e^{j\omega})$ will be approximately the same as that for the periodogram, i.e.,

$$\boxed{\text{Var}\left\{\hat{P}_M(e^{j\omega})\right\} \approx P_x^2(e^{j\omega})} \tag{8.53}$$

Therefore, the modified periodogram is not a consistent estimate of the power spectrum and the data window offers no benefit in terms of reducing the variance. What the window does provide, however, is a trade-off between spectral resolution (main lobe width) and spectral masking (sidelobe amplitude). For example, with the resolution of $\hat{P}_M(e^{j\omega})$ defined to be the 3 dB bandwidth of the data window,[5]

$$\boxed{\text{Res}\left[\hat{P}_M(e^{j\omega})\right] = (\Delta\omega)_{3\text{dB}}} \tag{8.54}$$

we see from Table 8.2 that a 43 dB reduction in the sidelobe amplitude that comes from using a Hamming data window instead of a rectangular window results in a reduction in the spectral resolution of about 50%. The properties of the modified periodogram are summarized in Table 8.3. An extensive list of windows and window characteristics may be found in [14].

Table 8.2 ***Properties of a Few Commonly Used Windows. Each Window is Assumed to be of Length N.***

Window	Sidelobe Level (dB)	3 dB BW $(\Delta\omega)_{3\text{dB}}$
Rectangular	−13	$0.89(2\pi/N)$
Bartlett	−27	$1.28(2\pi/N)$
Hanning	−32	$1.44(2\pi/N)$
Hamming	−43	$1.30(2\pi/N)$
Blackman	−58	$1.68(2\pi/N)$

[5]Note that this is consistent with the definition of resolution given in Eq. (8.28) for the periodogram. Specifically, although the periodogram resolution is defined to be the 6 dB bandwidth of $W_B(e^{j\omega})$, since $W_B(e^{j\omega}) = |W_R(e^{j\omega})|^2$, this is equivalent to the 3 dB bandwidth of $W_R(e^{j\omega})$.

Table 8.3 *Properties of the Modified Periodogram*

$$\hat{P}_M(e^{j\omega}) = \frac{1}{NU}\left|\sum_{n=-\infty}^{\infty} w(n)x(n)e^{-jn\omega}\right|^2$$

$$U = \frac{1}{N}\sum_{n=0}^{N-1}|w(n)|^2$$

Bias

$$E\left\{\hat{P}_M(e^{j\omega})\right\} = \frac{1}{2\pi NU}P_x(e^{j\omega}) * |W(e^{j\omega})|^2$$

Resolution Window dependent

Variance

$$\text{Var}\left\{\hat{P}_M(e^{j\omega})\right\} \approx P_x^2(e^{j\omega})$$

8.2.4 Bartlett's Method: Periodogram Averaging

In this section, we look at Bartlett's method of periodogram averaging, which, unlike either the periodogram or the modified periodogram, produces a consistent estimate of the power spectrum [5]. The motivation for this method comes from the observation that the expected value of the periodogram converges to $P_x(e^{j\omega})$ as the data record length N goes to infinity,

$$\lim_{N\to\infty} E\{\hat{P}_{per}(e^{j\omega})\} = P_x(e^{j\omega}) \tag{8.55}$$

Therefore, if we can find a consistent estimate of the mean, $E\{\hat{P}_{per}(e^{j\omega})\}$, then this estimate will be a consistent estimate of $P_x(e^{j\omega})$.

In our discussion of the sample mean in Section 3.2.8, we saw how averaging a set of uncorrelated measurements of a random variable x yields a consistent estimate of the mean, $E\{x\}$. This suggests that we consider estimating the power spectrum of a random process by periodogram averaging. Thus, let $x_i(n)$ for $i = 1, 2, \ldots, K$ be K uncorrelated realizations of a random process $x(n)$ over the interval $0 \leq n < L$. With $\hat{P}_{per}^{(i)}(e^{j\omega})$ the periodogram of $x_i(n)$,

$$\hat{P}_{per}^{(i)}(e^{j\omega}) = \frac{1}{L}\left|\sum_{n=0}^{L-1} x_i(n)e^{-jn\omega}\right|^2 \quad ; \quad i = 1, 2, \ldots, K \tag{8.56}$$

the average of these periodograms is

$$\hat{P}_x(e^{j\omega}) = \frac{1}{K}\sum_{i=1}^{K}\hat{P}_{per}^{(i)}(e^{j\omega}) \tag{8.57}$$

Evaluating the expected value of $\hat{P}_x(e^{j\omega})$ we have

$$E\left\{\hat{P}_x(e^{j\omega})\right\} = E\left\{\hat{P}_{per}^{(i)}(e^{j\omega})\right\} = \frac{1}{2\pi}P_x(e^{j\omega}) * W_B(e^{j\omega}) \tag{8.58}$$

where $W_B(e^{j\omega})$ is the Fourier transform of a Bartlett window, $w_B(k)$, that extends from $-L$

to L. Therefore, as with the periodogram, $\hat{P}_x(e^{j\omega})$ is asymptotically unbiased. In addition, with our assumption that the data records are uncorrelated, it follows that the variance of $\hat{P}_x(e^{j\omega})$ is

$$\text{Var}\left\{\hat{P}_x(e^{j\omega})\right\} = \frac{1}{K}\text{Var}\left\{\hat{P}^{(i)}_{per}(e^{j\omega})\right\} \approx \frac{1}{K}P_x^2(e^{j\omega}) \tag{8.59}$$

which goes to zero as K goes to infinity. Therefore, $\hat{P}_x(e^{j\omega})$ is a consistent estimate of the power spectrum provided that both K and L are allowed to go to infinity. However, the difficulty with this approach is that uncorrelated realizations of a process are generally not available. Instead, one typically only has a single realization of length N. Therefore, Bartlett proposed that $x(n)$ be partitioned into K nonoverlapping sequences of length L where $N = KL$ as illustrated in Fig. 8.12. The Bartlett estimate is then computed as in Eq. (8.56) and Eq. (8.57) with

$$\begin{aligned} x_i(n) &= x(n + iL) \qquad n = 0, 1, \ldots, L-1 \\ &\qquad\qquad\quad i = 0, 1, \ldots, K-1 \end{aligned}$$

Thus, the Bartlett estimate is

$$\boxed{\hat{P}_B(e^{j\omega}) = \frac{1}{N}\sum_{i=0}^{K-1}\left|\sum_{n=0}^{L-1}x(n+iL)e^{-jn\omega}\right|^2} \tag{8.60}$$

A MATLAB program to compute the Bartlett estimate is given in Fig. 8.13.

Based on our analysis of the periodogram and the modified periodogram, we may easily evaluate the performance of Bartlett's method as follows. First, as in Eq. (8.58), the expected value of Bartlett's estimate is

$$\boxed{E\left\{\hat{P}_B(e^{j\omega})\right\} = \frac{1}{2\pi}P_x(e^{j\omega}) * W_B(e^{j\omega})} \tag{8.61}$$

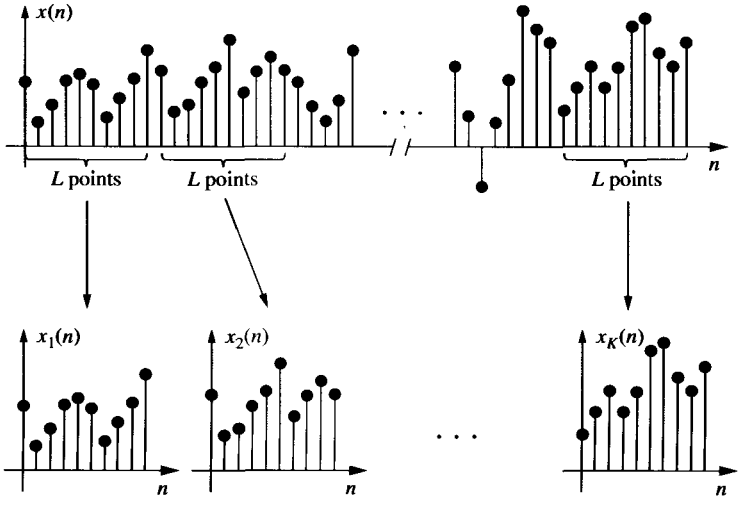


Figure 8.12 *Partitioning $x(n)$ into nonoverlapping subsequences.*

Bartlett's Method

```
function Px = bart(x,nsect)
%
   L  = floor(length(x)/nsect);
   Px = 0;
   n1 = 1;
   for i=1:nsect
       Px = Px + periodogram(x(n1:n1+L-1))/nsect;
       n1 = n1 + L;
       end;
```

Figure 8.13 *A* MATLAB *program for estimating the power spectrum using Bartlett's method of averaging periodograms. Note that this m-file calls* `periodogram.m`.

Therefore, $\hat{P}_B(e^{j\omega})$ is asymptotically unbiased. Second, since the periodograms used in $\hat{P}_B(e^{j\omega})$ are computed using sequences of length L, then the resolution is

$$\text{Res}\left[\hat{P}_B(e^{j\omega})\right] = 0.89\frac{2\pi}{L} = 0.89K\frac{2\pi}{N} \tag{8.62}$$

which is K times larger (worse) than the periodogram. Finally, since the sequences $x_i(n)$ are generally correlated with one another, unless $x(n)$ is white noise, then the variance reduction will not be as large as that given in Eq. (8.59). However, the variance will be inversely proportional to K and, assuming that the data sequences are approximately uncorrelated, for large N the variance is approximately

$$\text{Var}\left\{\hat{P}_B(e^{j\omega})\right\} \approx \frac{1}{K}\text{Var}\left\{\hat{P}_{per}^{(i)}(e^{j\omega})\right\} \approx \frac{1}{K}P_x^2(e^{j\omega}) \tag{8.63}$$

Thus, if both K and L are allowed to go to infinity as $N \to \infty$, then $\hat{P}_B(e^{j\omega})$ will be a consistent estimate of the power spectrum. In addition, for a given value of N, Bartlett's method allows one to trade a reduction in spectral resolution for a reduction in variance by simply changing the values of K and L. Table 8.4 summarizes the properties of Bartlett's method.

Table 8.4 *Properties of Bartlett's Method*

$$\hat{P}_B(e^{j\omega}) = \frac{1}{N}\sum_{i=0}^{K-1}\left|\sum_{n=0}^{L-1}x(n+iL)e^{-jn\omega}\right|^2$$

Bias

$$E\left\{\hat{P}_B(e^{j\omega})\right\} = \frac{1}{2\pi}P_x(e^{j\omega}) * W_B(e^{j\omega})$$

Resolution

$$\Delta\omega = 0.89K\frac{2\pi}{N}$$

Variance

$$\text{Var}\left\{\hat{P}_B(e^{j\omega})\right\} \approx \frac{1}{K}P_x^2(e^{j\omega})$$

Example 8.2.5 *Bartlett's Method*

In Example 8.2.4, we considered using the periodogram to estimate the power spectrum of white noise. What we observed was that the variance of the estimate did not decrease as we increased the length N of the data sequence. Shown in Fig. 8.14*a* are the periodograms ($K = 1$) of 50 different unit variance white noise sequences of length $N = 512$ and in Fig. 8.14*b* is the ensemble average. The Bartlett estimates for these sequences with $K = 4$ sections of length $L = 128$ are shown in Fig. 8.14*c* with the average given in Fig. 8.14*d*. Similarly, the Bartlett estimates of these 50 processes using $K = 16$ sections of length $L = 32$ are shown in Fig. 8.14*e* and the average in Fig. 8.14*f*. What we observe in these examples is that the variance of the estimate decreases in proportion to the number of sections K.

As another example, let $x(n)$ be a process consisting of two sinusoids in unit variance white noise,

$$x(n) = A\sin(n\omega_1 + \phi_1) + \sin(n\omega_2 + \phi_2) + v(n)$$

where $\omega_1 = 0.2\pi$, $\omega_2 = 0.25\pi$, and $A = \sqrt{10}$. With $N = 512$, an overlay plot of 50 periodograms is shown in Fig. 8.15*a* and the ensemble average of these periodograms is given in Fig. 8.15*b*. Using Bartlett's method with $K = 4$ and $K = 16$ sections, the overlay plots and ensemble averages are shown in Fig. 8.15*c-f*. Note that although the variance of the estimate decreases with K, there is a corresponding decrease in the resolution as evidenced by the broadening of the spectral peaks.

8.2.5 Welch's Method: Averaging Modified Periodograms

In 1967, Welch proposed two modifications to Bartlett's method [59]. The first is to allow the sequences $x_i(n)$ to overlap, and the second is to allow a data window $w(n)$ to be applied to each sequence, thereby producing a set of *modified periodograms* that are to be averaged. Assuming that successive sequences are offset by D points and that each sequence is L points long then the ith sequence is given by

$$x_i(n) = x(n + iD) \quad ; \quad n = 0, 1, \ldots, L-1$$

Thus, the amount of overlap between $x_i(n)$ and $x_{i+1}(n)$ is $L - D$ points, and if K sequences cover the entire N data points, then

$$N = L + D(K - 1).$$

For example, with no overlap ($D = L$) we have $K = N/L$ sections of length L as in Bartlett's method. On the other hand, if the sequences are allowed to overlap by 50% ($D = L/2$) then we may form

$$K = 2\frac{N}{L} - 1$$

sections of length L, thus maintaining the same resolution (section length) as Bartlett's method while doubling the number of modified periodograms that are averaged, thereby reducing the variance. However, with a 50% overlap we could also form

$$K = \frac{N}{L} - 1$$

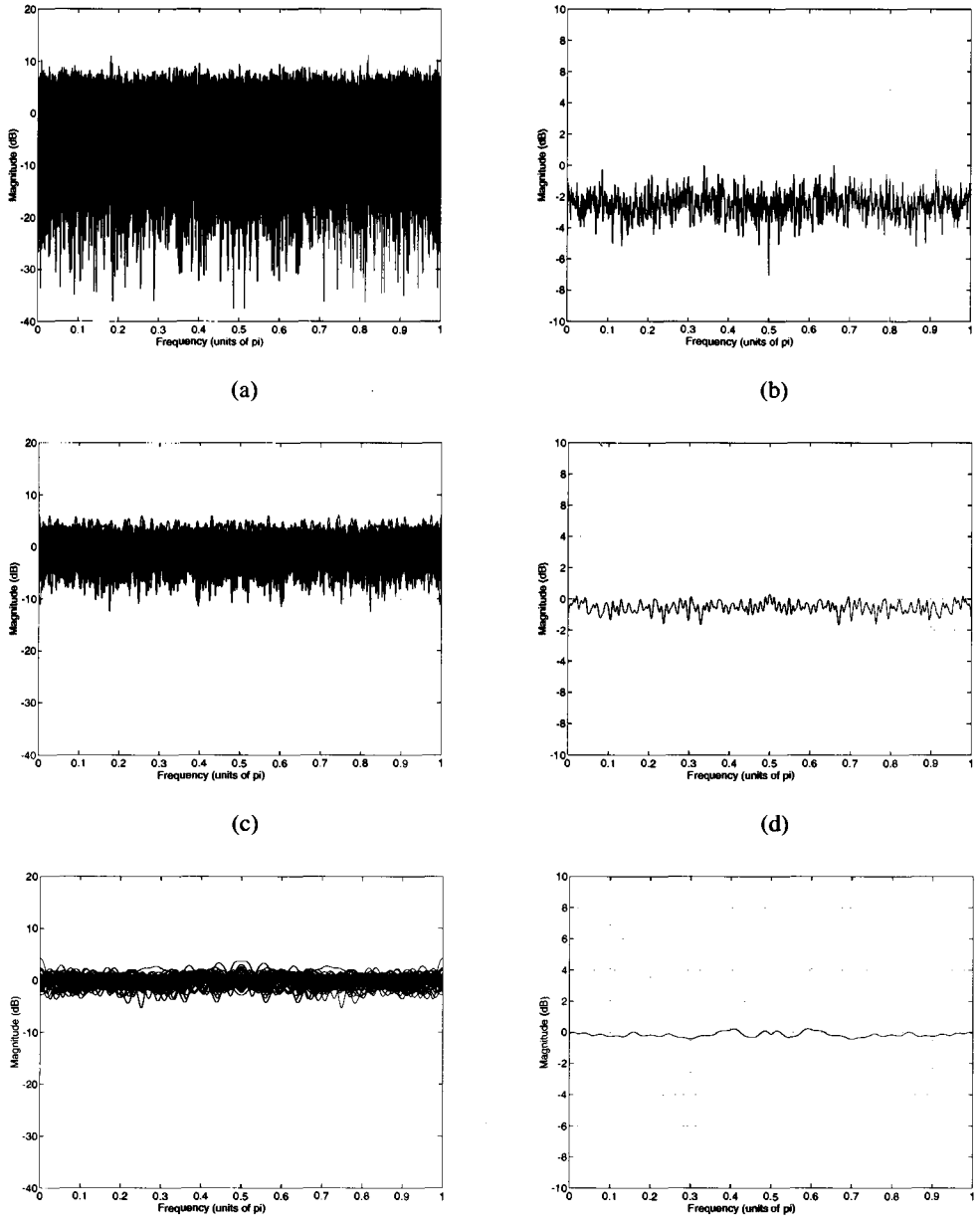


Figure 8.14 *Spectrum estimation of unit variance white Gaussian noise. (a) Overlay plot of 50 periodograms with $N = 512$ and (b) the ensemble average. (c) Overlay plot of 50 Bartlett estimates with $K = 4$ and $L = 128$ and (d) the ensemble average. (e) Overlay plot of 50 Bartlett estimates with $K = 8$ and $L = 64$ and (f) the ensemble average.*

sequences of length $2L$, thus increasing the resolution while maintaining the same variance as Bartlett's method. Therefore, by allowing the sequences to overlap it is possible to increase the number and/or length of the sequences that are averaged, thereby trading a

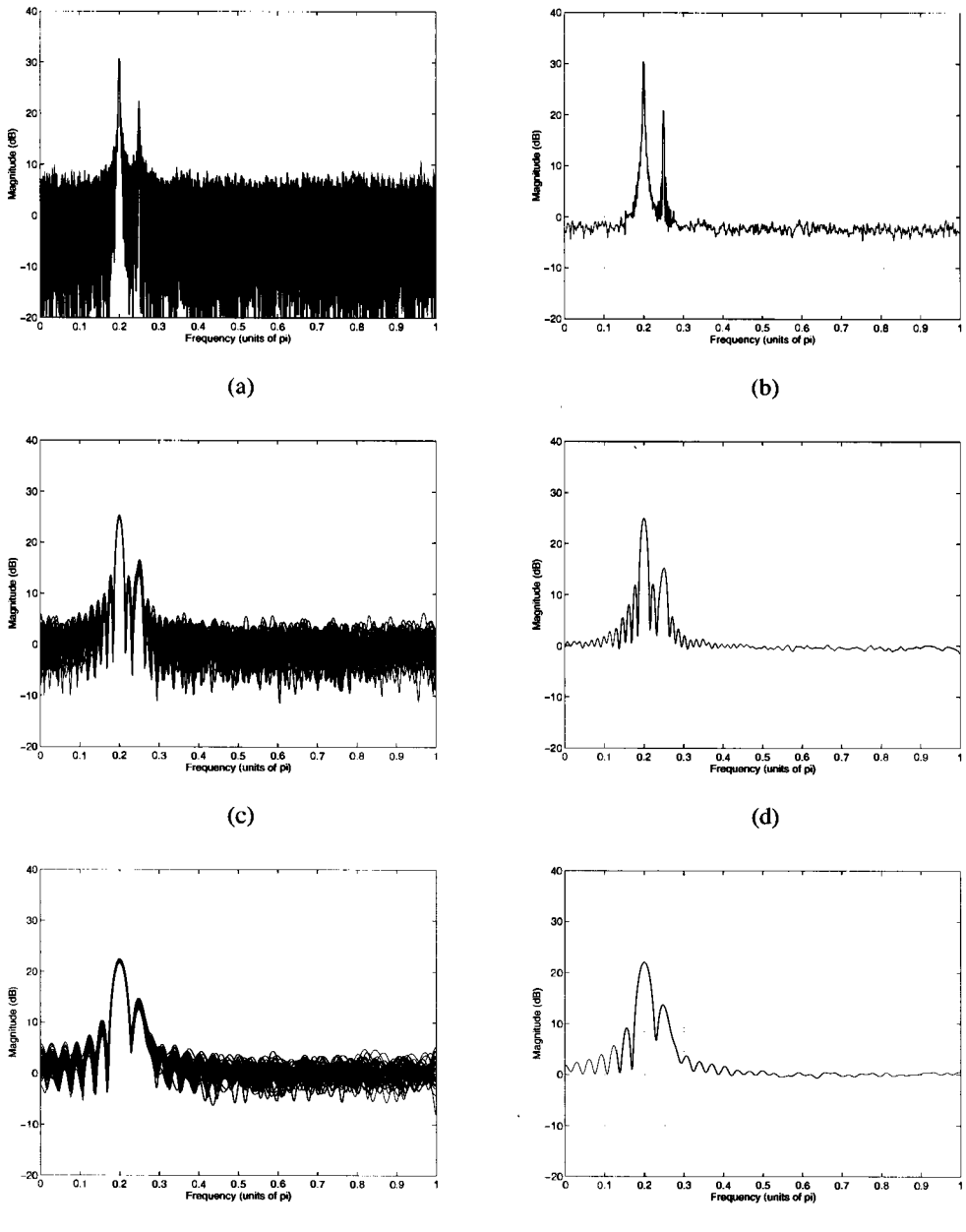


Figure 8.15 *Spectrum estimation of two sinusoids in white noise using* $N = 512$ *data values. (a) Overlay plot of 50 periodograms and (b) the ensemble average. (c) Overlay plot of 50 Bartlett estimates with* $K = 4$ *and* $L = 128$ *and (d) the ensemble average. (e) Overlay plot of 50 Bartlett estimates with* $K = 8$ *and* $L = 64$ *and (f) the ensemble average.*

reduction in variance for a reduction in resolution. A MATLAB program for estimating the power spectrum using Welch's method is given in Fig. 8.16.

Let us now evaluate the performance of Welch's method. Note that the estimate produced

Welch's Method

```
function Px = welch(x,L,over,win)
%
   if (over >= 1) | (over < 0)
      error('Overlap is invalid'), end
   n1 = 1;
   n0 = (1-over)*L;
   nsect=1+floor((length(x)-L)/(n0));
   Px=0;
   for i=1:nsect
       Px = Px + mper(x,win,n1,n1+L-1)/nsect;
       n1 = n1 + n0;
       end;
```

Figure 8.16 *A* MATLAB *program for estimating the power spectrum using Welch's method of averaging modified periodograms. Note that this m-file calls* `mper.m`.

with Welch's method may be written explicitly in terms of $x(n)$ as follows

$$\hat{P}_W(e^{j\omega}) = \frac{1}{KLU} \sum_{i=0}^{K-1} \left| \sum_{n=0}^{L-1} w(n)x(n+iD)e^{-jn\omega} \right|^2 \tag{8.64}$$

or, more succinctly, in terms of modified periodograms as

$$\hat{P}_W(e^{j\omega}) = \frac{1}{K} \sum_{i=0}^{K-1} \hat{P}_M^{(i)}(e^{j\omega}) \tag{8.65}$$

Therefore, the expected value of Welch's estimate is

$$E\{\hat{P}_W(e^{j\omega})\} = E\{\hat{P}_M(e^{j\omega})\} = \frac{1}{2\pi LU} P_x(e^{j\omega}) * |W(e^{j\omega})|^2 \tag{8.66}$$

where $W(e^{j\omega})$ is the Fourier transform of the L-point data window, $w(n)$, used in Eq. (8.64) to form the modified periodograms. Thus, as with each of the previous periodogram-based methods, Welch's method is an asymptotically unbiased estimate of the power spectrum. The resolution, however, depends on the data window. As with the modified periodogram, the resolution is defined to be the 3 dB bandwidth of the data window (see Table 8.2). The variance, on the other hand, is more difficult to compute since, with overlapping sequences, the modified periodograms cannot be assumed to be uncorrelated. Nevertheless, it has been shown that, with a Bartlett window and a 50% overlap, the variance is approximately [59]

$$\text{Var}\{\hat{P}_W(e^{j\omega})\} \approx \frac{9}{8K} P_x^2(e^{j\omega}) \tag{8.67}$$

Comparing Eqs. (8.67) and (8.63) we see that, for a given number of sections K, the variance of the estimate using Welch's method is larger than that for Bartlett's method by a factor of 9/8. However, for a fixed amount of data, N, and a given resolution (sequence length L), with a 50% overlap twice as many sections may be averaged in Welch's method. Expressing

Table 8.5 *Properties of Welch's Method*

$$\hat{P}_W(e^{j\omega}) = \frac{1}{KLU}\sum_{i=0}^{K-1}\left|\sum_{n=0}^{L-1} w(n)x(n+iD)e^{-jn\omega}\right|^2$$

$$U = \frac{1}{L}\sum_{n=0}^{L-1}|w(n)|^2$$

Bias

$$E\left\{\hat{P}_W(e^{j\omega})\right\} = \frac{1}{2\pi LU}P_x(e^{j\omega}) * |W(e^{j\omega})|^2$$

Resolution Window dependent

Variance†

$$\text{Var}\left\{\hat{P}_W(e^{j\omega})\right\} \approx \frac{9}{16}\frac{L}{N}P_x^2(e^{j\omega})$$

† Assuming 50% overlap and a Bartlett window.

the variance in terms of L and N, we have (assuming a 50% overlap)

$$\boxed{\text{Var}\left\{\hat{P}_W(e^{j\omega})\right\} \approx \frac{9}{16}\frac{L}{N}P_x^2(e^{j\omega})} \tag{8.68}$$

Since N/L is the number of sections that are used in Bartlett's method, it follows from Eq. (8.63) that

$$\text{Var}\left\{\hat{P}_W(e^{j\omega})\right\} \approx \frac{9}{16}\text{Var}\{\hat{P}_B(e^{j\omega})\}$$

Although it is possible to average more sequences for a given amount of data by increasing the amount of overlap, the computational requirements increase in proportion with K. In addition, since an increase in the amount of overlap increases the correlation between the sequences $x_i(n)$, there are diminishing returns when increasing K for a fixed N. Therefore the amount of overlap is typically either 50% or 75%. The properties of Welch's method are summarized in Table 8.5.

Example 8.2.6 ***Welch's Method***

Consider the process defined in Example 8.2.5 consisting of two sinusoids in unit variance white noise. Using Welch's method with $N = 512$, a section length $L = 128$, a 50% overlap (7 sections), and a Hamming window, an overlay plot of the spectrum estimates for 50 different realizations of the process are shown in Fig. 8.17*a* and the ensemble average is shown in Fig. 8.17*b*. Comparing these estimates with those shown in Fig. 8.15*e* and *f* of Example 8.2.5, we see that, since the number of sections used in both examples are about the same (7 versus 8), then the variance of the two estimates are approximately the same. In addition, although the width of the main lobe of the Hamming window used in Welch's method is 1.46 times the width of the rectangular window used in Bartlett's method, the resolution is about the same. The reason for this is due to the fact that the 50% overlap that

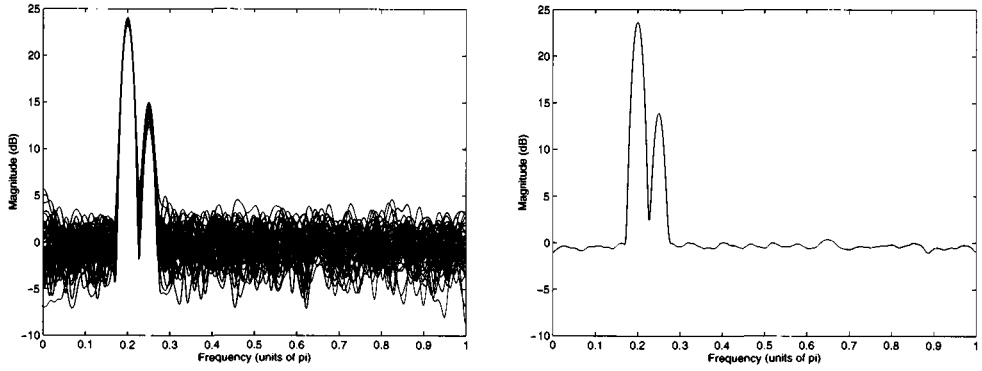


Figure 8.17 *(a) An overlay plot of 50 estimates of the spectrum of two sinusoids in noise using Welch's method with $N = 512$, a section length of $L = 128$, 50% overlap (7 sections), and a Hamming window. (b) The average of the estimates in (a).*

is used in Welch's method allows for the section length to be twice the length of that used in Bartlett's method. So what do we gain with Welch's method? We gain a reduction in the amount of spectral leakage that takes place through the sidelobes of the data window.

8.2.6 Blackman-Tukey Method: Periodogram Smoothing

The methods of Bartlett and Welch are designed to reduce the variance of the periodogram by averaging periodograms and modified periodograms, respectively. Another method for decreasing the statistical variability of the periodogram is periodogram smoothing, often referred to as the Blackman-Tukey method after the pioneering work of Blackman and Tukey in spectrum analysis [6]. To see how periodogram smoothing may reduce the variance of the periodogram, recall that the periodogram is computed by taking the Fourier transform of a consistent estimate of the autocorrelation sequence. However, for any finite data record of length N, the variance of $\hat{r}_x(k)$ will be large for values of k that are close to N. For example, note that the estimate of $r_x(k)$ at lag $k = N - 1$ is

$$\hat{r}_x(N-1) = \frac{1}{N}x(N-1)x(0)$$

Since there is little averaging that goes into the formation of the estimates of $r_x(k)$ for $|k| \approx N$, no matter how large N becomes, these estimates will always be unreliable. Consequently, the only way to reduce the variance of the periodogram is to reduce the variance of these estimates or to reduce the contribution that they make to the periodogram. In the methods of Bartlett and Welch, the variance of the periodogram is decreased by reducing the variance of the autocorrelation estimate by averaging. In the Blackman-Tukey method, the variance of the periodogram is reduced by applying a window to $\hat{r}_x(k)$ in order to decrease the contribution of the unreliable estimates to the periodogram. Specifically, the Blackman-Tukey spectrum estimate is

$$\hat{P}_{BT}(e^{j\omega}) = \sum_{k=-M}^{M} \hat{r}_x(k)w(k)e^{-jk\omega} \tag{8.69}$$

where $w(k)$ is a *lag window* that is applied to the autocorrelation estimate. For example, if $w(k)$ is a rectangular window extending from $-M$ to M with $M < N-1$, then the estimates of $r_x(k)$ having the largest variance are set to zero and, consequently, the power spectrum estimate will have a smaller variance. What is traded for this reduction in variance, however, is a reduction in resolution since a smaller number of autocorrelation estimates are used to form the estimate of the power spectrum.

Using the frequency convolution theorem, the Blackman-Tukey spectrum may be written in the frequency domain as follows:

$$\hat{P}_{BT}(e^{j\omega}) = \frac{1}{2\pi}\hat{P}_{per}(e^{j\omega}) * W(e^{j\omega}) = \frac{1}{2\pi}\int_{-\pi}^{\pi}\hat{P}_{per}(e^{ju})W(e^{j(\omega-u)})du \qquad (8.70)$$

Therefore, the Blackman-Tukey estimate smoothes the periodogram by convolving with the Fourier transform of the autocorrelation window, $W(e^{j\omega})$. Although there is considerable flexibility in the choice of the window that may be used, $w(k)$ should be conjugate symmetric so that $W(e^{jw})$ is real-valued, and the window should have a nonnegative Fourier transform, $W(e^{j\omega}) \geq 0$, so that $\hat{P}_{BT}(e^{j\omega})$ is guaranteed to be nonnegative. A MATLAB program for generating the Blackman-Tukey spectrum estimate is given in Fig. 8.18.

To analyze the performance of the Blackman-Tukey method, we will evaluate the bias and the variance (the resolution is window-dependent). The bias may be computed by taking the expected value of Eq. (8.70) as follows:

$$E\{\hat{P}_{BT}(e^{j\omega})\} = \frac{1}{2\pi}E\{\hat{P}_{per}(e^{j\omega})\} * W(e^{j\omega}) \qquad (8.71)$$

Blackman-Tukey Method

```
function Px = per_smooth(x,win,M,n1,n2)
%
   x   = x(:);
   if nargin == 3
       n1 = 1;  n2 = length(x);  end;
   R   = covar(x(n1:n2),M);
   r   = [fliplr(R(1,2:M)),R(1,1),R(1,2:M)];
   M   = 2*M-1;
   w   = ones(M,1);
   if (win == 2) w = hamming(M);
       elseif (win == 3) w = hanning(M);
       elseif (win == 4) w = bartlett(M);
       elseif (win == 5) w = blackman(M);
       end;
   r = r'.*w;
   Px = abs(fft(r,1024));
   Px(1)=Px(2);
   end;
```

Figure 8.18 *A* MATLAB *program for estimating the power spectrum using the Blackman-Tukey method (periodogram smoothing).*

Substituting Eq. (8.23) for the expected value of the periodogram we have

$$E\left\{\hat{P}_{BT}(e^{j\omega})\right\} = \frac{1}{2\pi} P_x(e^{j\omega}) * W_B(e^{j\omega}) * W(e^{j\omega}) \tag{8.72}$$

or, equivalently,

$$E\left\{\hat{P}_{BT}(e^{j\omega})\right\} = \sum_{k=-M}^{M} r_x(k) w_B(k) w(k) e^{-jk\omega} \tag{8.73}$$

If we let $w_{BT}(k) = w_B(k)w(k)$ be the combined window that is applied to the autocorrelation sequence $r_x(k)$, using the frequency convolution theorem we have

$$E\left\{\hat{P}_{BT}(e^{j\omega})\right\} = \frac{1}{2\pi} P_x(e^{j\omega}) * W_{BT}(e^{j\omega}) \tag{8.74}$$

If we assume that $M \ll N$ so that $w_B(k)w(k) \approx w(k)$, then we have

$$\boxed{E\left\{\hat{P}_{BT}(e^{j\omega})\right\} \approx \frac{1}{2\pi} P_x(e^{j\omega}) * W(e^{j\omega})} \tag{8.75}$$

where $W(e^{j\omega})$ is the Fourier transform of the lag window $w(k)$.

Evaluating the variance of the Blackman-Tukey spectrum estimate requires a a bit more work.[6] Since

$$\text{Var}\left\{\hat{P}_{BT}(e^{j\omega})\right\} = E\left\{\hat{P}^2_{BT}(e^{j\omega})\right\} - E^2\left\{\hat{P}_{BT}(e^{j\omega})\right\} \tag{8.76}$$

we begin by finding the mean-square value $E\left\{\hat{P}^2_{BT}(e^{j\omega})\right\}$. From Eq. (8.70) we have

$$\hat{P}^2_{BT}(e^{j\omega}) = \frac{1}{4\pi^2} \int_{-\pi}^{\pi}\int_{-\pi}^{\pi} \hat{P}_{per}(e^{ju}) \hat{P}_{per}(e^{jv}) W(e^{j(\omega-u)}) W(e^{j(\omega-v)}) du dv$$

Therefore, the mean-square value is

$$E\left\{\hat{P}^2_{BT}(e^{j\omega})\right\} = \frac{1}{4\pi^2} \int_{-\pi}^{\pi}\int_{-\pi}^{\pi} E\left\{\hat{P}_{per}(e^{ju}) \hat{P}_{per}(e^{jv})\right\} W(e^{j(\omega-u)}) W(e^{j(\omega-v)}) du dv$$

Using the approximation given in Eq. (8.43) for $E\left\{\hat{P}_{per}(e^{ju}) \hat{P}_{per}(e^{jv})\right\}$ leads to an expression for the mean-square value that contains two terms. The first term is

$$\begin{aligned} &\frac{1}{4\pi^2} \int_{-\pi}^{\pi}\int_{-\pi}^{\pi} P_x(e^{ju}) P_x(e^{jv}) W(e^{j(\omega-u)}) W(e^{j(\omega-v)}) du dv \\ &\qquad = \left[\frac{1}{2\pi} \int_{-\pi}^{\pi} P_x(e^{ju}) W(e^{j(\omega-u)}) du\right]^2 = E^2\left\{\hat{P}_{BT}(e^{j\omega})\right\} \end{aligned} \tag{8.77}$$

which is cancelled by the second term in Eq. (8.76). Therefore, the variance is

$$\begin{aligned} \text{Var}\left\{\hat{P}_{BT}(e^{j\omega})\right\} = \frac{1}{4\pi^2} &\int_{-\pi}^{\pi}\int_{-\pi}^{\pi} P_x(e^{ju}) P_x(e^{jv}) \\ &\times \left[\frac{\sin N(u-v)/2}{N \sin(u-v)/2}\right]^2 W(e^{j(\omega-u)}) W(e^{j(\omega-v)}) du dv \end{aligned} \tag{8.78}$$

[6]The reader may jump to the final result given in Eq. (8.79) without any loss in continuity.

Since

$$W_B(e^{j\omega}) = \frac{1}{N}\left[\frac{\sin(N\omega/2)}{\sin(\omega/2)}\right]^2$$

is the discrete-time Fourier transform of a Bartlett window, then $w_B(k)$ approaches a constant as $N \to \infty$ and $W_B(e^{j\omega})$ converges to an impulse. Therefore, if N is large then the term in brackets may be approximated by an impulse of area $2\pi/N$,

$$\left[\frac{\sin N(u-v)/2}{N\sin(u-v)/2}\right]^2 \approx \frac{2\pi}{N}u_0(u-v)$$

Thus, for large N, the variance of the Blackman-Tukey estimate is approximately

$$\text{Var}\left\{\hat{P}_{BT}(e^{j\omega})\right\} \approx \frac{1}{2\pi N}\int_{-\pi}^{\pi} P_x^2(e^{ju})W^2(e^{j(\omega-u)})du$$

If M is large enough so that we may assume that $P_x(e^{j\omega})$ is constant across the main lobe of $W(e^{j(\omega-u)})$, then $P_x^2(e^{j\omega})$ may be pulled out of the integral,

$$\text{Var}\left\{\hat{P}_{BT}(e^{j\omega})\right\} \approx \frac{1}{2\pi N}P_x^2(e^{j\omega})\int_{-\pi}^{\pi} W^2(e^{j(\omega-u)})du$$

Finally, using Parseval's theorem we have

$$\boxed{\text{Var}\left\{\hat{P}_{BT}(e^{j\omega})\right\} \approx P_x^2(e^{j\omega})\frac{1}{N}\sum_{k=-M}^{M} w^2(k)} \tag{8.79}$$

provided $N \gg M \gg 1$. Thus, from Eqs. (8.75) and (8.79) we again see the trade-off between bias and variance. For a small bias, M should be large in order to minimize the width of the main lobe of $W(e^{jw})$ whereas M should be small in order to minimize the sum in Eq. (8.79). Generally, it is recommended that M have a maximum value of $M = N/5$ [26]. The properties of the Blackman-Tukey method are summarized in Table 8.6.

Table 8.6 *Properties of the Blackman-Tukey Method*

$$\hat{P}_{BT}(e^{j\omega}) = \sum_{k=-M}^{M} \hat{r}_x(k)w(k)e^{-jk\omega}$$

Bias

$$E\left\{\hat{P}_{BT}(e^{j\omega})\right\} \approx \frac{1}{2\pi}P_x(e^{j\omega}) * W(e^{j\omega})$$

Resolution Window dependent

Variance

$$\text{Var}\left\{\hat{P}_{BT}(e^{j\omega})\right\} \approx P_x^2(e^{j\omega})\frac{1}{N}\sum_{k=-M}^{M} w^2(k)$$

8.2.7 Performance Comparisons

An important issue in the selection of a spectrum estimation technique is the performance of the estimator. In the previous sections, we have seen that, in comparing one nonparametric method to another, there is a trade-off between resolution and variance. This section summarizes the performance of each nonparametric technique in terms of two criteria. The first is the *variability* of the estimate,

$$\mathcal{V} = \frac{\text{Var}\left\{\hat{P}_x(e^{j\omega})\right\}}{E^2\left\{\hat{P}_x(e^{j\omega})\right\}}$$

which is a *normalized variance*. The second measure is an overall *figure of merit* that is defined as the product of the variability and the resolution,

$$\mathcal{M} = \mathcal{V}\Delta\omega$$

This figure of merit should be as small as possible. However, as we will soon discover, this figure of merit is approximately the same for all of the nonparametric methods that we have considered.

Periodogram. In Section 8.2.2 it was shown that the periodogram is asymptotically unbiased and that, for large N, the variance is approximately equal to $P_x^2(e^{j\omega})$. Asymptotically, therefore, the variability of the periodogram is equal to one

$$\mathcal{V}_{per} = \frac{P_x^2(e^{j\omega})}{P_x^2(e^{j\omega})} = 1$$

Thus, since the resolution of the periodogram is

$$\Delta\omega = 0.89\frac{2\pi}{N}$$

then the overall figure of merit is

$$\mathcal{M}_{per} = 0.89\frac{2\pi}{N}$$

which is inversely proportional to the data record length, N.

Bartlett's Method. In Bartlett's method, a reduction in variance is achieved by averaging periodograms. With $N = KL$, if N is large then the variance is approximately

$$\text{Var}\left\{\hat{P}_B(e^{j\omega})\right\} \approx \frac{1}{K}P_x^2(e^{j\omega})$$

and the variability is

$$\mathcal{V}_B = \frac{1}{K}\frac{P_x^2(e^{j\omega})}{P_x^2(e^{j\omega})} = \frac{1}{K}$$

Since the resolution is $\Delta\omega = 0.89(2\pi K/N)$ then the figure of merit is

$$\mathcal{M}_B = 0.89\frac{2\pi}{N}$$

which is the same as the figure of merit for the periodogram.

Welch's Method. The statistical properties of Welch's method depend on the amount of overlap that is used and on the type of data window. With a 50% overlap and a Bartlett window, the variability for large N is approximately

$$\mathcal{V}_W = \frac{9}{8}\frac{1}{K} = \frac{9}{16}\frac{L}{N}$$

Since the 3 dB bandwidth of a Bartlett window of length L is $1.28(2\pi/L)$ the resolution is

$$\Delta\omega = 1.28\frac{2\pi}{L}$$

and the figure of merit becomes

$$\mathcal{M}_W = 0.72\frac{2\pi}{N}$$

Blackman-Tukey Method. Since the variance and resolution of the Blackman-Tukey method depend on the window that is used, suppose $w(k)$ is a Bartlett window of length $2M$ that extends from $k = -M$ to $k = M$. Assuming that $N \gg M \gg 1$, the variance of the Blackman-Tukey estimate is approximately

$$\text{Var}\left\{\hat{P}_{BT}(e^{j\omega})\right\} \approx P_x^2(e^{j\omega})\frac{1}{N}\sum_{k=-M}^{M}\left(1-\frac{|k|}{M}\right)^2 \approx P_x^2(e^{j\omega})\frac{2M}{3N}$$

(here we have used the series given in Table 2.3 on p. 16 to evaluate the sum). Therefore, the variability is

$$\mathcal{V}_{BT} = \frac{2M}{3N}$$

Since the 3 dB bandwidth of a Bartlett window of length $2M$ is equal to $1.28(2\pi/2M)$, then the resolution is

$$\Delta\omega = 0.64\frac{2\pi}{M}$$

and the figure of merit becomes

$$\mathcal{M}_{BT} = 0.43\frac{2\pi}{N}$$

which is slightly smaller than the figure of merit for Welch's method.

Summary. Table 8.7 provides a summary of the performance measures presented above for the periodogram-based spectrum estimation techniques discussed in this section. What is apparent from this table is that each technique has a figure of merit that is approximately the same, and that these figures of merit are inversely proportional to the length of the data sequence, N. Therefore, although each method differs in its resolution and variance, the overall performance is fundamentally limited by the amount of data that is available. In the following sections, we will look at some entirely different approaches to spectrum estimation in the hope of being able to find a high-resolution spectrum estimate with a small variance that works well on short data records.

Table 8.7 *Performance Measures for the Nonparametric Methods of Spectrum Estimation*

	Variability $\mathcal{V}$	Resolution $\Delta\omega$	Figure of Merit $\mathcal{M}$
Periodogram	1	$0.89\frac{2\pi}{N}$	$0.89\frac{2\pi}{N}$
Bartlett	$\frac{1}{K}$	$0.89K\frac{2\pi}{N}$	$0.89\frac{2\pi}{N}$
Welch†	$\frac{9}{8}\frac{1}{K}$	$1.28\frac{2\pi}{L}$	$0.72\frac{2\pi}{N}$
Blackman-Tukey	$\frac{2}{3}\frac{M}{N}$	$0.64\frac{2\pi}{M}$	$0.43\frac{2\pi}{N}$

† 50% overlap and a Bartlett window.

8.3 MINIMUM VARIANCE SPECTRUM ESTIMATION

Up to this point, we have been considering nonparametric techniques for estimating the power spectrum of a random process. Relying on the DTFT of an estimated autocorrelation sequence, the performance of these methods is limited by the length of the data record. In this section, we develop the Minimum Variance (MV) method of spectrum estimation, which is an adaptation of the Maximum Likelihood Method (MLM) developed by Capon for the analysis of two-dimensional power spectral densities [8]. In the MV method, the power spectrum is estimated by filtering a process with a bank of narrowband bandpass filters. The motivation for this approach may be seen by looking, once again, at the effect of filtering a WSS random process with a narrowband bandpass filter. Therefore, let $x(n)$ be a zero mean wide-sense stationary random process with a power spectrum $P_x(e^{j\omega})$ and let $g_i(n)$ be an ideal bandpass filter with a bandwidth Δ and center frequency ω_i,

$$|G_i(e^{j\omega})| = \begin{cases} 1 & ; \quad |\omega - \omega_i| < \Delta/2 \\ 0 & ; \quad \text{otherwise} \end{cases}$$

If $x(n)$ is filtered with $g_i(n)$, then the power spectrum of the output process, $y_i(n)$, is

$$P_i(e^{j\omega}) = P_x(e^{j\omega})|G_i(e^{j\omega})|^2$$

and the power in $y_i(n)$ is

$$\begin{aligned} E\left\{|y_i(n)|^2\right\} &= \frac{1}{2\pi}\int_{-\pi}^{\pi} P_i(e^{j\omega})d\omega = \frac{1}{2\pi}\int_{-\pi}^{\pi} P_x(e^{j\omega})|G_i(e^{j\omega})|^2 d\omega \\ &= \frac{1}{2\pi}\int_{\omega_i-\Delta/2}^{\omega_i+\Delta/2} P_x(e^{j\omega})d\omega \end{aligned} \tag{8.80}$$

If Δ is small enough so that $P_x(e^{j\omega})$ is approximately constant over the passband of the filter, then the power in $y_i(n)$ is approximately

$$E\left\{|y_i(n)|^2\right\} \approx P_x(e^{j\omega_i})\frac{\Delta}{2\pi} \tag{8.81}$$

Therefore, it is possible to estimate the power spectral density of $x(n)$ at frequency $\omega = \omega_i$ from the filtered process by estimating the power in $y_i(n)$ and dividing by the normalized filter bandwidth, $\Delta/2\pi$,

$$\hat{P}_x(e^{j\omega_i}) = \frac{E\left\{|y_i(n)|^2\right\}}{\Delta/2\pi} \tag{8.82}$$

As we saw in Section 8.2.1, the periodogram produces an estimate of the power spectrum in a similar fashion. Specifically, $x(n)$ is filtered with a bank of bandpass filters, $h_i(n)$, where

$$|H_i(e^{j\omega})| = \frac{\sin[N(\omega - \omega_i)/2]}{N \sin[(\omega - \omega_i)/2]} \tag{8.83}$$

and the power in each of the filtered signals is estimated using a one-point sample average,

$$\hat{E}\left\{|y_i(n)|^2\right\} = |y_i(N-1)|^2$$

The periodogram is then formed by dividing this power estimate by the filter bandwidth, $\Delta = 2\pi/N$. Since each filter in the filter bank for the periodogram is the same, differing only in the center frequency, these filters are *data independent*. As a result, when a random process contains a significant amount of power in frequency bands within the sidelobes of the bandpass filter, leakage through the sidelobes will lead to significant distortion in the power estimates. Therefore, a better approach would be to allow each filter in the filter bank to be *data adaptive* so that each filter may be designed to be "optimum" in the sense of rejecting as much out-of-band signal power as possible. The minimum variance spectrum estimation technique described in this section is based on this idea and involves the following steps:

1. Design a bank of bandpass filters $g_i(n)$ with center frequency ω_i so that each filter rejects the maximum amount of out-of-band power while passing the component at frequency ω_i with no distortion.
2. Filter $x(n)$ with each filter in the filter bank and estimate the power in each output process $y_i(n)$.
3. Set $\hat{P}_x(e^{j\omega_i})$ equal to the power estimated in step (2) divided by the filter bandwidth.

To derive the minimum variance spectrum estimate, we begin with the design of the bandpass filter bank.

Suppose that we would like to estimate the power spectral density of $x(n)$ at frequency ω_i. Let $g_i(n)$ be a complex-valued FIR bandpass filter of order p. To ensure that the filter does not alter the power in the input process at frequency ω_i, $G_i(e^{j\omega})$ will be constrained to have a gain of one at $\omega = \omega_i$,

$$G_i(e^{j\omega_i}) = \sum_{n=0}^{p} g_i(n)e^{-jn\omega_i} = 1 \tag{8.84}$$

Let $\mathbf{g}_i$ be the vector of filter coefficients $g_i(n)$,

$$\mathbf{g}_i = \left[g_i(0), g_i(1), \ldots, g_i(p)\right]^T$$

and let $\mathbf{e}_i$ be the vector of complex exponentials $e^{jk\omega_i}$,

$$\mathbf{e}_i = \left[1, e^{j\omega_i}, \ldots, e^{jp\omega_i}\right]^T$$

The constraint on the frequency response given in Eq. (8.84) may be written in vector form as follows[7]

$$\mathbf{g}_i^H \mathbf{e}_i = \mathbf{e}_i^H \mathbf{g}_i = 1 \tag{8.85}$$

Now, in order for the power spectrum of $x(n)$ at frequency ω_i to be measured as accurately as possible, the bandpass filter should reject as much out-of-band power as possible. Therefore, the criterion that will be used to design the bandpass filter will be to minimize the power in the output process subject to the linear constraint given in Eq. (8.85). Since the power in $y_i(n)$ may be expressed in terms of the autocorrelation matrix $\mathbf{R}_x$ as follows (see Eq. (3.90) on p. 101)

$$E\{|y_i(n)|^2\} = \mathbf{g}_i^H \mathbf{R}_x \mathbf{g}_i \tag{8.86}$$

the filter design problem becomes one of minimizing Eq. (8.86) subject to the linear constraint given in Eq. (8.85). As we saw in Section 2.3.10, the solution to this problem is

$$\mathbf{g}_i = \frac{\mathbf{R}_x^{-1}\mathbf{e}_i}{\mathbf{e}_i^H \mathbf{R}_x^{-1}\mathbf{e}_i} \tag{8.87}$$

where the minimum value of $E\{|y_i(n)|^2\}$ is equal to

$$\min_{\mathbf{g}_i} E\{|y_i(n)|^2\} = \frac{1}{\mathbf{e}_i^H \mathbf{R}_x^{-1}\mathbf{e}_i} \tag{8.88}$$

Thus, Eq. (8.87) defines the optimum filter for estimating the power in $x(n)$ at frequency ω_i, and Eq. (8.88) gives the power in $y_i(n)$, which is used as the estimate, $\hat{\sigma}_x^2(\omega_i)$, of the power in $x(n)$ at frequency ω_i. Note that although these equations were derived for a specific frequency ω_i, since this frequency was arbitrary, then these equations are valid for all ω. Thus, the optimum filter for estimating the power in $x(n)$ at frequency ω is

$$\boxed{\mathbf{g} = \frac{\mathbf{R}_x^{-1}\mathbf{e}}{\mathbf{e}^H \mathbf{R}_x^{-1}\mathbf{e}}} \tag{8.89}$$

and the power estimate is

$$\boxed{\hat{\sigma}_x^2(\omega) = \frac{1}{\mathbf{e}^H \mathbf{R}_x^{-1}\mathbf{e}}} \tag{8.90}$$

where $\mathbf{e} = [1, e^{j\omega}, \ldots, e^{jp\omega}]^T$.

Example 8.3.1 *White Noise*

Let us consider using Eq. (8.90) to estimate the power in white noise that has a variance of σ_x^2. Since the autocorrelation matrix is $\mathbf{R}_x = \sigma_x^2 \mathbf{I}$, then the minimum variance bandpass filter is

$$\mathbf{g} = \frac{\mathbf{R}_x^{-1}\mathbf{e}}{\mathbf{e}^H \mathbf{R}_x^{-1}\mathbf{e}} = \frac{\sigma_x^{-2}\mathbf{e}}{\sigma_x^{-2}\mathbf{e}^H\mathbf{e}} = \frac{1}{p+1}\mathbf{e}$$

[7] The first equality follows from the fact that $\mathbf{g}_i^H \mathbf{e}_i$ is constrained to be equal to one and, therefore, is real-valued.

which is the same, to within a constant, as the filter in Eq. (8.11) that appears in the filter bank interpretation of the periodogram. From Eq. (8.90) it follows that the estimate of the power in $x(n)$ at frequency ω is

$$\hat{\sigma}_x^2(\omega) = \frac{1}{\mathbf{e}^H \mathbf{R}_x^{-1} \mathbf{e}} = \frac{1}{p+1}\sigma_x^2$$

which is independent of ω. Therefore, the distribution of power in $x(n)$ is a constant. In addition, note that the estimated power decreases as the filter order, p, increases. This is a result of the fact that, as the order of the bandpass filter increases, the bandwidth decreases and less power is allowed to pass through the filter. Since the power spectral density of white noise is constant, the total power over a frequency band of width $\Delta\omega$ is $\sigma_x^2 \Delta\omega$, which goes to zero as $\Delta\omega$ goes to zero. As we will see, what must be done in order to obtain an estimate of the power spectrum is to divide the power estimate by the bandwidth of the filter.

Having designed the bandpass filter bank and estimated the distribution of power in $x(n)$ as a function of frequency, we may now estimate the power spectrum by dividing the power estimate by the bandwidth of the bandpass filter. Although there are several different criteria that may be used to define bandwidth, the simplest is to use the value of Δ that produces the correct power spectral density for white noise. Since the minimum variance estimate of the power in white noise is $E\left\{|y_i(n)|^2\right\} = \sigma_x^2/(p+1)$, it follows from Eq. (8.82) that the spectrum estimate is

$$\hat{P}_x(e^{j\omega_i}) = \frac{E\left\{|y_i(n)|^2\right\}}{\Delta/2\pi} = \frac{\sigma_x^2}{p+1}\frac{2\pi}{\Delta} \tag{8.91}$$

Therefore, if we set

$$\Delta = \frac{2\pi}{p+1} \tag{8.92}$$

then $\hat{P}_x(e^{j\omega}) = \sigma_x^2$. Using Eq. (8.92) as the bandwidth of the filter $g(n)$, the power spectrum estimate becomes, in general,

$$\boxed{\hat{P}_{MV}(e^{j\omega}) = \frac{p+1}{\mathbf{e}^H \mathbf{R}_x^{-1} \mathbf{e}}} \tag{8.93}$$

which is the *minimum variance spectrum estimate*. Note that $\hat{P}_{MV}(e^{j\omega})$ is defined in terms of the autocorrelation matrix $\mathbf{R}_x$, of $x(n)$. If, as is normally the case, the autocorrelation matrix is unknown, then $\mathbf{R}_x$ may be replaced with an estimated autocorrelation matrix, $\hat{\mathbf{R}}_x$. A MATLAB program for computing the MV spectrum estimate in given in Fig. 8.19.

From a computational point of view, the minimum variance spectrum estimate requires the inversion of the autocorrelation matrix $\mathbf{R}_x$ (or $\hat{\mathbf{R}}_x$). Since $\mathbf{R}_x$ is Toeplitz, the inverse may be found using either the Levinson recursion or the Cholesky decomposition. Once the inverse has been found, the quadratic form $\mathbf{e}^H \mathbf{R}_x^{-1} \mathbf{e}$ must be evaluated, which may be done

The Minimum Variance Method

```
function Px = minvar(x,p)
%
   x = x(:);
   R = covar(x,p);
   [v,d]=eig(R);
   U = diag(inv(abs(d)+eps));
   V  = abs(fft(v,1024)).^2;
   Px = 10*log10(p)-10*log10(V*U);
   end;
```

Figure 8.19 *A* MATLAB *program for estimating the power spectrum using the minimum variance method.*

efficiently as follows. Let $v_x(k,l)$ denote the (k,l)th entry in the inverse of $\mathbf{R}_x$

$$\mathbf{R}_x^{-1} = \begin{bmatrix} v_x(0,0) & v_x(0,1) & \cdots & v_x(0,p) \\ v_x(1,0) & v_x(1,1) & \cdots & v_x(1,p) \\ \vdots & \vdots & & \vdots \\ v_x(p,0) & v_x(p,1) & \cdots & v_x(p,p) \end{bmatrix}$$

The quadratic form is

$$\mathbf{e}^H\mathbf{R}_x^{-1}\mathbf{e} = \sum_{k=0}^{p}\sum_{l=0}^{p} e^{-jk\omega}v_x(k,l)e^{jl\omega} = \sum_{k=0}^{p}\sum_{l=0}^{p} v_x(k,l)e^{j(l-k)\omega} \tag{8.94}$$

which may be written as

$$\mathbf{e}^H\mathbf{R}_x^{-1}\mathbf{e} = \sum_{n=-p}^{p}\left[\sum_{k=\max(0,n)}^{\min(p,p+n)} v_x(k,k-n)\right]e^{-jn\omega}$$

Note that the expression in brackets,

$$q(n) = \sum_{k=\max(0,n)}^{\min(p,p+n)} v_x(k,k-n)$$

is the sequence that is formed by summing the terms in the inverse matrix along the diagonals, i.e., $q(0)$ is the sum along the main diagonal, $q(1)$ is the sum along the first diagonal below the main diagonal, $q(-1)$ is the sum along the first diagonal above the main diagonal, and so on. Therefore,

$$\mathbf{e}^H\mathbf{R}_x^{-1}\mathbf{e} = \sum_{n=-p}^{p} q(n)e^{-jn\omega}$$

which is the discrete-time Fourier transform of the sequence $q(n)$. Since $\mathbf{R}_x^{-1}$ is Hermitian, then $q(n) = q^*(-n)$ and we also have

$$\mathbf{e}^H\mathbf{R}_x^{-1}\mathbf{e} = q(0) + 2\sum_{n=1}^{p}\mathrm{Re}\left\{q(n)e^{-jn\omega}\right\}$$

or, if $r_x(k)$ is real,

$$\mathbf{e}^H\mathbf{R}_x^{-1}\mathbf{e} = q(0) + 2\sum_{n=1}^{p} q(n)\cos(n\omega)$$

In the following example, we consider the minimum variance estimate of an AR(1) process.

Example 8.3.2 *The MV Estimate of an AR(1) Process*

Let $x(n)$ be an AR(1) random process with autocorrelation

$$r_x(k) = \frac{1}{1-\alpha^2}\alpha^{|k|}$$

where $|\alpha| < 1$. The power spectrum of $x(n)$ is

$$P_x(e^{j\omega}) = \frac{1}{|1-\alpha e^{-j\omega}|^2} = \frac{1}{1+\alpha^2-2\alpha\cos\omega}$$

Given $r_x(k)$ for $|k| \leq p$, the pth-order minimum variance spectrum estimate is

$$\hat{P}_{MV}(e^{j\omega}) = \frac{p+1}{\mathbf{e}^H\mathbf{R}_x^{-1}\mathbf{e}}$$

where $\mathbf{R}_x = \frac{1}{1-\alpha^2}\text{Toep}\{1, \alpha, \ldots, \alpha^p\}$. From Example 5.2.11 (p. 261) it follows that the inverse of $\mathbf{R}_x$ is

$$\mathbf{R}_x^{-1} = \begin{bmatrix} 1 & -\alpha & 0 & \cdots & 0 & 0 \\ -\alpha & 1+\alpha^2 & -\alpha & \cdots & 0 & 0 \\ 0 & -\alpha & 1+\alpha^2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1+\alpha^2 & -\alpha \\ 0 & 0 & 0 & \cdots & -\alpha & 1 \end{bmatrix}$$

Therefore,

$$q(0) = [2 + (p-1)(1+\alpha^2)]$$

and

$$q(1) = -\alpha p$$

so the minimum variance estimate is

$$\hat{P}_{MV}(e^{j\omega}) = \frac{(p+1)}{2+(p-1)(1+\alpha^2)-2\alpha p\cos\omega}$$

Note that $\hat{P}_{MV}(e^{j\omega})$ converges to $P_x(e^{j\omega})$ as $p \to \infty$.

In a number of applications it is important to be able to estimate the frequency of a sinusoid or a complex exponential in noise. The following example considers the use of the minimum variance method for frequency estimation.

Example 8.3.3 *The MV Estimate of a Complex Exponential in Noise*

Let $x(n)$ be a random phase complex exponential in white noise

$$x(n) = A_1 e^{jn\omega_1} + w(n)$$

where $A_1 = |A_1|e^{j\phi}$ with ϕ a random variable that is uniformly distributed over the interval $[-\pi, \pi]$. If the variance of $w(n)$ is σ_w^2, then the autocorrelation of $x(n)$ is

$$r_x(k) = P_1 e^{jk\omega_1} + \sigma_w^2 \delta(k)$$

where $P_1 = |A_1|^2$. Therefore, we may write the autocorrelation matrix as follows:

$$\mathbf{R}_x = P_1 \mathbf{e}_1 \mathbf{e}_1^H + \sigma_w^2 \mathbf{I}$$

where $\mathbf{e}_1 = \left[1, e^{j\omega_1}, \ldots, e^{jp\omega_1}\right]^T$. Using Woodbury's identity (see p. 29) it follows that the inverse of $\mathbf{R}_x$ is

$$\mathbf{R}_x^{-1} = \frac{1}{\sigma_w^2}\mathbf{I} - \frac{\frac{1}{\sigma_w^4} P_1 \mathbf{e}_1 \mathbf{e}_1^H}{1 + \frac{P_1}{\sigma_w^2}\mathbf{e}_1^H \mathbf{e}_1} = \frac{1}{\sigma_w^2}\left[\mathbf{I} - \frac{P_1}{\sigma_w^2 + (p+1)P_1}\mathbf{e}_1 \mathbf{e}_1^H\right]$$

(also see Example 2.3.9 on p. 45). Thus,

$$\begin{aligned} \hat{P}_{MV}(e^{j\omega}) &= \frac{p+1}{\mathbf{e}^H \mathbf{R}_x^{-1} \mathbf{e}} = \frac{p+1}{\frac{1}{\sigma_w^2}\mathbf{e}^H\left[\mathbf{I} - \frac{P_1}{\sigma_w^2 + (p+1)P_1}\mathbf{e}_1 \mathbf{e}_1^H\right]\mathbf{e}} \\ &= \frac{\sigma_w^2}{1 - \frac{P_1/(p+1)}{\sigma_w^2 + (p+1)P_1}|\mathbf{e}^H \mathbf{e}_1|^2} \end{aligned} \tag{8.95}$$

Since

$$\mathbf{e}^H \mathbf{e}_1 = \sum_{k=0}^{p} e^{-jk\omega} e^{jk\omega_1} = \sum_{k=0}^{p} e^{-jk(\omega-\omega_1)} = W_R(e^{j(\omega-\omega_1)})$$

where $W_R(e^{j\omega})$ is the discrete-time Fourier transform of a rectangular window that extends from $k = 0$ to $k = p$, then the minimum variance estimate is

$$\hat{P}_{MV}(e^{j\omega}) = \frac{\sigma_w^2}{1 - \frac{P_1/(p+1)}{\sigma_w^2 + (p+1)P_1}\left|W_R(e^{j(\omega-\omega_1)})\right|^2}$$

From this expression we see that the minimum variance estimate attains its maximum at $\omega = \omega_1$ with

$$\hat{P}_{MV}(e^{j\omega})\Big|_{\omega=\omega_1} = \frac{\sigma_w^2}{1 - \frac{(p+1)P_1}{\sigma_w^2 + (p+1)P_1}} = \sigma_w^2 + (p+1)P_1$$

Therefore, the estimate of the *power* in $x(n)$ at frequency $\omega = \omega_1$ is

$$\hat{\sigma}_x^2(\omega_1) = \frac{1}{p+1}\hat{P}_{MV}(e^{j\omega_1}) = \frac{\sigma_w^2}{p+1} + P_1$$

Thus, if the signal-to-noise ratio is large, $P_1 \gg \sigma_w^2$, then the minimum variance estimate of the power at $\omega = \omega_1$ becomes

$$\hat{\sigma}_x^2(\omega_1) = P_1$$

Furthermore, if $p \gg 1$ and $\omega \neq \omega_1$, then $W_R(e^{j(\omega-\omega_1)}) \approx 0$ and the minimum variance estimate of the power spectrum becomes

$$\hat{P}_{MV}(e^{j\omega}) \approx \sigma_w^2$$

One issue that we have not yet addressed concerns the question of how to select the filter order, p. Clearly, the larger the order of the filter the better the filter will be in rejecting out-of-band power. Therefore, with this in mind, the order of the filter should be as large as possible. From a practical point of view, however, there is a limit on how large the filter order may be. Specifically, note that for a pth-order filter the MV spectrum estimate requires the evaluation of the inverse of a $(p+1) \times (p+1)$ autocorrelation matrix $\mathbf{R}_x$. However, in order to be able to invert this matrix, it is necessary that $r_x(k)$ be known or estimated for $k = 0, 1, \ldots, p$. Therefore, for a fixed data record length, N, since it is only possible to estimate $r_x(k)$ for $k = 0, 1, \ldots, N-1$, the filter order is limited to $p \leq N$. In practice, however, the filter order will generally be much smaller than this due to the large variance of the autocorrelation estimates for values of k that are close to N.

In addition to the approach described above, there are other forms of minimum variance spectrum estimates that differ either in the way that the filter is designed or in the way that the power in the output signal is computed [12,31,54]. For example, in the Data Adaptive Spectrum Estimation (DASE) algorithm [12], $G_i(e^{j\omega})$ is constrained to have unit energy within a passband of width Δ that is centered at frequency ω_i,

$$\frac{1}{\Delta}\int_{\omega_i-\Delta/2}^{\omega_i+\Delta/2} |G_i(e^{j\omega})|^2 \, d\omega = 1$$

Solving the constrained minimization problem in this case leads to a generalized eigenvalue problem.

8.4 THE MAXIMUM ENTROPY METHOD

One of the limitations with the classical approach to spectrum estimation is that, for a data record of length N, the autocorrelation sequence can only be estimated for lags $|k| < N$. As a result, $\hat{r}_x(k)$ is set to zero for $|k| \geq N$. Since many signals of interest have autocorrelations that are nonzero for $|k| \geq N$, this windowing may significantly limit the resolution and accuracy of the estimated spectrum. This is particularly true, for example, in the case of narrowband processes that have autocorrelations that decay slowly with k. Since the classical methods effectively extrapolate the autocorrelation sequence with zeros, if it were possible to perform a more accurate extrapolation of the autocorrelation sequence, then the effects of the window could be mitigated and a more accurate estimate of the spectrum could be found. A difficult question to answer, however, is the following: How should this extrapolation be performed? The maximum entropy method (MEM) that is developed in this section suggests one possible way to perform this extrapolation.

Given the autocorrelation $r_x(k)$ of a WSS process for lags $|k| \leq p$, the problem that we wish to address, illustrated in Fig. 8.20, is how to extrapolate $r_x(k)$ for $|k| > p$. Denoting the extrapolated values by $r_e(k)$, it is clear that some constraints should be placed on $r_e(k)$. For example, if

$$P_x(e^{j\omega}) = \sum_{k=-p}^{p} r_x(k)e^{-jk\omega} + \sum_{|k|>p} r_e(k)e^{-jk\omega} \tag{8.96}$$

then $P_x(e^{j\omega})$ should correspond to a valid power spectrum, i.e., $P_x(e^{j\omega})$ should be real-valued and nonnegative for all ω. In general, however, only constraining $P_x(e^{j\omega})$ to be real and nonnegative is not sufficient to guarantee a unique extrapolation. Therefore, some additional constraints must be imposed on the set of allowable extrapolations.[8] One such constraint, proposed by Burg [7], is to perform the extrapolation in such a way so as to maximize the entropy of the process.[9] Since entropy is a measure of randomness or uncertainty, a maximum entropy extrapolation is equivalent to finding the sequence of autocorrelations, $r_e(k)$, that make $x(n)$ as *white* (random) as possible. In some sense, such an extrapolation places as few constraints as possible or the least amount of structure on $x(n)$. In terms of the power spectrum, this corresponds to the constraint that $P_x(e^{j\omega})$ be "as flat as possible" (see Fig. 8.21).

For a Gaussian random process with power spectrum $P_x(e^{j\omega})$, the entropy is [11]

$$H(x) = \frac{1}{2\pi}\int_{-\pi}^{\pi} \ln P_x(e^{j\omega})d\omega \tag{8.97}$$

Therefore, for Gaussian processes with a given partial autocorrelation sequence, $r_x(k)$ for $|k| \leq p$, the maximum entropy power spectrum is the one that maximizes Eq. (8.97) subject to the constraint that the inverse discrete-time Fourier transform of $P_x(e^{j\omega})$ equals the given set of autocorrelations for $|k| \leq p$,

$$\frac{1}{2\pi}\int_{-\pi}^{\pi} P_x(e^{j\omega})e^{jk\omega}d\omega = r_x(k) \quad ; \quad |k| \leq p \tag{8.98}$$

The values of $r_e(k)$ that maximize the entropy may be found by setting the derivative of $H(x)$ with respect to $r_e^*(k)$ equal to zero as follows

$$\frac{\partial H(x)}{\partial r_e^*(k)} = \frac{1}{2\pi}\int_{-\pi}^{\pi} \frac{1}{P_x(e^{j\omega})}\frac{\partial P_x(e^{j\omega})}{\partial r_e^*(k)}d\omega = 0 \quad ; \quad |k| > p \tag{8.99}$$

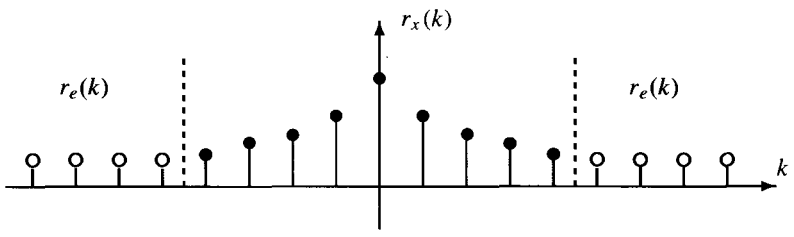


Figure 8.20 *Extrapolating the autocorrelation sequence.*

[8] It is assumed that the given partial autocorrelation sequence is *extendible* so that there is at least one extrapolation that produces a valid power spectrum. See Section 5.2.8 for a discussion of the conditions under which an autocorrelation sequence is extendible.

[9] Entropy has its origins in information theory and is a subject that is well outside the scope of this book. For a treatment of entropy the reader may consult [4, 11, 40].

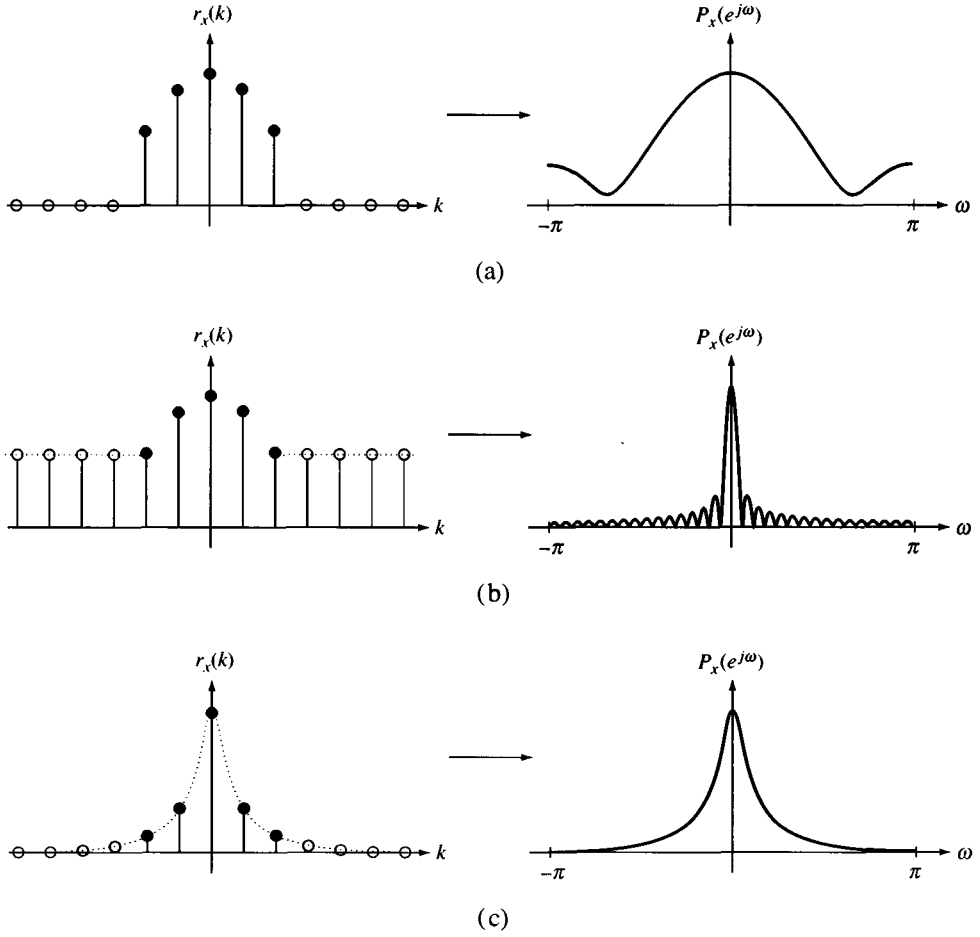


Figure 8.21 *Different extrapolations of a partial autocorrelation sequence and the corresponding power spectral densities. The problem is to find the extrapolation that produces a spectrum that is as flat as possible.*

From Eq. (8.96) we see that[10]

$$\frac{\partial P_x(e^{j\omega})}{\partial r_e^*(k)} = e^{jk\omega}$$

which, when substituted into Eq. (8.99), yields

$$\frac{1}{2\pi}\int_{-\pi}^{\pi} \frac{1}{P_x(e^{j\omega})} e^{jk\omega} d\omega = 0 \quad ; \quad |k| > p \tag{8.100}$$

Defining $Q_x(e^{j\omega}) = 1/P_x(e^{j\omega})$, Eq. (8.100) states that the inverse discrete-time Fourier transform of $Q_x(e^{j\omega})$ is a finite-length sequence that is equal to zero for $|k| > p$,

$$q_x(k) = \frac{1}{2\pi}\int_{-\pi}^{\pi} Q_x(e^{j\omega}) e^{jk\omega} d\omega = 0 \quad ; \quad |k| > p \tag{8.101}$$

[10]Recall that $r_x(k)$ is conjugate symmetric so $r_x(-k) = r_x^*(k)$.

Therefore,

$$Q_x(e^{j\omega}) = \frac{1}{P_x(e^{j\omega})} = \sum_{k=-p}^{p} q_x(k)e^{-jk\omega}$$

and it follows that the maximum entropy power spectrum for a Gaussian process, which we will denote by $\hat{P}_{mem}(e^{j\omega})$, is an all-pole power spectrum,

$$\hat{P}_{mem}(e^{j\omega}) = \frac{1}{\sum_{k=-p}^{p} q_x(k)e^{-jk\omega}} \tag{8.102}$$

Using the spectral factorization theorem, it follows that Eq. (8.102) may be expressed as

$$\hat{P}_{mem}(e^{j\omega}) = \frac{|b(0)|^2}{A_p(e^{j\omega})A_p^*(e^{j\omega})} = \frac{|b(0)|^2}{\left|1 + \sum_{k=1}^{p} a_p(k)e^{-jk\omega}\right|^2}$$

Alternatively, in terms of the vectors $\mathbf{a}_p = [1, a_p(1), \ldots, a_p(p)]^T$ and $\mathbf{e} = [1, e^{j\omega}, \ldots, e^{jp\omega}]^T$, the MEM spectrum may be written as

$$\hat{P}_{mem}(e^{j\omega}) = \frac{|b(0)|^2}{|\mathbf{e}^H\mathbf{a}_p|^2} \tag{8.103}$$

Having determined the form of the MEM spectrum, all that remains is to find the coefficients $a_p(k)$ and $b(0)$. Due to the constraint given in Eq. (8.98), these coefficients must be chosen in such a way that the inverse discrete-time Fourier transform of $\hat{P}_{mem}(e^{j\omega})$ produces an autocorrelation sequence that matches the given values of $r_x(k)$ for $|k| \leq p$. As we saw in Section 5.2.3, if the coefficients $a_p(k)$ are the solution to the autocorrelation normal equations

$$\begin{bmatrix} r_x(0) & r_x^*(1) & \cdots & r_x^*(p) \\ r_x(1) & r_x(0) & \cdots & r_x^*(p-1) \\ \vdots & \vdots & & \vdots \\ r_x(p) & r_x(p-1) & \cdots & r_x(0) \end{bmatrix} \begin{bmatrix} 1 \\ a_p(1) \\ \vdots \\ a_p(p) \end{bmatrix} = \epsilon_p \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{8.104}$$

and if

$$|b(0)|^2 = r_x(0) + \sum_{k=1}^{p} a_p(k)r_x^*(k) = \epsilon_p \tag{8.105}$$

then the autocorrelation matching constraint given in Eq. (8.98) will be satisfied. Thus, the MEM spectrum is

$$\boxed{\hat{P}_{mem}(e^{j\omega}) = \frac{\epsilon_p}{|\mathbf{e}^H\mathbf{a}_p|^2}} \tag{8.106}$$

where $\mathbf{a}_p$ is the solution to Eq. (8.104). In summary, given a sequence of autocorrelations, $r_x(k)$ for $k = 0, 1, \ldots, p$, the MEM spectrum is computed as follows. First, the autocorrelation normal equations (8.104) are solved for the all-pole coefficients $a_p(k)$ and ϵ_p. Then, the MEM spectrum is formed by incorporating these parameters into Eq. (8.106). Note that since $\hat{P}_{mem}(e^{j\omega})$ is an all-pole power spectrum, then $r_x(k)$ satisfies the Yule-Walker

The Maximum Entropy Method

```
function Px = mem(x,p)
%
   [a,e] = acm(x,p);
   Px    = 20*(log10(e)-log10(abs(fft(a,1024))));
   end;
```

Figure 8.22 *A* MATLAB *program for estimating the power spectrum using the maximum entropy method. Note that this m-file calls* `acm.m`.

equations

$$r_x(l) = -\sum_{k=1}^{p} a_p(k) r_x(k-l) \quad \text{for } l > 0 \tag{8.107}$$

Therefore, the maximum entropy method extrapolates the autocorrelation sequence according to this recursion. A MATLAB program for computing the MEM spectrum is given in Fig. 8.22.

The properties of the maximum entropy method have been studied extensively and, as a spectrum analysis tool, the maximum entropy method is subject to different interpretations [10,29]. It may be argued, for example, that in the absence of any information or constraints on a process $x(n)$, given a set of autocorrelation values, $r_x(0), \ldots, r_x(p)$, the best way to estimate the power spectrum is to Fourier Transform the autocorrelation sequence formed from the given values along with an extrapolation that imposes the least amount of structure on the data, i.e., perform a maximum entropy extrapolation. This would seem to be preferable to an extrapolation that somewhat arbitrarily sets $r_x(k) = 0$ for $|k| > p$ as in the classical approach. On the other hand, it may also be argued that since the maximum entropy extrapolation imposes an all-pole model on the data, unless the process is known to be consistent with this model, then the estimated spectrum may not be very accurate. Whether or not the MEM estimate is "better" than the minimum variance method or an estimate produced using a classical approach depends critically on what type of process is being analyzed and on how closely the process may be modeled as an autoregressive process.

Example 8.4.1 *The MEM Estimate of a Complex Exponential in Noise*

In Example 8.3.3, we found the minimum variance estimate of a complex exponential in white noise

$$x(n) = A_1 e^{jn\omega_1} + w(n)$$

where $A_1 = |A_1| e^{j\phi}$ with ϕ being a random variable that is uniformly distributed over the interval $[-\pi, \pi]$ and with the variance of $w(n)$ being equal to σ_w^2. To find the pth-order MEM spectrum, we must solve the autocorrelation normal equations (8.104) for the AR coefficients $a_p(k)$. The $(p+1) \times (p+1)$ autocorrelation matrix for $x(n)$ is

$$\mathbf{R}_x = P_1 \mathbf{e}_1 \mathbf{e}_1^H + \sigma_w^2 \mathbf{I}$$

where $\mathbf{e}_1 = [1, e^{j\omega_1}, \ldots, e^{jp\omega_1}]^T$ and $P_1 = |A_1|^2$. As we saw in Example 8.3.3, the inverse

of $\mathbf{R}_x$ is

$$\mathbf{R}_x^{-1} = \frac{1}{\sigma_w^2}\left[\mathbf{I} - \frac{P_1}{\sigma_w^2 + (p+1)P_1}\mathbf{e}_1\mathbf{e}_1^H\right]$$

Therefore,

$$\mathbf{a}_p = \epsilon_p \mathbf{R}_x^{-1}\mathbf{u}_1 = \frac{\epsilon_p}{\sigma_w^2}\left[\mathbf{u}_1 - \frac{P_1}{\sigma_w^2 + (p+1)P_1}\mathbf{e}_1\right] \tag{8.108}$$

where $\mathbf{u}_1 = [1, 0, \ldots, 0]^T$, and the MEM spectrum is

$$\hat{P}_{mem}(e^{j\omega}) = \frac{\epsilon_p}{|\mathbf{e}^H\mathbf{a}_p|^2} = \frac{\epsilon_p}{\left(\frac{\epsilon_p}{\sigma_w^2}\right)^2\left|\mathbf{e}^H\left(\mathbf{u}_1 - \frac{P_1}{\sigma_w^2 + (p+1)P_1}\mathbf{e}_1\right)\right|^2}$$

$$= \frac{\epsilon_p}{\left(\frac{\epsilon_p}{\sigma_w^2}\right)^2\left|1 - \frac{P_1}{\sigma_w^2 + (p+1)P_1}\mathbf{e}^H\mathbf{e}_1\right|^2}$$

Since

$$\mathbf{e}^H\mathbf{e}_1 = \sum_{k=0}^{p} e^{-jk(\omega-\omega_1)} = W_R(e^{j(\omega-\omega_1)})$$

where $W_R(e^{j\omega})$ is the discrete-time Fourier transform of a rectangular window, then the MEM spectrum becomes

$$\hat{P}_{mem}(e^{j\omega}) = \frac{\epsilon_p}{\left(\frac{\epsilon_p}{\sigma_w^2}\right)^2\left|1 - \frac{P_1}{\sigma_w^2 + (p+1)P_1}W_R(e^{j(\omega-\omega_1)})\right|^2}$$

To determine the value of ϵ_p, note that since $a_p(0) = 1$, then it follows from Eq. (8.108) that

$$a_p(0) = \frac{\epsilon_p}{\sigma_w^2}\left[1 - \frac{P_1}{\sigma_w^2 + (p+1)P_1}\right] = 1$$

Solving for ϵ_p we have

$$\epsilon_p = \sigma_w^2\left[1 + \frac{P_1}{\sigma_w^2 + pP_1}\right]$$

and the MEM spectrum becomes

$$\hat{P}_{mem}(e^{j\omega}) = \frac{\sigma_w^2\left[1 - \frac{P_1}{\sigma_w^2 + (p+1)P_1}\right]}{\left|1 - \frac{P_1}{\sigma_w^2 + (p+1)P_1}W_R(e^{j(\omega-\omega_1)})\right|^2}$$

As with the minimum variance method, the peak of the MEM spectrum occurs at frequency $\omega = \omega_1$, and

$$\hat{P}_{mem}(e^{j\omega})\Big|_{\omega=\omega_1} = \frac{\sigma_w^2\left[1 - \frac{P_1}{\sigma_w^2 + (p+1)P_1}\right]}{\left|1 - \frac{(p+1)P_1}{\sigma_w^2 + (p+1)P_1}\right|^2} = \frac{1}{\sigma_w^2}[\sigma_w^2 + pP_1][\sigma_w^2 + (p+1)P_1]$$

If the signal-to-noise ratio is large, $P_1 \gg \sigma_w^2$, then the MEM spectrum estimate at $\omega = \omega_1$ is approximately

$$\hat{P}_{mem}(e^{j\omega})\Big|_{\omega=\omega_1} \approx p^2 \frac{P_1^2}{\sigma_w^2}$$

Thus, the peak in the MEM spectrum is proportional to the *square* of the power in the complex exponential.

There is an interesting relationship that exists between the MEM and MV spectrum estimates. This relationship states that the pth-order MV estimate is the harmonic mean of the MEM estimates of orders $k = 0, 1, \ldots, p$. To derive this relationship, we will use the recursion given in Eq. (5.114) for computing the inverse of a Toeplitz matrix. This recursion provides the following expression for the inverse of the $(p+1) \times (p+1)$ Toeplitz matrix $\mathbf{R}_p$,

$$\mathbf{R}_p^{-1} = \left[\begin{array}{c|ccc} 0 & 0 & \cdots & 0 \\ \hline 0 & & & \\ \vdots & & \mathbf{R}_{p-1}^{-1} & \\ 0 & & & \end{array}\right] + \frac{1}{\epsilon_p}\mathbf{a}_p\mathbf{a}_p^H \tag{8.109}$$

For a WSS process with an autocorrelation matrix $\mathbf{R}_p$, the pth-order MV estimate is

$$\hat{P}_{MV}(e^{j\omega}) = \frac{p+1}{\mathbf{e}^H\mathbf{R}_p^{-1}\mathbf{e}} \tag{8.110}$$

Multiplying $\mathbf{R}_p^{-1}$ on the left by $\mathbf{e}^H$ and on the right by $\mathbf{e}$ we have

$$\mathbf{e}^H\mathbf{R}_p^{-1}\mathbf{e} = \mathbf{e}^H \left[\begin{array}{c|ccc} 0 & 0 & \cdots & 0 \\ \hline 0 & & & \\ \vdots & & \mathbf{R}_{p-1}^{-1} & \\ 0 & & & \end{array}\right] \mathbf{e} + \frac{1}{\epsilon_p}\mathbf{e}^H\mathbf{a}_p\mathbf{a}_p^H\mathbf{e} \tag{8.111}$$

or,[11]

$$\mathbf{e}^H\mathbf{R}_p^{-1}\mathbf{e} = \mathbf{e}^H\mathbf{R}_{p-1}^{-1}\mathbf{e} + \frac{1}{\epsilon_p}|\mathbf{e}^H\mathbf{a}_p|^2 \tag{8.112}$$

Note that the first two terms in Eq. (8.112) are proportional to the inverse of the minimum variance estimates of order p and $p-1$, respectively, and the last term is the reciprocal of the pth-order MEM spectrum. Therefore, we have

$$\frac{p+1}{\hat{P}_{MV}^{(p)}(e^{j\omega})} = \frac{p}{\hat{P}_{MV}^{(p-1)}(e^{j\omega})} + \frac{1}{\hat{P}_{mem}^{(p)}(e^{j\omega})}$$

which is a recursion for the pth-order MV estimate in terms of the $(p-1)$st-order MV estimate and the pth-order MEM spectrum. Solving this recursion for $\hat{P}_{MV}^{(p)}(e^{j\omega})$ we find

$$\boxed{\frac{1}{\hat{P}_{MV}^{(p)}(e^{j\omega})} = \frac{1}{p+1}\sum_{k=0}^{p}\frac{1}{\hat{P}_{mem}^{(k)}(e^{j\omega})}} \tag{8.113}$$

[11]Note that, in order to keep the notation as simple as possible, we are using $\mathbf{e}$ to denote a vector of complex exponentials $e^{jk\omega}$ of varying lengths. Thus, the number of elements in $\mathbf{e}$ is determined by the size of the vector or matrix that it is multiplied by.

Therefore, the MV estimate is the harmonic mean of the MEM spectra from the low order to the high order estimates. As a result of this smoothing, for WSS processes consisting of narrowband components in noise, the MEM spectrum generally provides a higher resolution spectrum estimate than the minimum variance method.

8.5 PARAMETRIC METHODS

One of the limitations of the nonparametric methods of spectrum estimation presented in Sections 8.2 and 8.3 is that they are not designed to incorporate information that may be available about the process into the estimation procedure. In some applications this may be an important limitation, particularly when some knowledge is available about how the data samples are generated. In speech processing, for example, an acoustic tube model for the vocal tract imposes an autoregressive model on the speech waveform [43]. As a result, for intervals of time over which the speech waveform may be assumed to be approximately stationary, the spectrum is of the form,

$$P_x(e^{j\omega}) = \frac{|b(0)|^2}{\left|1 + \sum_{k=1}^{p} a_p(k)e^{-jk\omega}\right|^2}$$

However, with a nonparametric approach such as the periodogram, the power spectrum has a form that is consistent with a moving average process,

$$\hat{P}_{per}(e^{j\omega}) = |X_N(e^{j\omega})|^2 = \sum_{k=-N}^{N} \hat{r}_x(k)e^{-jk\omega}$$

Therefore, one would hope that if it were possible to incorporate a model for the process directly into the spectrum estimation algorithm, then a more accurate and higher resolution estimate could be found. This may be easily done using a parametric approach to spectrum estimation. With a parametric approach, the first step is to select an appropriate model for the process. This selection may be based on a priori knowledge about how the process is generated or, perhaps, on experimental results indicating that a particular model "works well." Models that are commonly used include autoregressive (AR), moving average (MA), autoregressive moving average (ARMA), and harmonic (complex exponentials in noise). Once a model has been selected, the next step is to estimate the model parameters from the given data. The final step is to estimate the power spectrum by incorporating the estimated parameters into the parametric form for the spectrum. For example, with an autoregressive moving average model, with estimates $\hat{b}_q(k)$ and $\hat{a}_p(k)$ of the model parameters, the spectrum estimate would be

$$\hat{P}_x(e^{j\omega}) = \frac{\left|\sum_{k=0}^{q} \hat{b}_q(k)e^{-jk\omega}\right|^2}{\left|1 + \sum_{k=1}^{p} \hat{a}_p(k)e^{-jk\omega}\right|^2}$$

Although it is possible to significantly improve the resolution of the spectrum estimate with a parametric method, it is important to realize that, unless the model that is used

is appropriate for the process that is being analyzed, inaccurate or misleading estimates may be obtained. Consider, for example, the two estimates shown in Fig. 8.23*a* that were formed from $N = 64$ values of a process consisting of two sinusoids in unit variance white noise

$$x(n) = 5\sin(0.45\pi n + \phi_1) + 5\sin(0.55\pi n + \phi_2) + w(n)$$

The spectrum estimate shown by the solid line was derived assuming an all-pole model for $x(n)$ whereas the estimate shown by the dashed line was formed using the Blackman-Tukey method. Clearly, the estimate produced with the model-based approach provides much better resolution than the Blackman-Tukey method. Fig. 8.23*b*, on the other hand, shows the spectrum estimates that are formed using the same two approaches for the MA(2) process

$$x(n) = w(n) - w(n-2)$$

where $w(n)$ is unit variance white noise. The power spectrum of $x(n)$ is

$$P_x(e^{j\omega}) = 2 - 2\cos 2\omega$$

In this case, the model-based approach inaccurately represents the underlying spectrum, indicating that the process may contain two narrowband components. The Blackman-Tukey method, on the other hand, which makes no assumptions about the process, produces a more accurate estimate of the power spectrum.

In this section, we consider power spectrum estimation using AR, MA, and ARMA models. The problem of frequency estimation for processes consisting of complex exponentials in noise will be considered in Section 8.6.

8.5.1 Autoregressive Spectrum Estimation

An autoregressive process, $x(n)$, may be represented as the output of an all-pole filter that is driven by unit variance white noise. The power spectrum of a pth-order autoregressive

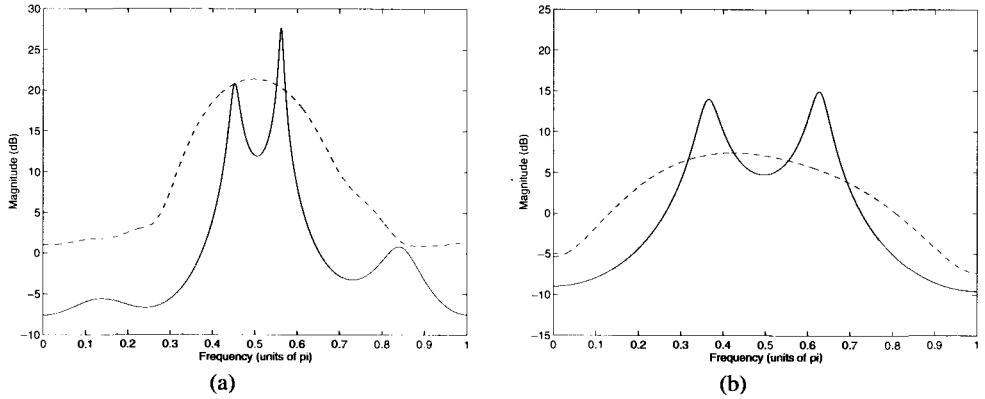


Figure 8.23 *Illustration of how an inappropriate model may lead to an inaccurate spectrum estimate. Using a spectrum estimation technique that assumes an all-pole model for the process (solid line) and the Blackman-Tukey method (dashed line), the spectrum estimates are shown for (a) a process consisting of two sinusoids in noise and (b) a second-order moving average process.*

process is

$$P_x(e^{j\omega}) = \frac{|b(0)|^2}{\left|1 + \sum_{k=1}^{p} a_p(k)e^{-jk\omega}\right|^2} \tag{8.114}$$

Therefore, if $b(0)$ and $a_p(k)$ can be estimated from the data, then an estimate of the power spectrum may be formed using

$$\hat{P}_{AR}(e^{j\omega}) = \frac{|\hat{b}(0)|^2}{\left|1 + \sum_{k=1}^{p} \hat{a}_p(k)e^{-jk\omega}\right|^2} \tag{8.115}$$

Clearly, the accuracy of $\hat{P}_{AR}(e^{j\omega})$ will depend on how accurately the model parameters may be estimated and, even more importantly, on whether or not an autoregressive model is consistent with the way in which the data is generated. If, for example, Eq. (8.115) is applied to a moving average process, then one should expect the estimate to be poor.

Since autoregressive spectrum estimation requires that an all-pole model be found for the process, there is a variety of techniques that may be used to estimate the all-pole parameters. However, once the all-pole parameters have been estimated, each method generates an estimate of the power spectrum in exactly the same way, i.e., using Eq. (8.115). In the following subsections, we briefly review the AR modeling techniques and describe some of the properties of these techniques as they apply to spectrum estimation.

The Autocorrelation Method. In the autocorrelation method of all-pole modeling, the AR coefficients $a_p(k)$ are found by solving the autocorrelation normal equations

$$\begin{bmatrix} r_x(0) & r_x^*(1) & r_x^*(2) & \cdots & r_x^*(p) \\ r_x(1) & r_x(0) & r_x^*(1) & \cdots & r_x^*(p-1) \\ r_x(2) & r_x(1) & r_x(0) & \cdots & r_x^*(p-2) \\ \vdots & \vdots & \vdots & & \vdots \\ r_x(p) & r_x(p-1) & r_x(p-2) & \cdots & r_x(0) \end{bmatrix} \begin{bmatrix} 1 \\ a_p(1) \\ a_p(2) \\ \vdots \\ a_p(p) \end{bmatrix} = \epsilon_p \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{8.116}$$

where

$$r_x(k) = \frac{1}{N} \sum_{n=0}^{N-1-k} x(n+k)x^*(n) \quad ; \quad k = 0, 1, \ldots, p \tag{8.117}$$

(for simplicity we are suppressing the hats over the autocorrelations $r_x(k)$ and all-pole parameters $a_p(k)$). Solving Eq. (8.116) for the coefficients $a_p(k)$, setting

$$|b(0)|^2 = \epsilon_p = r_x(0) + \sum_{k=1}^{p} a_p(k)r_x^*(k) \tag{8.118}$$

and incorporating these parameters into Eq. (8.115) produces an estimate of the power spectrum, which is sometimes referred to as the *Yule-Walker method*.[12] Note that the Yule-Walker method is equivalent to the maximum entropy method. In fact, the only difference between the two methods lies in the assumptions that are made about the process $x(n)$.

[12]This name comes from the fact that the autocorrelation normal equations are equivalent to the Yule-Walker equations for autoregressive processes (see Section 3.6.2).

Specifically, with the Yule-Walker method it is assumed that $x(n)$ is an autoregressive process, whereas with the maximum entropy method it is assumed that $x(n)$ is Gaussian.

Since the autocorrelation matrix $\mathbf{R}_x$ in the autocorrelation normal equations is Toeplitz, the Levinson-Durbin recursion may be used to solve these equations for $a_p(k)$. Furthermore, if $\mathbf{R}_x > 0$, then the roots of $A_p(z)$ will lie inside the unit circle. However, because the autocorrelation method effectively applies a rectangular window to the data when estimating the autocorrelation sequence using Eq. (8.117), the data is effectively extrapolated with zeros. Therefore, the autocorrelation method generally produces a lower resolution estimate than the approaches that do not window the data, such as the covariance method and Burg's method. Consequently, for short data records the autocorrelation method is not generally used.

An artifact that may be observed with the autocorrelation method is *spectral line splitting*. This artifact involves the splitting of a single spectral peak into two separate and distinct peaks. Typically, spectral line splitting occurs when $x(n)$ is *overmodeled*, i.e., when p is too large. An example of spectral line splitting is shown in Fig. 8.24 for an AR(2) process that is generated according to the difference equation

$$x(n) = -0.9x(n-2) + w(n)$$

where $w(n)$ is unit variance white noise. Using model orders of $p = 4$ and $p = 12$ and a data record of length $N = 64$, spectrum estimates are shown for the autocorrelation method. What we observe is that, although the true spectrum has a single spectral peak at $\omega = \pi/2$, when $p = 12$ this peak is split into two peaks.[13]

Since the autocorrelation estimate in Eq. (8.117) is biased, a variation of the autocorrelation method is to use the unbiased estimate

$$\hat{r}_x(k) = \frac{1}{N-k} \sum_{n=0}^{N-1-k} x(n+k)x^*(n) \quad ; \quad k = 0, 1, \ldots, p \tag{8.119}$$

In this case, however, the autocorrelation matrix is not guaranteed to be positive definite and, as a result, the variance of the spectrum estimate tends to become large when $\hat{\mathbf{R}}_x$

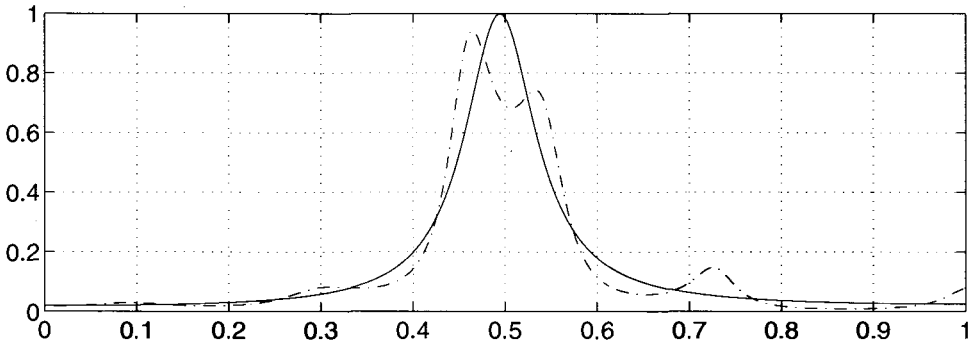


Figure 8.24 *Spectral line splitting of an AR(2) process. Two all-pole spectrum estimates were computed using the autocorrelation method with orders $p = 4$ (solid line) and $p = 12$ (dash-dot line).*

[13]Note that for a process such as this, spectral line splitting will not *always* be observed. Whether or not line splitting occurs depends on the specific white noise process that generates $x(n)$.

is ill-conditioned or singular [26,37]. Therefore, the biased estimate of $r_x(k)$ is generally preferred over the unbiased estimate.

The Covariance Method. Another approach for estimating the AR parameters is the covariance method. The covariance method requires finding the solution to the set of linear equations,

$$\begin{bmatrix} r_x(1,1) & r_x(2,1) & \cdots & r_x(p,1) \\ r_x(1,2) & r_x(2,2) & \cdots & r_x(p,2) \\ \vdots & \vdots & & \vdots \\ r_x(1,p) & r_x(2,p) & \cdots & r_x(p,p) \end{bmatrix} \begin{bmatrix} a_p(1) \\ a_p(2) \\ \vdots \\ a_p(p) \end{bmatrix} = - \begin{bmatrix} r_x(0,1) \\ r_x(0,2) \\ \vdots \\ r_x(0,p) \end{bmatrix} \tag{8.120}$$

where

$$r_x(k,l) = \sum_{n=p}^{N-1} x(n-l)x^*(n-k) \tag{8.121}$$

Unlike the linear equations in the autocorrelation method, these equations are not Toeplitz. However, the advantage of the covariance method over the autocorrelation method is that no windowing of the data is required in the formation of the autocorrelation estimates, $r_x(k,l)$. Therefore, for short data records the covariance method generally produces higher resolution spectrum estimates than the autocorrelation method. However, as the data record length increases and becomes large compared to the model order, $N \gg p$, the effect of the data window becomes small and the difference between the two approaches becomes negligible.

The Modified Covariance Method. The modified covariance method is similar to the covariance method in that no window is applied to the data. However, instead of finding the autoregressive model that minimizes the sum of the squares of the forward prediction error, the modified covariance method minimizes the sum of the squares of the forward and backward prediction errors.[14] As a result, the autoregressive parameters in the modified covariance method are found by solving a set of linear equations of the form given in Eq. (8.120) with

$$r_x(k,l) = \sum_{n=p}^{N-1} \Big[x(n-l)x^*(n-k) + x(n-p+l)x^*(n-p+k) \Big]$$

replacing the estimate in Eq. (8.121). As with the covariance method, the autocorrelation matrix is not Toeplitz.

In contrast to other AR spectrum estimation techniques, the modified covariance method appears to give statistically stable spectrum estimates with high resolution [37,51]. Furthermore, in the spectral analysis of sinusoids in white noise, although the modified covariance method is characterized by a shifting of the peaks from their true locations due to additive noise, this shifting appears to be less pronounced than with other autoregressive estimation techniques [53]. In addition, the peak locations tend to be less sensitive to phase [9,57]. Finally, unlike the previous methods, it appears that the modified covariance method is not subject to spectral line splitting [27].

[14] The modified covariance method has also been referred to as the *Forward-Backward Method* [37] and as the *Least Squares Method* [57].

The Burg Algorithm. As with the modified covariance algorithm, the Burg algorithm finds a set of all-pole model parameters that minimizes the sum of the squares of the forward and backward prediction errors. However, in order to assure that the model is stable, this minimization is performed sequentially with respect to the reflection coefficients. Although less accurate than the modified covariance method, since the Burg algorithm does not apply a window to the data, the estimates of the autoregressive parameters are more accurate than those obtained with the autocorrelation method. In the analysis of sinusoids in noise, the Burg algorithm is subject to spectral line splitting and the peak locations are highly dependent upon the phases of the sinusoids [27,35].

Examples. It is difficult to provide a complete set of examples to illustrate the properties of the AR spectrum estimation techniques for different types of processes and to compare them to other spectrum estimation algorithms. It would be interesting, for example, to compare the effectiveness of each approach in estimating the spectra of narrowband and wideband autoregressive processes of various orders using different data record lengths. It would also be interesting to look at what happens when these techniques are applied to other types of processes such as MA processes, ARMA processes, and harmonic processes. Rather than attempting to cover all of the possibilities, in the following we only consider the use of AR spectrum estimation techniques to analyze a short data record that is derived from a fourth-order narrowband autoregressive process. Additional experiments with the AR techniques are left to the computer exercises.

Consider the AR(4) process that is generated by the difference equation

$$\begin{aligned} x(n) = {} & 2.7377x(n-1) - 3.7476x(n-2) + 2.6293x(n-3) \\ & - 0.9224x(n-4) + w(n) \end{aligned} \tag{8.122}$$

where $w(n)$ is unit variance white Gaussian noise. The filter that generates $x(n)$ has a pair of complex poles at $z = 0.98e^{\pm j(0.2\pi)}$ and a pair of complex poles at $z = 0.98e^{\pm j(0.3\pi)}$. Using data records of length $N = 128$, an ensemble of 50 spectrum estimates were computed using the Yule-Walker method, the covariance method, the modified covariance method, and Burg's method. Shown in part *a* of Figs. 8.25 to 8.28 are overlay plots of these estimates and in part *b* the ensemble average is plotted along with the true power spectrum. What we observe from these plots is that, for this narrowband process, all of the estimates, except the Yule-Walker method, appear to be unbiased and to have a comparable variance. The Yule-Walker method, on the other hand, is unable to resolve the spectral peaks and has a larger variance.

Selecting the Model Order. A question that remains to be answered in the use of an AR spectrum estimation method is how to select the model order p of the AR process. If the model order that is used is too small, then the resulting spectrum will be smoothed and will have poor resolution. If, on the other hand, the model order is too large, then the spectrum may contain spurious peaks and, as illustrated in Fig. 8.24, may lead to spectral line splitting. Therefore, it would be useful to have a criterion that indicates the appropriate model order to use for a given set of data. One approach would be to increase the model order until the modeling error is minimized. The difficulty with this, however, is that the error is a monotonically nonincreasing function of the model order p. This problem may be overcome by incorporating a penalty function that increases with the model order p. Several criteria have been proposed that include a penalty term that increases *linearly* with p

$$C(p) = N \log \mathcal{E}_p + f(N)p \tag{8.123}$$

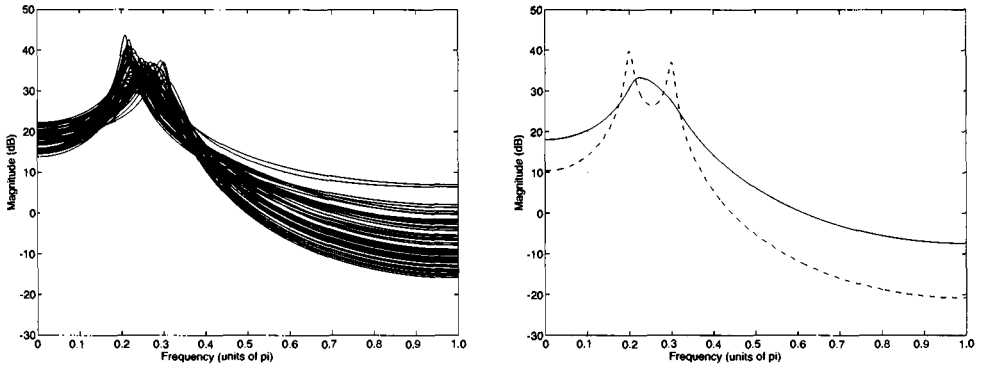


Figure 8.25 *Spectrum estimation of a fourth-order autoregressive process using the Yule-Walker method. (a) Overlay plot of 50 spectrum estimates. (b) The average of the estimates in (a) with the true power spectrum indicated by the dashed line.*

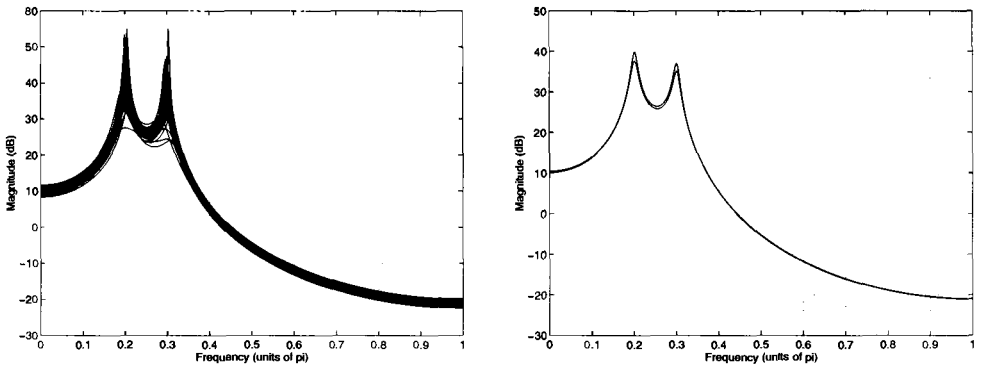


Figure 8.26 *Spectrum estimation of a fourth-order autoregressive process using the covariance method. (a) Overlay plot of 50 spectrum estimates. (b) The average of the estimates in (a) with the true power spectrum indicated by the dashed line.*

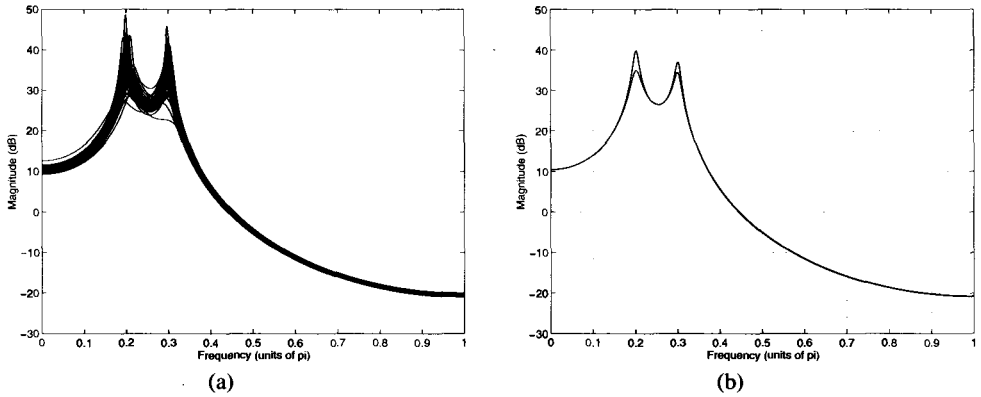


Figure 8.27 *Spectrum estimation of a fourth-order autoregressive process using the modified covariance method. (a) Overlay plot of 50 spectrum estimates. (b) The average of the estimates in (a) with the true power spectrum indicated by the dashed line.*

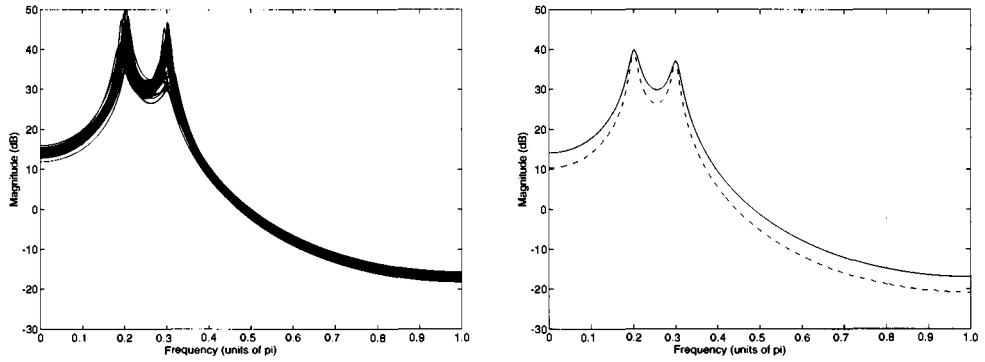


Figure 8.28 *Spectrum estimation of a fourth-order autoregressive process using Burg's method. (a) Overlay plot of 50 spectrum estimates. (b) The average of the estimates in (a) with the true power spectrum indicated by the dashed line.*

Here, $\mathcal{E}_p$ is the modeling error, N is the data record length, and $f(N)$ is a constant that may depend upon N. The idea, then, is to select the value of p that minimizes $C(p)$. Two criteria that are of this form are the Akaike Information Criterion [1,2]

$$\text{AIC}(p) = N \log \mathcal{E}_p + 2p \tag{8.124}$$

and the minimum description length proposed by Rissanen [45],

$$\text{MDL}(p) = N \log \mathcal{E}_p + (\log N)p \tag{8.125}$$

The AIC was derived by minimizing an information theoretic function and includes the penalty $2p$ for any extra AR coefficients that do not significantly reduce the prediction error. It has been observed that the AIC gives an estimate for the order p that is too small when applied to nonautoregressive processes and that it tends to overestimate the order as N increases [23,24,55]. The MDL, on the other hand, contains the penalty term $(\log N)p$, which increases with the data record length N and the model order p. It has been shown that the MDL is a consistent model-order estimator in the sense that it converges to the true order as the number of observations, N, increases [45,58]. Two other model order selection criteria that are often used are Akaike's Final Prediction Error [2],

$$\text{FPE}(p) = \mathcal{E}_p \frac{N + p + 1}{N - p - 1} \tag{8.126}$$

and Parzen's Criterion Autoregressive Transfer function [39]

$$\text{CAT}(p) = \left[\frac{1}{N} \sum_{j=1}^{p} \frac{N - j}{N\mathcal{E}_j} \right] - \frac{N - p}{N\mathcal{E}_p} \tag{8.127}$$

The success of each of these criteria in estimating model orders has been mixed. In fact, for short data sequences, none of the criteria tend to work particularly well [56]. Therefore, these criteria should only be used as "indicators" of the model order. It should also be pointed out that since each of these criteria depends upon the prediction error, $\mathcal{E}_p$, the model order will depend on the modeling technique that is used, e.g. the predicted model order may not be the same when using the autocorrelation method and the Burg algorithm. Table 8.8 summarizes these model order criteria.

Table 8.8 *Model Order Selection Criteria*

$$\text{AIC}(p) = N \log \mathcal{E}_p + 2p$$

$$\text{MDL}(p) = N \log \mathcal{E}_p + (\log N)p$$

$$\text{FPE}(p) = \mathcal{E}_p \frac{N+p+1}{N-p-1}$$

$$\text{CAT}(p) = \left[\frac{1}{N} \sum_{j=1}^{p} \frac{N-j}{N\mathcal{E}_j} \right] - \frac{N-p}{N\mathcal{E}_p}$$

8.5.2 Moving Average Spectrum Estimation

A moving average process may be generated by filtering unit variance white noise, $w(n)$, with an FIR filter as follows:

$$x(n) = \sum_{k=0}^{q} b_q(k) w(n-k) \tag{8.128}$$

As discussed in Section 3.6.3, the relationship between the power spectrum of a moving average process and the coefficients $b_q(k)$ is

$$P_x(e^{j\omega}) = \left| \sum_{k=0}^{q} b_q(k) e^{-jk\omega} \right|^2 \tag{8.129}$$

Equivalently, the power spectrum may be written in terms of the autocorrelation sequence $r_x(k)$ as

$$P_x(e^{j\omega}) = \sum_{k=-q}^{q} r_x(k) e^{-jk\omega} \tag{8.130}$$

where $r_x(k)$ is related to the filter coefficients $b_q(k)$ through the Yule-Walker equations (see Section 3.6.3)

$$r_x(k) = \sum_{l=0}^{q-k} b_q(l+k) b_q^*(l) \quad ; \quad k = 0, 1, \ldots, q \tag{8.131}$$

with $r_x(-k) = r_x^*(k)$ and $r_x(k) = 0$ for $|k| > q$.

With a moving average model, the spectrum may be estimated in one of two ways. The first approach is to take advantage of the fact that the autocorrelation sequence of a moving average process is finite in length. Specifically, since $r_x(k) = 0$ for $|k| > q$, then a natural estimate to use is

$$\boxed{\hat{P}_{MA}(e^{j\omega}) = \sum_{k=-q}^{q} \hat{r}_x(k) e^{-jk\omega}} \tag{8.132}$$

where $\hat{r}_x(k)$ is a suitable estimate of the autocorrelation sequence. Note that although $\hat{P}_{MA}(e^{j\omega})$ is equivalent to the Blackman-Tukey estimate using a rectangular window, there

is a subtle difference in the assumptions that are behind these two estimates. In particular, since Eq. (8.132) assumes that $x(n)$ is a moving average process of order q, then the true autocorrelation sequence is zero for $|k| > q$. Thus, if an unbiased estimate of the autocorrelation sequence is used for $|k| \leq q$, then

$$E\{\hat{P}_{MA}(e^{j\omega})\} = P_x(e^{j\omega})$$

i.e., $\hat{P}_{MA}(e^{j\omega})$ is unbiased. The Blackman-Tukey method, on the other hand, makes no assumptions about $x(n)$ and may be applied to any type of process. Therefore, due to the windowing of the autocorrelation sequence, unless $x(n)$ is a moving average process, the Blackman-Tukey spectrum will be biased.

The second approach is to estimate the moving average parameters, $b_q(k)$, from $x(n)$ and then substitute these estimates into Eq. (8.129) as follows:

$$\boxed{\hat{P}_{MA}(e^{j\omega}) = \left| \sum_{k=0}^{q} \hat{b}_q(k) e^{-jk\omega} \right|^2} \tag{8.133}$$

For example, $b_q(k)$ may be estimated using the two-stage approach developed by Durbin (see Section 4.7.3).

As with autoregressive spectrum estimation, it is useful to have a criterion for estimating the order of the MA model that should be used for a given process $x(n)$. A discussion of some of these estimation methods may be found in [25].

Examples. As we did with the autoregressive methods, here we present only a few examples of MA spectrum estimation rather than attempting to be complete in covering all of the interesting possibilities. Specifically, we consider a fourth-order moving average process that is generated by the difference equation

$$\begin{aligned} x(n) = w(n) &- 1.5857w(n-1) + 1.9208w(n-2) \\ &- 1.5229w(n-3) + 0.9224w(n-4) \end{aligned} \tag{8.134}$$

where $w(n)$ is unit variance white Gaussian noise. The filter that generates $x(n)$ has a pair of complex zeroes at $z = 0.98e^{\pm j(0.2\pi)}$ and a pair of complex zeroes at $z = 0.98e^{\pm j(0.5\pi)}$. Using data records of length $N = 128$, an ensemble of 50 spectrum estimates were computed using the Blackman-Tukey method with a rectangular window that extends from $k = -4$ to $k = 4$. Shown in Fig. 8.29*a* is an overlay plot of these estimates and in *b* is the ensemble average along with the true power spectrum. These plots are repeated in Fig. (8.30) using Durbin's method with $q = 4$ and $p = 32$ (recall that p is the order of the all-pole model that is used to model $x(n)$ prior to estimating the moving average coefficients).

8.5.3 Autoregressive Moving Average Spectrum Estimation

An autoregressive moving average process has a power spectrum of the form

$$P_x(e^{j\omega}) = \frac{\left| \sum_{k=0}^{q} b_q(k) e^{-jk\omega} \right|^2}{\left| 1 + \sum_{k=1}^{p} a_p(k)^{-jk\omega} \right|^2} \tag{8.135}$$

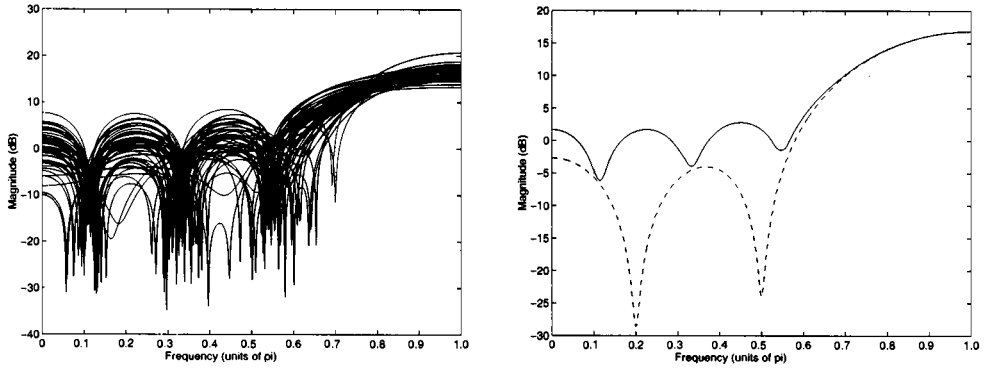


Figure 8.29 *Spectrum estimation of a fourth-order moving average process using the Blackman-Tukey method. (a) Overlay plot of 50 spectrum estimates. (b) The average of the estimates in (a) with the true power spectrum indicated by the dashed line.*

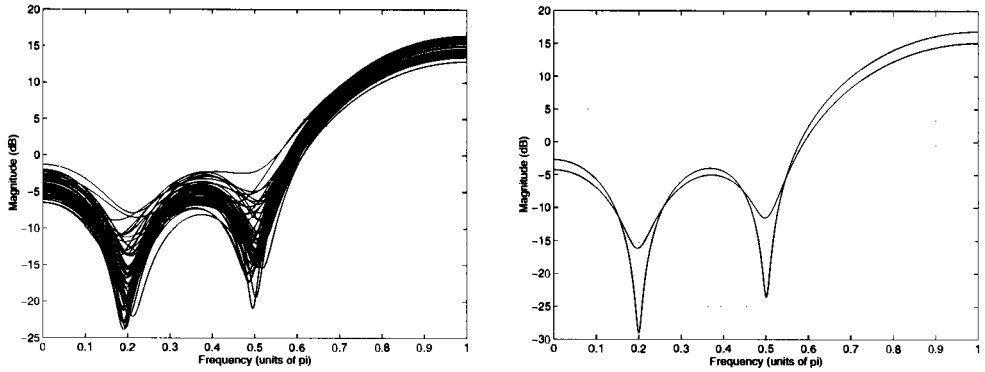


Figure 8.30 *Spectrum estimation of a fourth-order moving average process using Durbin's method. (a) Overlay plot of 50 spectrum estimates. (b) The average of the estimates in (a) with the true power spectrum indicated by the dashed line.*

which may be generated by filtering unit variance white noise with a filter having both poles and zeros,

$$H(z) = \frac{B_q(z)}{A_p(z)} = \frac{\sum_{k=0}^{q} b_q(k) z^{-k}}{1 + \sum_{k=1}^{p} a_p(k) z^{-k}}$$

Following the approach used for AR(p) and MA(q) spectrum estimation, the spectrum of an ARMA(p, q) process may be estimated from Eq. (8.135) using estimates of the model parameters

$$\hat{P}_{ARMA}(e^{j\omega}) = \frac{\left| \sum_{k=0}^{q} \hat{b}_q(k) e^{-jk\omega} \right|^2}{\left| 1 + \sum_{k=1}^{p} \hat{a}_p(k) e^{-jk\omega} \right|^2} \tag{8.136}$$

As discussed in Section 4.7.1, the AR model parameters may be estimated from the modified Yule-Walker equations either directly or by using a least squares approach. Once the coefficients $\hat{a}_p(k)$ have been estimated, a moving average modeling technique such as Durbin's method may be used to estimate the moving average parameters $b_q(k)$. Examples may be found in [25].

8.6 FREQUENCY ESTIMATION

In the previous section, we considered the problem of estimating the power spectrum of a WSS random process that could be modeled as the output of a linear shift-invariant filter that is driven by white noise. Another model of importance is one in which $x(n)$ is a sum of complex exponentials in white noise,

$$x(n) = \sum_{i=1}^{p} A_i e^{jn\omega_i} + w(n) \tag{8.137}$$

It is assumed that the amplitudes A_i are complex,

$$A_i = |A_i| e^{j\phi_i}$$

with ϕ_i uncorrelated random variables that are uniformly distributed over the interval $[-\pi, \pi]$. Although the frequencies and magnitudes of the complex exponentials, ω_i and $|A_i|$, respectively, are not random, they are assumed to be unknown. Thus, the power spectrum of $x(n)$ consists of a set of p impulses of area $2\pi|A_i|$ at frequency ω_i for $i = 1, 2, \ldots, p$, plus the power spectrum of the additive noise $w(n)$. Signals of this form are found in a number of applications such as sonar signal processing and speech processing. Typically, the complex exponentials are the "information bearing" part of the signal and it is the estimation of the frequencies and amplitudes that is of interest rather than the estimation of the power spectrum itself. In the case of sonar signals, for example, the frequencies ω_i may represent bearing or velocity information, whereas for speech signals they would correspond to the formant frequencies [42]. Although it is possible to estimate the frequencies of the complex exponentials from the peaks of the spectrum that is estimated using any of the techniques discussed in the previous sections, this approach would not fully exploit the assumed parametric form of the process. Therefore, in this section we consider *frequency estimation* algorithms that take into account the known properties of the process. The methods that we will be considering are based on an eigendecomposition of the autocorrelation matrix into two subspaces, a *signal* subspace and a *noise* subspace. Once these subspaces have been determined, a frequency estimation function is then used to extract estimates of the frequencies. Before looking at these *subspace methods*, we begin with a discussion of the eigendecomposition of the autocorrelation matrix.

8.6.1 Eigendecomposition of the Autocorrelation Matrix

In order to motivate the use of an eigendecomposition of the autocorrelation matrix as an approach that may be used for frequency estimation, consider the first-order process

$$x(n) = A_1 e^{jn\omega_1} + w(n)$$

that consists of a single complex exponential in white noise. The amplitude of the complex exponential is $A_1 = |A_1| e^{j\phi_1}$ where ϕ_1 is a uniformly distributed random variable, and

$w(n)$ is white noise that has a variance of σ_w^2. As shown in Section 3.6.4, the autocorrelation sequence of $x(n)$ is

$$r_x(k) = P_1 e^{jk\omega_1} + \sigma_w^2 \delta(k)$$

where $P_1 = |A_1|^2$ is the *power* in the complex exponential. Therefore, the $M \times M$ autocorrelation matrix for $x(n)$ is a sum of an autocorrelation matrix due to the signal, $\mathbf{R}_s$, and an autocorrelation matrix due to the noise, $\mathbf{R}_n$,

$$\boxed{\mathbf{R}_x = \mathbf{R}_s + \mathbf{R}_n} \tag{8.138}$$

where the signal autocorrelation matrix is

$$\mathbf{R}_s = P_1 \begin{bmatrix} 1 & e^{-j\omega_1} & e^{-j2\omega_1} & \cdots & e^{-j(M-1)\omega_1} \\ e^{j\omega_1} & 1 & e^{-j\omega_1} & \cdots & e^{-j(M-2)\omega_1} \\ e^{j2\omega_1} & e^{j\omega_1} & 1 & \cdots & e^{-j(M-3)\omega_1} \\ \vdots & \vdots & \vdots & & \vdots \\ e^{j(M-1)\omega_1} & e^{j(M-2)\omega_1} & e^{j(M-3)\omega_1} & \cdots & 1 \end{bmatrix} \tag{8.139}$$

and has a rank of one, and the noise autocorrelation matrix is diagonal,

$$\mathbf{R}_n = \sigma_w^2 \mathbf{I}$$

and has full rank. Note that if we define

$$\mathbf{e}_1 = [1, e^{j\omega_1}, e^{j2\omega_1}, \ldots, e^{j(M-1)\omega_1}]^T \tag{8.140}$$

then $\mathbf{R}_s$ may be written in terms of $\mathbf{e}_1$ as follows:

$$\mathbf{R}_s = P_1 \mathbf{e}_1 \mathbf{e}_1^H$$

Since the rank of $\mathbf{R}_s$ is equal to one, then $\mathbf{R}_s$ has only one nonzero eigenvalue. With

$$\mathbf{R}_s \mathbf{e}_1 = P_1 \left(\mathbf{e}_1 \mathbf{e}_1^H\right) \mathbf{e}_1 = P_1 \mathbf{e}_1 \left(\mathbf{e}_1^H \mathbf{e}_1\right) = M P_1 \mathbf{e}_1$$

it follows that the nonzero eigenvalue is equal to $M P_1$, and that $\mathbf{e}_1$ is the corresponding eigenvector. In addition, since $\mathbf{R}_s$ is Hermitian then the remaining eigenvectors, $\mathbf{v}_2, \mathbf{v}_3, \ldots, \mathbf{v}_M$, will be orthogonal to $\mathbf{e}_1$,[15]

$$\mathbf{e}_1^H \mathbf{v}_i = 0 \quad ; \quad i = 2, 3, \ldots, M \tag{8.141}$$

Finally, note that if we let λ_i^s be the eigenvalues of $\mathbf{R}_s$, then

$$\mathbf{R}_x \mathbf{v}_i = \left(\mathbf{R}_s + \sigma_w^2 \mathbf{I}\right) \mathbf{v}_i = \lambda_i^s \mathbf{v}_i + \sigma_w^2 \mathbf{v}_i = (\lambda_i^s + \sigma_w^2) \mathbf{v}_i \tag{8.142}$$

Therefore, the eigenvectors of $\mathbf{R}_x$ are the same as those of $\mathbf{R}_s$, and the eigenvalues of $\mathbf{R}_x$ are

$$\lambda_i = \lambda_i^s + \sigma_w^2$$

As a result, the largest eigenvalue of $\mathbf{R}_x$ is

$$\lambda_{\max} = M P_1 + \sigma_w^2$$

and the remaining $M - 1$ eigenvalues are equal to σ_w^2. Thus, it is possible to extract all of the parameters of interest about $x(n)$ from the eigenvalues and eigenvectors of $\mathbf{R}_x$ as follows:

[15]Recall that for a Hermitian matrix the eigenvectors corresponding to distinct eigenvalues are orthogonal. See Property 4 on p. 43 of Chapter 2.

1. Perform an eigendecomposition of the autocorrelation matrix, $\mathbf{R}_x$. The largest eigenvalue will be equal to $MP_1 + \sigma_w^2$ and the remaining eigenvalues will be equal to σ_w^2.
2. Use the eigenvalues of $\mathbf{R}_x$ to solve for the power P_1 and the noise variance as follows:

$$\sigma_w^2 = \lambda_{\min}$$
$$P_1 = \frac{1}{M}(\lambda_{\max} - \lambda_{\min})$$

3. Determine the frequency ω_1 from the eigenvector $\mathbf{v}_{\max}$ that is associated with the largest eigenvalue using, for example, the second coefficient of $\mathbf{v}_{\max}$,[16]

$$\omega_i = \arg\{v_{\max}(1)\}$$

The following example illustrates the procedure.

Example 8.6.1 *Eigendecomposition of a Complex Exponential in Noise*

Let $x(n)$ be a first-order harmonic process consisting of a single complex exponential in white noise,

$$x(n) = A_1 e^{jn\omega_1} + w(n)$$

with a 2×2 autocorrelation matrix given by

$$\mathbf{R}_x = \begin{bmatrix} 3 & 2(1-j) \\ 2(1+j) & 3 \end{bmatrix}$$

The eigenvalues of $\mathbf{R}_x$ are

$$\lambda_{1,2} = 3 \pm |2(1+j)| = 3 \pm 2\sqrt{2}$$

and the eigenvectors are

$$\mathbf{v}_{1,2} = \begin{bmatrix} 1 \\ \pm\frac{\sqrt{2}}{2}(1+j) \end{bmatrix}$$

Therefore, the variance of the white noise is

$$\sigma_w^2 = \lambda_{\min} = 3 - 2\sqrt{2}$$

and the power in the complex exponential is

$$P_1 = \frac{1}{M}(\lambda_{\max} - \lambda_{\min}) = 2\sqrt{2}$$

Finally, the frequency of the complex exponential is

$$\omega_1 = \arg\left\{\frac{\sqrt{2}}{2}(1+j)\right\} = \pi/4$$

It should be pointed out that, for a first-order harmonic process, finding the parameters of the process is actually a bit easier than this. For example, note that

$$r_x(1) = 2(1+j) = 2\sqrt{2}e^{j\pi/4} = P_1 e^{j\omega_1}$$

[16]The components of the eigenvectors are numbered from $v_i(0)$ to $v_i(M-1)$.

Therefore, we see immediately that $P_1 = 2\sqrt{2}$ and $\omega_1 = \pi/4$. Furthermore, once P_1 is known, then the variance of the white noise may be determined as follows:

$$\sigma_w^2 = r_x(0) - P_1 = 3 - 2\sqrt{2}$$

In practice, the approach described above is of limited value in frequency estimation since it requires that the autocorrelation matrix be known exactly. Although an estimated autocorrelation matrix could be used in place of $\mathbf{R}_x$, if this is done then the largest eigenvalue will only be approximately equal to $P_1 + \sigma_w^2$ and the corresponding eigenvector will only be an approximation to $\mathbf{e}_1$. Since the eigenvalues and eigenvectors may be quite sensitive to small errors in $r_x(k)$, instead of estimating the frequency of the complex exponential from a single eigenvector, we may consider using a weighted average as follows. Let $\mathbf{v}_i$ be a *noise eigenvector* of $\mathbf{R}_x$, i.e., one that has an eigenvalue of σ_w^2, and let $v_i(k)$ be the kth component of $\mathbf{v}_i$. If we compute the discrete-time Fourier transform of the coefficients in $\mathbf{v}_i$,

$$V_i(e^{j\omega}) = \sum_{k=0}^{M-1} v_i(k) e^{-jk\omega} = \mathbf{e}^H \mathbf{v}_i \tag{8.143}$$

then the orthogonality condition given in Eq. (8.141) implies that $V_i(e^{j\omega})$ will be equal to zero at $\omega = \omega_1$, the frequency of the complex exponential. Therefore, if we form the *frequency estimation function*

$$\hat{P}_i(e^{j\omega}) = \frac{1}{\left|\sum_{k=0}^{M-1} v_i(k) e^{-jk\omega}\right|^2} = \frac{1}{|\mathbf{e}^H \mathbf{v}_i|^2} \tag{8.144}$$

then $\hat{P}_i(e^{j\omega})$ will be large (in theory, infinite) at $\omega = \omega_1$. Thus, the location of the peak of this frequency estimation function may be used to estimate the frequency of the complex exponential. However, since Eq. (8.144) uses only a single eigenvector and, therefore, may be sensitive to errors in the estimation of $\mathbf{R}_x$, we may consider using a weighted average of all of the noise eigenvectors as follows:

$$\hat{P}(e^{j\omega}) = \frac{1}{\sum_{i=2}^{M} \alpha_i |\mathbf{e}^H \mathbf{v}_i|^2} \tag{8.145}$$

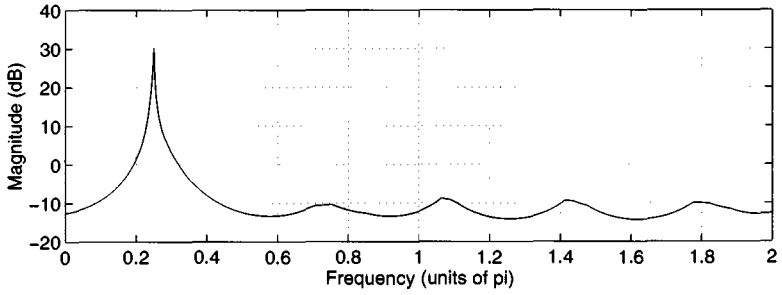
where α_i are some appropriately chosen constants.

Example 8.6.2 ***Eigendecomposition of a Complex Exponential in Noise (cont.)***

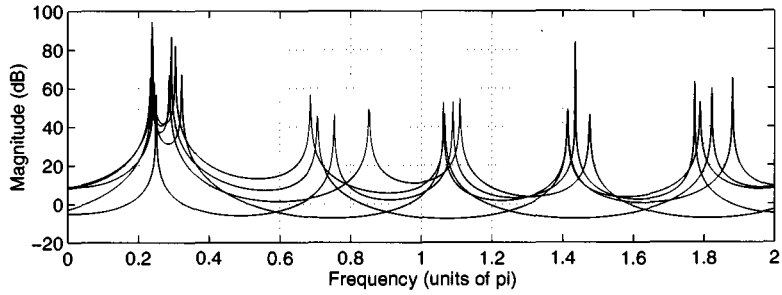
Let $x(n)$ be a WSS process consisting of a single complex exponential in unit variance white noise,

$$x(n) = 4\, e^{j(n\pi/4+\phi)} + w(n)$$

where ϕ is a uniformly distributed random variable. Using $N = 64$ values of $x(n)$, a 6×6 autocorrelation matrix was estimated and an eigendecomposition performed. Shown in Fig. 8.31*a* is a plot of the frequency estimation function defined in Eq. (8.145) with $\alpha_i = 1$. The peak of $\hat{P}(e^{j\omega})$ occurs at frequency $\omega = 0.2539\pi$ and the minimum eigenvalue is



(a)



(b)

Figure 8.31 *Frequency estimation functions of a single complex exponential in white noise. (a) The frequency estimation function that uses all of the noise eigenvectors with a weighting* $\alpha_i = 1$. *(b) An overlay plot of the frequency estimation functions* $V_i(e^{j\omega}) = 1/|\mathbf{e}^H \mathbf{v}_i|^2$ *that are derived from each noise eigenvector.*

$\lambda_{\min} = 1.08$, which is close to the variance of the white noise. By contrast, shown in Fig. 8.31*b* is an overlay plot of the frequency estimation functions $V_i(e^{j\omega})$ for each of the noise eigenvectors as defined in Eq. (8.143). As we see from this figure, although each plot has a peak that is close to $\omega = 0.25\pi$, with a single plot it is difficult to distinguish the correct peak from a spurious one.

Let us now consider what happens in the case of two complex exponentials in white noise,

$$x(n) = A_1 e^{jn\omega_1} + A_2 e^{jn\omega_2} + w(n)$$

where $A_i = |A_i| e^{j\phi_i}$ for $i = 1, 2$ are the amplitudes of the complex exponentials, and ω_1 and ω_2 are the frequencies with $\omega_1 \neq \omega_2$. If the variance of $w(n)$ is σ_w^2 then the autocorrelation of $x(n)$ is

$$r_x(k) = P_1 e^{jk\omega_1} + P_2 e^{jk\omega_2} + \sigma_w^2 \delta(k)$$

where $P_1 = |A_1|^2$ and $P_2 = |A_2|^2$. Thus, the autocorrelation matrix may again be written

as a sum

$$\mathbf{R}_x = P_1\mathbf{e}_1\mathbf{e}_1^H + P_2\mathbf{e}_2\mathbf{e}_2^H + \sigma_w^2\mathbf{I}$$

where

$$\mathbf{R}_s = P_1\mathbf{e}_1\mathbf{e}_1^H + P_2\mathbf{e}_2\mathbf{e}_2^H$$

is a rank two matrix representing the component of $\mathbf{R}_x$ that is due to the signal, and

$$\mathbf{R}_n = \sigma_w^2\mathbf{I}$$

is a diagonal matrix that is due to the noise. A more concise way to express this decomposition is to write $\mathbf{R}_x$ as follows:

$$\mathbf{R}_x = \mathbf{E}\mathbf{P}\mathbf{E}^H + \sigma_w^2\mathbf{I} \tag{8.146}$$

where

$$\mathbf{E} = [\mathbf{e}_1, \mathbf{e}_2]$$

is an $M \times 2$ matrix containing the two signal vectors $\mathbf{e}_1$ and $\mathbf{e}_2$ and $\mathbf{P} = \text{diag}\{P_1, P_2\}$ is a diagonal matrix containing the signal powers.

In addition to decomposing $\mathbf{R}_x$ into a sum of two autocorrelation matrices as in Eq. (8.146), we may also perform an eigendecomposition of $\mathbf{R}_x$ as follows. Let $\mathbf{v}_i$ and λ_i be the eigenvectors and eigenvalues of $\mathbf{R}_x$, respectively, with the eigenvalues arranged in decreasing order,

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_M$$

Since $\mathbf{R}_x = \mathbf{R}_s + \sigma_w^2\mathbf{I}$ then

$$\lambda_i = \lambda_i^s + \sigma_w^2$$

where λ_i^s are the eigenvalues of $\mathbf{R}_s$. Since the rank of $\mathbf{R}_s$ is equal to two, then $\mathbf{R}_s$ has only two nonzero eigenvalues, and both of these are greater than zero ($\mathbf{R}_s$ is nonnegative definite). Therefore, the first two eigenvalues of $\mathbf{R}_x$ are greater than σ_w^2 and the remaining eigenvalues are equal to σ_w^2. Thus, the eigenvalues and eigenvectors of $\mathbf{R}_x$ may be divided into two groups. The first group, consisting of the two eigenvectors that have eigenvalues greater than σ_w^2, are referred to as *signal eigenvectors* and span a two-dimensional subspace called the *signal subspace*. The second group, consisting of those eigenvectors that have eigenvalues equal to σ_w^2, are referred to as the *noise eigenvectors* and span an $(M-2)$-dimensional subspace called the *noise subspace*.[17] Since $\mathbf{R}_x$ is Hermitian, the eigenvectors $\mathbf{v}_i$ form an orthonormal set (see the discussion following Property 4 on p. 43). Therefore, the signal and noise subspaces are orthogonal. That is to say, for any vector $\mathbf{u}$ in the signal subspace and for any vector $\mathbf{v}$ in the noise subspace, $\mathbf{u}^H\mathbf{v} = 0$. The geometry of these subspaces is illustrated in Fig. 8.32.

Unlike the case for a single complex exponential, with a sum of two complex exponentials in noise, the signal eigenvectors will generally not be equal to $\mathbf{e}_1$ and $\mathbf{e}_2$.[18] Nevertheless, $\mathbf{e}_1$ and $\mathbf{e}_2$ will lie in the signal subspace that is spanned by the signal eigenvectors $\mathbf{v}_1$ and $\mathbf{v}_2$, and since the signal subspace is orthogonal to the noise subspace, then $\mathbf{e}_1$ and $\mathbf{e}_2$ will be

[17] Note that this term is a bit misleading since the noise has components in both the noise and signal subspaces.

[18] This may be seen easily by noting that the signal vectors will not, in general, be orthogonal whereas the eigenvectors will always be orthogonal.

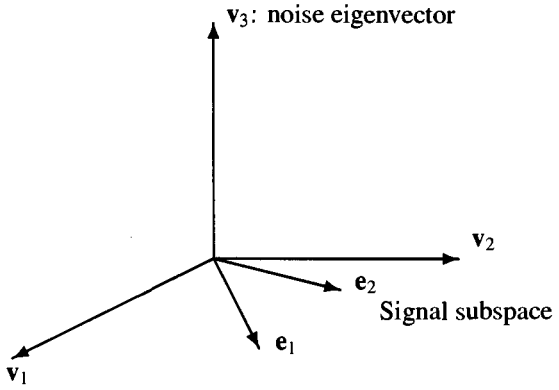


Figure 8.32 *Geometrical interpretation of the orthogonality of the signal and noise subspaces. The one-dimensional noise subspace is spanned by the noise eigenvector* $\mathbf{v}_3$ *and the signal subspace which contains the signal vectors* $\mathbf{e}_1$ *and* $\mathbf{e}_2$ *is spanned by the signal eigenvectors* $\mathbf{v}_1$ *and* $\mathbf{v}_2$.

orthogonal to the noise eigenvectors $\mathbf{v}_i$, i.e.,

$$\begin{aligned} \mathbf{e}_1^H \mathbf{v}_i &= 0 \quad ; \quad i = 3, 4, \ldots, M \\ \mathbf{e}_2^H \mathbf{v}_i &= 0 \quad ; \quad i = 3, 4, \ldots, M \end{aligned} \tag{8.147}$$

Therefore, as in the case of one complex exponential, the complex exponential frequencies, ω_1 and ω_2, may be estimated using a frequency estimation function of the form

$$\hat{P}(e^{j\omega}) = \frac{1}{\sum_{i=3}^{M} \alpha_i |\mathbf{e}^H \mathbf{v}_i|^2}.$$

Let us now consider the general case of a wide-sense stationary process consisting of p *distinct* complex exponentials in white noise. The $M \times M$ autocorrelation sequence is

$$r_x(k) = \sum_{i=1}^{p} P_i e^{jk\omega_i} + \sigma_w^2 \delta(k)$$

where $P_i = |A_i|^2$ is the *power* in the ith component. Therefore, the autocorrelation matrix may be written as

$$\mathbf{R}_x = \mathbf{R}_s + \mathbf{R}_n = \sum_{i=1}^{p} P_i \mathbf{e}_i \mathbf{e}_i^H + \sigma_w^2 \mathbf{I} \tag{8.148}$$

where

$$\mathbf{e}_i = [1, \; e^{j\omega_i}, \; e^{j2\omega_i}, \; \ldots, \; e^{j(M-1)\omega_i}]^T \quad ; \quad i = 1, 2, \ldots, p$$

is a set of p linearly independent vectors. As was the case for two complex exponentials, Eq. (8.148) may be written concisely as follows:

$$\boxed{\mathbf{R}_x = \mathbf{E}\mathbf{P}\mathbf{E}^H + \sigma_w^2 \mathbf{I}} \tag{8.149}$$

where $\mathbf{E} = [\mathbf{e}_1, \ldots, \mathbf{e}_p]$ is an $M \times p$ matrix containing the p signal vectors, $\mathbf{e}_i$, and $\mathbf{P} = \text{diag}\{P_1, \ldots, P_p\}$ is a diagonal matrix of signal powers. Since the eigenvalues of $\mathbf{R}_x$ are $\lambda_i = \lambda_i^s + \sigma_w^2$ where λ_i^s are the eigenvalues of $\mathbf{R}_s$, and since $\mathbf{R}_s$ is a matrix of rank p, then the first p eigenvalues of $\mathbf{R}_x$ will be greater than σ_w^2 and the last $M - p$ eigenvalues will be equal to σ_w^2. Therefore, the eigenvalues and eigenvectors of $\mathbf{R}_x$ may again be divided into two groups: the signal eigenvectors $\mathbf{v}_1, \ldots, \mathbf{v}_p$ that have eigenvalues greater than σ_w^2, and the noise eigenvectors $\mathbf{v}_{p+1}, \ldots, \mathbf{v}_M$ that have eigenvalues equal to σ_w^2. Assuming that the eigenvectors have been normalized to have unit norm, we may use the spectral theorem (p. 44) to decompose $\mathbf{R}_x$ as follows:

$$\mathbf{R}_x = \sum_{i=1}^{p} (\lambda_i^s + \sigma_w^2)\mathbf{v}_i\mathbf{v}_i^H + \sum_{i=p+1}^{M} \sigma_w^2 \mathbf{v}_i\mathbf{v}_i^H$$

In matrix notation, this decomposition may be written as

$$\boxed{\mathbf{R}_x = \mathbf{V}_{ss}\mathbf{V}_s^H + \mathbf{V}_{nn}\mathbf{V}_n^H} \tag{8.150}$$

where

$$\mathbf{V}_s = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_p]$$

is the $M \times p$ matrix of signal eigenvectors and

$$\mathbf{V}_n = [\mathbf{v}_{p+1}, \mathbf{v}_{p+2}, \ldots, \mathbf{v}_M]$$

is the $M \times (M - p)$ matrix of noise eigenvectors, and where $_s$ and $_n$ are diagonal matrices that contain the eigenvalues $\lambda_i = \lambda_i^s + \sigma_w^2$ and $\lambda_i = \sigma_w^2$, respectively. Later, we will be interested in projecting a vector onto either the signal subspace or the noise subspace. The projection matrices $\mathbf{P}_s$ and $\mathbf{P}_n$ that will perform these projections onto the signal and noise subspaces, respectively, are (see p. 34)

$$\boxed{\mathbf{P}_s = \mathbf{V}_s\mathbf{V}_s^H \quad ; \quad \mathbf{P}_n = \mathbf{V}_n\mathbf{V}_n^H} \tag{8.151}$$

As was the case for one and two complex exponentials in white noise, the orthogonality of the signal and noise subspaces may be used to estimate the frequencies of the complex exponentials. Specifically, since each signal vector $\mathbf{e}_1, \ldots, \mathbf{e}_p$ lies in the signal subspace, this orthogonality implies that $\mathbf{e}_i$ will be orthogonal to each of the noise eigenvectors,

$$\mathbf{e}_i^H\mathbf{v}_k = 0 \quad ; \quad \begin{matrix} i = 1, 2, \ldots, p \\ k = p+1, p+2, \ldots, M \end{matrix}$$

Therefore, the frequencies may be estimated using a frequency estimation function such as

$$\hat{P}(e^{j\omega}) = \frac{1}{\displaystyle\sum_{i=p+1}^{M} \alpha_i |\mathbf{e}^H\mathbf{v}_i|^2} \tag{8.152}$$

In the following sections, we will develop several different types of frequency estimation algorithms that are based on Eq. (8.152). We begin with the Pisarenko harmonic decomposition, which uses a frequency estimator of this form with $M = p + 1$ and $\alpha_M = 1$.

8.6.2 Pisarenko Harmonic Decomposition

In 1973, V. Pisarenko considered the problem of estimating the frequencies of a sum of complex exponentials in white noise [41]. Based on a theorem of Carathedory, he demonstrated that the frequencies could be derived from the eigenvector corresponding to the minimum eigenvalue of the autocorrelation matrix. Although the resulting technique, referred to as the *Pisarenko harmonic decomposition*, is somewhat limited in its usefulness due to its sensitivity to noise, this decomposition is of theoretical interest, has led to important insights into the frequency estimation problem, and has provided the stimulus for the development of other eigenvalue decomposition methods that are more robust.

In the Pisarenko harmonic decomposition, it is assumed that $x(n)$ is a sum of p complex exponentials in white noise, and that the number of complex exponentials, p, is known. It is also assumed that $p+1$ values of the autocorrelation sequence are either known or have been estimated. With a $(p+1) \times (p+1)$ autocorrelation matrix, the dimension of the noise subspace is equal to one, and it is spanned by the eigenvector corresponding to the minimum eigenvalue, $\lambda_{\min} = \sigma_w^2$. Denoting this noise eigenvector by $\mathbf{v}_{\min}$, it follows that $\mathbf{v}_{\min}$ will be orthogonal to each of the signal vectors, $\mathbf{e}_i$,

$$\mathbf{e}_i^H \mathbf{v}_{\min} = \sum_{k=0}^{p} v_{\min}(k) e^{-jk\omega_i} = 0 \quad ; \quad i = 1, 2, \ldots, p \tag{8.153}$$

Therefore,

$$V_{\min}(e^{j\omega}) = \sum_{k=0}^{p} v_{\min}(k) e^{-jk\omega}$$

is equal to zero at each of the complex exponential frequencies ω_i for $i = 1, 2, \ldots, p$. Consequently, the z-transform of the noise eigenvector, referred to as an *eigenfilter*, has p zeros on the unit circle,

$$\boxed{V_{\min}(z) = \sum_{k=0}^{p} v_{\min}(k) z^{-k} = \prod_{k=1}^{p} \left(1 - e^{j\omega_k} z^{-1}\right)} \tag{8.154}$$

and the frequencies of the complex exponentials may be extracted from the roots of the eigenfilter. As an alternative to rooting $V_{\min}(z)$, we may also form the *frequency estimation function*

$$\boxed{\hat{P}_{PHD}(e^{j\omega}) = \frac{1}{|\mathbf{e}^H \mathbf{v}_{\min}|^2}} \tag{8.155}$$

which is a special case of Eq. (8.152) with $M = p+1$ and $\alpha_{p+1} = 1$. Since $\hat{P}_{PHD}(e^{j\omega})$ will be large (in theory, infinite) at the frequencies of the complex exponentials, the locations of the peaks in $\hat{P}_{PHD}(e^{j\omega})$ may be used as frequency estimates. Although written in the form of a power spectrum, $\hat{P}_{PHD}(e^{j\omega})$ is called a *pseudospectrum* (sometimes referred to as an *eigenspectrum*) since it does not contain any information about the power in the complex exponentials, nor does it contain a component due to the noise.

Once the frequencies of the complex exponentials have been determined, the powers P_i may be found from the eigenvalues of $\mathbf{R}_x$ as follows. Let us assume that the signal subspace eigenvectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_p$, have been normalized so that $\mathbf{v}_i^H \mathbf{v}_i = 1$. With

$$\mathbf{R}_x \mathbf{v}_i = \lambda_i \mathbf{v}_i \quad ; \quad i = 1, 2, \ldots, p \tag{8.156}$$

if both sides of Eq. (8.156) are multiplied on the left by $\mathbf{v}_i^H$, then

$$\mathbf{v}_i^H \mathbf{R}_x \mathbf{v}_i = \lambda_i \mathbf{v}_i^H \mathbf{v}_i = \lambda_i \quad ; \quad i = 1, 2, \ldots, p \tag{8.157}$$

Substituting the expression for $\mathbf{R}_x$ given in Eq. (8.148) into Eq. (8.157) we have

$$\mathbf{v}_i^H \mathbf{R}_x \mathbf{v}_i = \mathbf{v}_i^H \left\{ \sum_{k=1}^{p} P_k \mathbf{e}_k \mathbf{e}_k^H + \sigma_w^2 \mathbf{I} \right\} \mathbf{v}_i = \lambda_i$$

which may be simplified to

$$\boxed{\sum_{k=1}^{p} P_k |\mathbf{e}_k^H \mathbf{v}_i|^2 = \lambda_i - \sigma_w^2 \quad ; \quad i = 1, 2, \ldots, p} \tag{8.158}$$

Note that the terms $|\mathbf{e}_k^H \mathbf{v}_i|^2$ in the sum correspond to the squared magnitude of the DTFT of the signal subspace eigenvector $\mathbf{v}_i$ evaluated at frequency ω_k,

$$|\mathbf{e}_k^H \mathbf{v}_i|^2 = \left| V_i(e^{j\omega_k}) \right|^2$$

where

$$V_i(e^{j\omega}) = \sum_{l=0}^{p} v_i(l) e^{-jl\omega}$$

Therefore, Eq. (8.158) may also be written as

$$\sum_{k=1}^{p} P_k |V_i(e^{j\omega_k})|^2 = \lambda_i - \sigma_\omega^2 \quad ; \quad i = 1, 2, \ldots, p \tag{8.159}$$

Equation (8.159) is a set of p linear equations in the p unknowns, P_k,

$$\begin{bmatrix} |V_1(e^{j\omega_1})|^2 & |V_1(e^{j\omega_2})|^2 & \cdots & |V_1(e^{j\omega_p})|^2 \\ |V_2(e^{j\omega_1})|^2 & |V_2(e^{j\omega_2})|^2 & \cdots & |V_2(e^{j\omega_p})|^2 \\ \vdots & \vdots & & \vdots \\ |V_p(e^{j\omega_1})|^2 & |V_p(e^{j\omega_2})|^2 & \cdots & |V_p(e^{j\omega_p})|^2 \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ \vdots \\ P_p \end{bmatrix} = \begin{bmatrix} \lambda_1 - \sigma_w^2 \\ \lambda_2 - \sigma_w^2 \\ \vdots \\ \lambda_p - \sigma_w^2 \end{bmatrix} \tag{8.160}$$

which may be solved for the powers P_k. Thus, as summarized in Table 8.9, the Pisarenko

Table 8.9 *Pisarenko's Method for Frequency Estimation*

Step 1: Given that a process consists of p complex exponentials in white noise, find the minimum eigenvalue $\lambda_{\min}$ and the corresponding eigenvector $\mathbf{v}_{\min}$ of the $(p+1) \times (p+1)$ autocorrelation matrix $\mathbf{R}_x$.

Step 2: Set the white noise power equal to the minimum eigenvalue, $\lambda_{\min} = \sigma_w^2$, and set the frequencies equal to the angles of the roots of the eigenfilter

$$V_{\min}(z) = \sum_{k=0}^{p} v_{\min}(k) z^{-k}$$

or the location of the peaks in the frequency estimation function

$$\hat{P}_{PHD}(e^{j\omega}) = \frac{1}{|\mathbf{e}^H \mathbf{v}_{\min}|^2}$$

Step 3: Compute the powers of the complex exponentials by solving the linear equations (8.160).

The Pisarenko Harmonic Decomposition

```
function [vmin,sigma] = phd(x,p)
%
   x = x(:);
   R = covar(x,p+1);
   [v,d]=eig(R);
   sigma=min(diag(d));
   index=find(diag(d)==sigma);
   vmin = v(:,index);
   end;
```

Figure 8.33 *A* MATLAB *program to estimate the frequencies of p complex exponentials in white noise using Pisarenko's method.*

harmonic decomposition estimates the complex exponential frequencies either from the roots of the eigenfilter $V_{\min}(z)$ or from the locations of the peaks in the frequency estimation function $\hat{P}_{PHD}(e^{j\omega})$, and then solves the linear equations (8.160) for the powers of the complex exponentials. A MATLAB program for the Pisarenko decomposition is given in Fig 8.33.

Example 8.6.3 ***Pisarenko's Method for Two Complex Exponentials in Noise***

Suppose that the first three autocorrelations of a random process consisting of two complex exponentials in white noise are

$$\begin{aligned} r_x(0) &= 6 \\ r_x(1) &= 1.92705 + j4.58522 \\ r_x(2) &= -3.42705 + j3.49541 \end{aligned}$$

Let us use Pisarenko's method to find the frequencies and powers of the complex exponentials. Since $p = 2$, we must perform an eigendecomposition of the 3×3 autocorrelation matrix

$$\mathbf{R}_x = \begin{bmatrix} 6 & 1.92705 - j4.58522 & -3.42705 - j3.49541 \\ 1.92705 + j4.58522 & 6 & 1.92705 - j4.58522 \\ -3.42705 + j3.49541 & 1.92705 + j4.58522 & 6 \end{bmatrix}$$

The eigenvectors are

$$\mathbf{V} = \begin{bmatrix} \mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3 \end{bmatrix} = \begin{bmatrix} 0.5763 - 0.0000j & -0.2740 + 0.6518j & -0.2785 - 0.3006j \\ 0.2244 + 0.5342j & 0.0001 + 0.0100j & -0.3209 + 0.7492j \\ -0.4034 + 0.4116j & 0.2830 + 0.6480j & 0.4097 - 0.0058j \end{bmatrix}$$

and the eigenvalues are

$$\lambda_1 = 15.8951 \quad ; \quad \lambda_2 = 1.1049 \quad ; \quad \lambda_3 = 1.0000$$

Therefore, the minimum eigenvalue is $\lambda_{\min} = 1$ and the corresponding eigenvector is

$$\mathbf{v}_{\min} = \begin{bmatrix} -0.2785 - 0.3006j \\ -0.3209 + 0.7492j \\ 0.4097 - 0.0058j \end{bmatrix}$$

Since the roots of the eigenfilter are

$$z_1 = 0.5 + j0.8660 = e^{j\pi/3} \quad ; \quad z_2 = 0.3090 + j0.9511 = e^{j2\pi/5}$$

then the frequencies of the complex exponentials are

$$\omega_1 = \pi/3 \quad ; \quad \omega_2 = 2\pi/5$$

To find the powers in the complex exponentials, we must compute the squared magnitude of the DTFT of the signal eigenvectors, $\mathbf{v}_1$ and $\mathbf{v}_2$, at the complex exponential frequencies ω_1 and ω_2. With

$$|V_1(e^{j\omega_1})|^2 = 2.9685 \quad ; \quad |V_1(e^{j\omega_2})|^2 = 2.9861$$

and

$$|V_2(e^{j\omega_1})|^2 = 0.0315 \quad ; \quad |V_2(e^{j\omega_2})|^2 = 0.0139$$

then Eq. (8.160) becomes

$$\begin{bmatrix} 2.9685 & 2.9861 \\ 0.0315 & 0.0139 \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \end{bmatrix} = \begin{bmatrix} \lambda_1 - \sigma_w^2 \\ \lambda_2 - \sigma_w^2 \end{bmatrix}$$

where $\sigma_w^2 = \lambda_{\min} = 1$. Solving for P_1 and P_2 we find

$$P_1 = 2 \quad ; \quad P_2 = 3$$

In the previous example, we used Pisarenko's method to estimate the frequencies of two complex exponentials in white noise. In the following example we consider the problem of estimating the frequency of a single sinusoid in white noise. Although a sinusoid is a sum of two complex exponentials, the two frequencies are constrained to be negatives of each other, $\omega_2 = -\omega_1$. This constraint results in an autocorrelation sequence that is real-valued, which forces the eigenvectors to be real and, therefore, constrains the roots of the eigenfilter to occur in complex conjugate pairs.

Example 8.6.4 *Pisarenko's Method for One Sinusoid*

Let $x(n)$ be a random phase sinusoid in white noise

$$x(n) = A\sin(n\omega_0 + \phi) + w(n)$$

with

$$r_x(0) = 2.2 \quad ; \quad r_x(1) = 1.3 \quad ; \quad r_x(2) = 0.8$$

The eigenvalues of the 3×3 autocorrelation matrix $\mathbf{R}_x = \text{toep}\{2.2, 1.3, 0.8\}$ are

$$\lambda_1 = 4.4815 \quad ; \quad \lambda_2 = 1.4 \quad ; \quad \lambda_3 = 0.7185$$

Therefore, the white noise power is

$$\sigma_w^2 = \lambda_{\min} = 0.7185$$

The eigenvectors of $\mathbf{R}_x$ are

$$\mathbf{v}_1 = \begin{bmatrix} 0.5506 \\ 0.6275 \\ 0.5506 \end{bmatrix}; \quad \mathbf{v}_2 = \begin{bmatrix} -0.7071 \\ 0 \\ 0.7071 \end{bmatrix}; \quad \mathbf{v}_3 = \begin{bmatrix} 0.4437 \\ -0.7787 \\ 0.4437 \end{bmatrix}$$

Note the symmetry of the eigenvectors (See Problem 2.9). To estimate the frequency of the sinusoid we find the roots of the eigenfilter $V_{\min}(z)$, which is the z-transform of $\mathbf{v}_3$,

$$V_{\min}(z) = 0.4437(1 - 1.755z^{-1} + z^{-2})$$

Rooting the eigenfilter we find that $V_{\min}(z)$ has roots at $z = e^{\pm j\omega_0}$ where

$$2\cos\omega_0 = 1.755$$

or

$$\omega_0 = 0.159\pi$$

Finally, the power in the sinusoid may be estimated using Eq. (8.160). However, the computation may be simplified by taking into account the fact that $x(n)$ contains a single sinusoid. Specifically, since the autocorrelation sequence for a single sinusoid in white noise is (see Section 3.6.4)

$$r_x(k) = \frac{1}{2}A^2\cos(k\omega_0) + \sigma_w^2\delta(k)$$

then, for $k = 0$,

$$r_x(0) = \frac{1}{2}A^2 + \sigma_w^2$$

With $r_x(0) = 2.2$ and $\sigma_w^2 = 0.7185$ it follows that

$$A^2 = 2.963$$

In spite of the mathematical elegance of the Pisarenko harmonic decomposition, it is not commonly used in practice. One of the reasons for this is that it requires that the number of complex exponentials be known. In the ideal case of exact autocorrelations, the number of complex exponentials may be determined by overestimating the number of exponentials and examining the multiplicity of the minimum eigenvalue. However, with estimated autocorrelations, the multiplicity rule will not work since the minimum eigenvalue will rarely have a multiplicity greater than one. Another limitation with the Pisarenko decomposition is that it assumes that the additive noise is white. In practice, this will generally not be the case and the frequency estimates will be biased. Although Pisarenko's method may be modified to account for nonwhite noise, it is necessary that the power spectrum of the additive noise be known [46].

From a computational point of view, the Pisarenko harmonic decomposition requires finding the minimum eigenvalue and eigenvector of the signal autocorrelation matrix. For high-order problems, this may be computationally time consuming. It is possible, however, to find the minimum eigenvalue and eigenvector efficiently using an iterative algorithm based on the Levinson-Durbin recursion [13,15,19].

8.6.3 MUSIC

In 1979, an improvement to the Pisarenko harmonic decomposition known as the MUltiple SIgnal Classification method (MUSIC) was presented by Schmidt [48]. Like Pisarenko's method, the MUSIC algorithm is a frequency estimation technique. To see how the MUSIC algorithm works, assume that $x(n)$ is a random process consisting of p complex exponentials

in white noise with a variance of σ_w^2, and let $\mathbf{R}_x$ be the $M \times M$ autocorrelation matrix of $x(n)$ with $M > p + 1$.[19] If the eigenvalues of $\mathbf{R}_x$ are arranged in decreasing order, $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_M$, and if $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_M$ are the corresponding eigenvectors, then we may divide these eigenvectors into two groups: the p signal eigenvectors corresponding to the p largest eigenvalues, and the $M - p$ noise eigenvectors that, ideally, have eigenvalues equal to σ_w^2. However, with inexact autocorrelations the smallest $M - p$ eigenvalues will only be approximately equal to σ_w^2. Although we could consider estimating the white noise variance by averaging the $M - p$ smallest eigenvalues

$$\hat{\sigma}_w^2 = \frac{1}{M-p} \sum_{k=p+1}^{M} \lambda_k \tag{8.161}$$

estimating the frequencies of the complex exponentials is a bit more difficult. Since the eigenvectors of $\mathbf{R}_x$ are of length M, each of the noise subspace eigenfilters

$$V_i(z) = \sum_{k=0}^{M-1} v_i(k) z^{-k} \quad ; \quad i = p+1, \ldots, M$$

will have $M - 1$ roots (zeros). Ideally, p of these roots will lie on the unit circle at the frequencies of the complex exponentials, and the eigenspectrum

$$|V_i(e^{j\omega})|^2 = \frac{1}{\left| \sum_{k=0}^{M-1} v_i(k) e^{-jk\omega} \right|^2}$$

associated with the noise eigenvector $\mathbf{v}_i$ will exhibit sharp peaks at the frequencies of the complex exponentials. However, the remaining $(M - p - 1)$ zeros may lie anywhere and, in fact, some may lie close to the unit circle, giving rise to spurious peaks in the eigenspectrum. Furthermore, with inexact autocorrelations, the zeros of $V_i(z)$ that are on the unit circle may not remain on the unit circle. Therefore, when only one noise eigenvector is used to estimate the complex exponential frequencies, there may be some ambiguity in distinguishing the desired peaks from the spurious ones. As we saw in Example 8.6.2, for one complex exponential in white noise, the spurious peaks that are introduced from each of the noise subspace eigenfilters tend to occur at different frequencies. Therefore, in the MUSIC algorithm, the effects of these spurious peaks are reduced by averaging, using the frequency estimation function

$$\boxed{\hat{P}_{MU}(e^{j\omega}) = \frac{1}{\sum_{i=p+1}^{M} |\mathbf{e}^H \mathbf{v}_i|^2}} \tag{8.162}$$

The frequencies of the complex exponentials are then taken as the locations of the p largest peaks in $\hat{P}_{MU}(e^{j\omega})$. Once the frequencies have been determined the power of each complex exponential may be found using Eq. (8.160).

Instead of searching for the peaks of $\hat{P}_{MU}(e^{j\omega})$, an alternative is to use a method called root MUSIC, which involves rooting a polynomial. Since the z-transform equivalent of

[19] Recall that in the Pisarenko decomposition, $M = p + 1$. Thus, here we are using additional autocorrelations. As we will see, if $M = p + 1$, then the MUSIC algorithm is equivalent to Pisarenko's method.

Eq. (8.162) is

$$\hat{P}_{MU}(z) = \frac{1}{\sum_{i=p+1}^{M} V_i(z)V_i^*(1/z^*)}$$

then the frequency estimates may be taken to be the angles of the p roots of the polynomial

$$D(z) = \sum_{i=p+1}^{M} V_i(z)V_i^*(1/z^*) \tag{8.163}$$

that are closest to the unit circle. A MATLAB program for the MUSIC algorithm is given in Fig 8.34.

8.6.4 Other Eigenvector Methods

In addition to the Pisarenko and Music algorithms, a number of other eigenvector methods have been proposed for estimating the frequencies of complex exponentials in noise. One of these, the EigenVector (EV) method [20], is closely related to the MUSIC algorithm. Specifically, the EV method estimates the exponential frequencies from the peaks of the eigenspectrum

$$\hat{P}_{EV}(e^{j\omega}) = \frac{1}{\sum_{i=p+1}^{M} \frac{1}{\lambda_i}|\mathbf{e}^H\mathbf{v}_i|^2} \tag{8.164}$$

where λ_i is the eigenvalue associated with the eigenvector $\mathbf{v}_i$. Note that if $w(n)$ is white noise and if the autocorrelation sequence $r_x(k)$ is known exactly for $k = 0, 1, \ldots, M-1$,

The MUSIC Algorithm

```
function Px = music(x,p,M)
%
   x   = x(:);
   if M<p+1 | length(x)<M, error('Size of R is inappropriate'), end
   R = covar(x,M);
   [v,d]=eig(R);
   [y,i]=sort(diag(d));
   Px=0;
   for j=1:M-p
       Px=Px+abs(fft(v(:,i(j)),1024));
       end;
   Px=-20*log10(Px);
   end;
```

Figure 8.34 *A* MATLAB *program for estimating the frequencies of p complex exponentials in white noise using the MUSIC algorithm.*

then the eigenvalues in Eq. (8.164) are equal to the white noise variance, $\lambda_i = \sigma_v^2$, and the EV eigenspectrum will be the same as the MUSIC pseudospectrum to within a constant. However, with estimated autocorrelations the eigenvector method differs from the MUSIC algorithm and appears to produce fewer spurious peaks [20,34]. A MATLAB program for finding the eigenspectrum using the eigenvector method is given in Fig 8.35.

Another eigendecomposition-based method of interest is the *minimum norm algorithm* [29]. Instead of forming an eigenspectrum that uses all of the noise eigenvectors as in the MUSIC and eigenvector algorithms, the minimum norm algorithm uses a single vector $\mathbf{a}$ that is constrained to lie in the noise subspace, and the complex exponential frequencies are estimated from the peaks of the frequency estimation function

$$\hat{P}_{MN}(e^{j\omega}) = \frac{1}{|\mathbf{e}^H\mathbf{a}|^2} \tag{8.165}$$

With $\mathbf{a}$ constrained to lie in the noise subspace, if the autocorrelation sequence is known exactly, then $|\mathbf{e}^H\mathbf{a}|^2$ will have nulls at the frequencies of each complex exponential. Therefore, the z-transform of the coefficients in $\mathbf{a}$ may be factored as follows:

$$A(z) = \sum_{k=0}^{M-1} a(k)z^{-k} = \prod_{k=1}^{p}(1 - e^{j\omega_k}z^{-1}) \prod_{k=p+1}^{M-1}(1 - z_k z^{-1})$$

where z_k for $k = p+1, \ldots, M-1$ are the spurious roots that do not, in general, lie on the unit circle. The problem then is to determine which vector in the noise subspace minimizes the effects of the spurious zeros on the peaks of $\hat{P}_{MN}(e^{j\omega})$. The approach that is used in the minimum norm algorithm is to find the vector $\mathbf{a}$ that satisfies the following three constraints:

1. The vector $\mathbf{a}$ lies in the noise subspace.
2. The vector $\mathbf{a}$ has minimum norm.
3. The first element of $\mathbf{a}$ is unity.

The first constraint ensures that p roots of $A(z)$ lie on the unit circle. The second constraint ensures that the spurious roots of $A(z)$ lie *inside* the unit circle, i.e., $|z_k| < 1$. Finally, the third constraint ensures that the minimum norm solution is not the zero vector.

The Eigenvector Method

```
function Px = ev(x,p,M)
%
   x   = x(:);
   if M<p+1, error('Specified size of R is too small'), end
   R=covar(x,M);
   [v,d]=eig(R);
   [y,i]=sort(diag(d));
   Px=0;
   for j=1:M-p
       Px=Px+abs(fft(v(:,i(j)),1024)).^2/abs(y(j));
       end;
   Px=-10*log10(Px);
   end;
```

Figure 8.35 *A* MATLAB *program for estimating the frequencies of p complex exponentials in white noise using the eigenvector method.*

To solve this constrained minimization problem, we begin by noting that the constraint that $\mathbf{a}$ lies in the noise subspace may be written as

$$\mathbf{a} = \mathbf{P}_n\mathbf{v} \tag{8.166}$$

where $\mathbf{P}_n = \mathbf{V}_n\mathbf{V}_n^H$ is the projection matrix that projects an arbitrary vector $\mathbf{v}$ onto the noise subspace (see Eq. (8.151) and p. 34). The third constraint may be expressed as follows:

$$\mathbf{a}^H\mathbf{u}_1 = 1 \tag{8.167}$$

where $\mathbf{u}_1 = [1, 0, \ldots, 0]^T$. This constraint may be combined with the constraint given in Eq. (8.166) as follows,

$$\mathbf{v}^H\left(\mathbf{P}_n^H\mathbf{u}_1\right) = 1 \tag{8.168}$$

Using Eq. (8.166) the norm of $\mathbf{a}$ may be written as

$$\|\mathbf{a}\|^2 = \|\mathbf{P}_n\mathbf{v}\|^2 = \mathbf{v}^H\left(\mathbf{P}_n^H\mathbf{P}_n\right)\mathbf{v}$$

Since $\mathbf{P}_n$ is a projection matrix then it is Hermitian, $\mathbf{P}_n = \mathbf{P}_n^H$, and idempotent, $\mathbf{P}_n^2 = \mathbf{P}_n$. Therefore,

$$\|\mathbf{a}\|^2 = \mathbf{v}^H\mathbf{P}_n\mathbf{v}$$

and it follows that minimizing the norm of $\mathbf{a}$ is equivalent to finding the vector $\mathbf{v}$ that minimizes the quadratic form $\mathbf{v}^H\mathbf{P}_n\mathbf{v}$. We may now reformulate the constrained minimization problem as follows:

$$\boxed{\min \mathbf{v}^H\mathbf{P}_n\mathbf{v} \quad \text{subject to} \quad \mathbf{v}^H\left(\mathbf{P}_n^H\mathbf{u}_1\right) = 1} \tag{8.169}$$

Once the solution to Eq. (8.169) has been found, the minimum norm solution is formed by projecting $\mathbf{v}$ onto the noise subspace using Eq. (8.166).

In Section 2.3.10 we considered the problem of solving constrained minimization problems of the form given in Eq. (8.169). What we found was that the solution is

$$\mathbf{v} = \lambda\mathbf{P}_n^{-1}\left(\mathbf{P}_n^H\mathbf{u}_1\right) = \lambda\mathbf{u}_1$$

where

$$\lambda = \frac{1}{\mathbf{u}_1^H\mathbf{P}_n\mathbf{u}_1}$$

Therefore, the minimum norm solution is

$$\boxed{\mathbf{a} = \mathbf{P}_n\mathbf{v} = \lambda\mathbf{P}_n\mathbf{u}_1 = \frac{\mathbf{P}_n\mathbf{u}_1}{\mathbf{u}_1^H\mathbf{P}_n\mathbf{u}_1}} \tag{8.170}$$

which is simply the projection of the unit vector onto the noise subspace, normalized so that the first coefficient is equal to one. In terms of the eigenvectors of the autocorrelation matrix, the minimum norm solution may be written as

$$\boxed{\mathbf{a} = \frac{(\mathbf{V}_n\mathbf{V}_n^H)\mathbf{u}_1}{\mathbf{u}_1^H(\mathbf{V}_n\mathbf{V}_n^H)\mathbf{u}_1}} \tag{8.171}$$

A MATLAB program for the minimum norm algorithm is given in Fig 8.36. The frequency estimation algorithms are summarized in Table 8.10.

The Minimum Norm Algorithm

```
function Px = min_norm(x,p,M)
%
   x   = x(:);
   if M<p+1, error('Specified size of R is too small'), end
   R=covar(x,M);
   [v,d]=eig(R);
   [y,i]=sort(diag(d));
   for j=1:M-p
       V=[V,v(:,i(j))];
       end;
   a=V*V(1,:)';
   Px=-20*log10(abs(fft(a,1024)));
   end;
```

Figure 8.36 *A* MATLAB *program for estimating the frequencies of p complex exponentials in white noise using the minimum norm algorithm.*

Table 8.10 *Noise Subspace Methods for Frequency Estimation*

Pisarenko	$\hat{P}_{PHD}(e^{j\omega}) = \dfrac{1}{\lvert\mathbf{e}^H\mathbf{v}_{\min}\rvert^2}$
MUSIC	$\hat{P}_{MU}(e^{j\omega}) = \dfrac{1}{\sum_{i=p+1}^{M}\lvert\mathbf{e}^H\mathbf{v}_i\rvert^2}$
Eigenvector Method	$\hat{P}_{EV}(e^{j\omega}) = \dfrac{1}{\sum_{i=p+1}^{M}\frac{1}{\lambda_i}\lvert\mathbf{e}^H\mathbf{v}_i\rvert^2}$
Minimum Norm	$\hat{P}_{MN}(e^{j\omega}) = \dfrac{1}{\lvert\mathbf{e}^H\mathbf{a}\rvert^2}$; $\mathbf{a} = \lambda\mathbf{P}_n\mathbf{u}_1$

Example 8.6.5 *A Comparison of Frequency Estimation Methods*

Let $x(n)$ be a process consisting of a sum of four complex exponentials in white noise

$$x(n) = \sum_{k=1}^{4} A_k e^{j(n\omega_k+\phi_k)} + w(n)$$

where the amplitudes A_k are equal to one, the frequencies ω_k are 0.2π, 0.3π, 0.8π, and 1.2π, the phases are uncorrelated random variables that are uniformly distributed over the interval $[0, 2\pi]$, and the variance of the white noise is $\sigma_w^2 = 0.5$. Using ten different realizations of $x(n)$ with $N = 64$ values, overlay plots of the frequency estimation functions using Pisarenko's method, the MUSIC algorithm, the eigenvector method, and the minimum norm algorithm are shown in Fig. 8.37. For Pisarenko's method, the frequency estimation

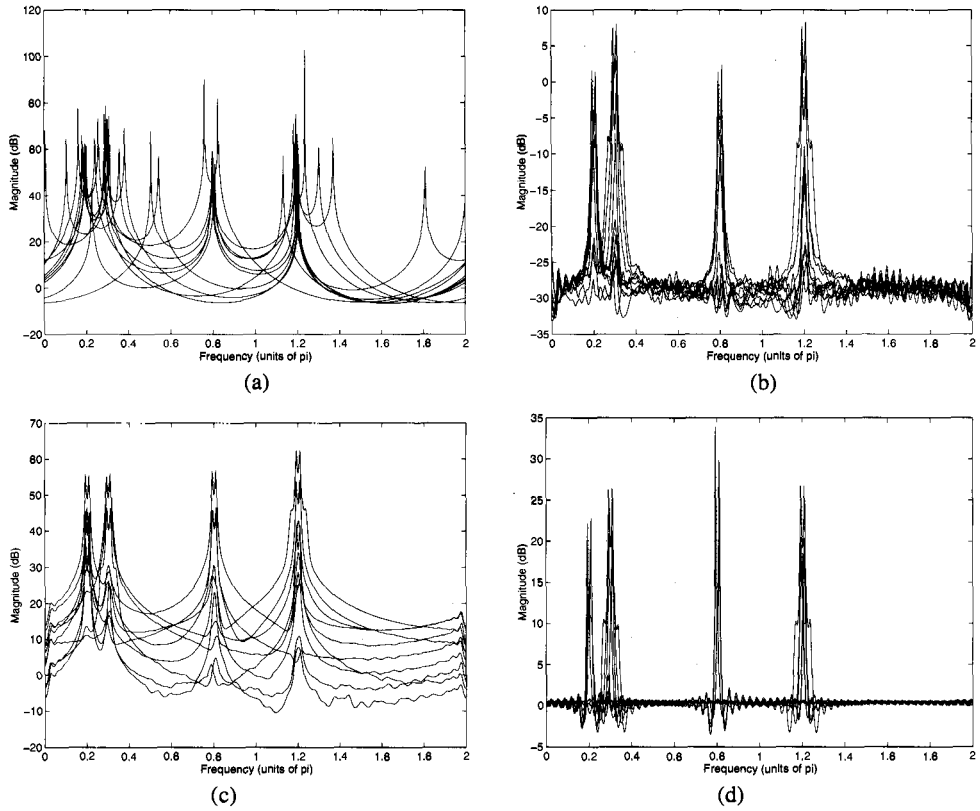


Figure 8.37 *The frequency estimation functions for a process consisting of four complex exponentials in white noise using (a) the Pisarenko harmonic decomposition, (b) the MUSIC algorithm, (c) the eigenvector method and (d) the minimum norm algorithm.*

function was derived from the 5 × 5 autocorrelation matrix that was estimated from the 64 values of $x(n)$. For the MUSIC, eigenvector, and minimum norm algorithms, the frequency estimation functions were formed from the 64 × 64 autocorrelation matrix that was estimated from $x(n)$, i.e., $M = 64$. Except for Pisarenko's method, the frequency estimation functions for this process produce accurate estimates of the exponential frequencies, with the most well-defined peaks being produced with the minimum norm algorithm. However, it is important to point out that not all of the frequency estimation functions shown in these overlay plots have four well-defined peaks. In some cases, for example, only two or three peaks are observed.

8.7 PRINCIPAL COMPONENTS SPECTRUM ESTIMATION

In the previous section, we saw how the orthogonality of the signal and noise subspaces could be used to estimate the frequencies of p complex exponentials in white noise. Since these methods only use vectors that lie in the noise subspace, they are often referred to as *noise subspace methods*. In this section, we consider another set of algorithms that use vectors that lie in the signal subspace. These methods are based on a principal components

analysis of the autocorrelation matrix and are referred to as *signal subspace methods*. The basic idea of these methods is as follows. Let $\mathbf{R}_x$ be an $M \times M$ autocorrelation matrix of a process that consists of p complex exponentials in white noise. With an eigendecomposition of $\mathbf{R}_x$ we have

$$\mathbf{R}_x = \sum_{i=1}^{M} \lambda_i \mathbf{v}_i \mathbf{v}_i^H = \sum_{i=1}^{p} \lambda_i \mathbf{v}_i \mathbf{v}_i^H + \sum_{i=p+1}^{M} \lambda_i \mathbf{v}_i \mathbf{v}_i^H \tag{8.172}$$

where it is assumed that the eigenvalues have been arranged in decreasing order, $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_M$. Since the second term in Eq. (8.172) is due only to the noise, we may form a *reduced rank* approximation to the signal autocorrelation matrix, $\mathbf{R}_s$, by retaining only the principal eigenvectors of $\mathbf{R}_x$,

$$\hat{\mathbf{R}}_s = \sum_{i=1}^{p} \lambda_i \mathbf{v}_i \mathbf{v}_i^H \tag{8.173}$$

This principal components approximation may then be used in the place of $\mathbf{R}_x$ in a spectral estimator such as the minimum variance method or the maximum entropy method. The net effect of this approach is to *filter out* a portion of the noise, thereby enhancing the estimate of the spectral component due to the signal alone, i.e., the complex exponentials. Another way to view this approach is in terms of a constraint that is being placed on the autocorrelation matrix. Specifically, given that a process consists of p complex exponentials in noise, since the rank of the autocorrelation matrix due to the signal, $\mathbf{R}_s$, is p, then a principal components representation simply imposes this rank-p constraint on $\mathbf{R}_x$. In the following subsections, we discuss how a principal components analysis of the autocorrelation matrix may be used in conjunction with the Blackman-Tukey method, the minimum variance method, and the maximum entropy method to form a principal components spectrum estimate.

8.7.1 Blackman-Tukey Frequency Estimation

The Blackman-Tukey estimate of the power spectrum is formed by taking the discrete-time Fourier transform of a windowed autocorrelation sequence

$$\hat{P}_{BT}(e^{j\omega}) = \sum_{k=-M}^{M} \hat{r}_x(k) w(k) e^{-jk\omega}$$

If $w(k)$ is a Bartlett window, then the Blackman-Tukey estimate may be written in terms of the autocorrelation matrix $\mathbf{R}_x$ as follows:

$$\hat{P}_{BT}(e^{j\omega}) = \frac{1}{M} \sum_{k=-M}^{M} (M - |k|) \hat{r}_x(k) e^{-jk\omega} = \frac{1}{M} \mathbf{e}^H \mathbf{R}_x \mathbf{e} \tag{8.174}$$

With an eigendecomposition of the autocorrelation matrix, $\hat{P}_{BT}(e^{j\omega})$ may therefore be expressed as

$$\hat{P}_{BT}(e^{j\omega}) = \frac{1}{M} \sum_{i=1}^{M} \lambda_i |\mathbf{e}^H \mathbf{v}_i|^2$$

If $x(n)$ is known to consist of p complex exponentials in white noise, and if the eigenvalues of $\mathbf{R}_x$ are arranged in decreasing order, $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_M$, then a principal components

version of this spectrum estimate is

$$\hat{P}_{PC-BT}(e^{j\omega}) = \frac{1}{M}\mathbf{e}^H\hat{\mathbf{R}}_x\mathbf{e} = \frac{1}{M}\sum_{i=1}^{p}\lambda_i|\mathbf{e}^H\mathbf{v}_i|^2 \tag{8.175}$$

A MATLAB program for the principal components Blackman-Tukey spectrum estimate is given in Fig. 8.38.

8.7.2 Minimum Variance Frequency Estimation

Given the autocorrelation sequence $r_x(k)$ of a process $x(n)$ for lags $|k| \leq M$, the Mth-order minimum variance spectrum estimate is

$$\hat{P}_{MV}(e^{j\omega}) = \frac{M}{\mathbf{e}^H\mathbf{R}_x^{-1}\mathbf{e}} \tag{8.176}$$

With an eigendecomposition of the autocorrelation matrix, the inverse of $\mathbf{R}_x$ is

$$\mathbf{R}_x^{-1} = \sum_{i=1}^{p}\frac{1}{\lambda_i}\mathbf{v}_i\mathbf{v}_i^H + \sum_{i=p+1}^{M}\frac{1}{\lambda_i}\mathbf{v}_i\mathbf{v}_i^H \tag{8.177}$$

where p is the number of complex exponentials. Retaining only the first p principal components of $\mathbf{R}_x^{-1}$ leads to the principal components minimum variance estimate

$$\hat{P}_{PC-MV}(e^{j\omega}) = \frac{M}{\displaystyle\sum_{i=1}^{p}\frac{1}{\lambda_i}|\mathbf{e}^H\mathbf{v}_i|^2} \tag{8.178}$$

It is interesting to compare Eq. (8.178) with the eigenvector method given in Eq. (8.164).

Principal Components Frequency Estimation Using the Blackman-Tukey Method

```
function Px = bt_pc(x,p,M)
%
   x   = x(:);
   if M<p+1, error('Specified size of R is too small'), end
   R=covar(x,M);
   [v,d]=eig(R);
   [y,i]=sort(diag(d));
   Px=0;
   for j=M-p+1:M
       Px=Px+abs(fft(v(:,i(j)),1024))*sqrt(real(y(j)));
       end;
   Px=20*log10(Px)-10*log10(M);
   end;
```

Figure 8.38 *A* MATLAB *program for estimating the frequencies of p complex exponentials in white noise using a principal components analysis with the Blackman-Tukey method.*

Whereas $\hat{P}_{PC-MV}(e^{j\omega})$ is based on the first term in the decomposition given in Eq. (8.177), the EV method uses the second. Recall, however, that the EV algorithm produces a frequency estimation function rather than an estimate of the spectrum. Therefore, whereas $\hat{P}_{EV}(e^{j\omega})$ simply produces peaks in the pseudospectrum at the complex exponential frequencies, the minimum variance method provides an estimate of the power spectrum.

8.7.3 Autoregressive Frequency Estimation

Autoregressive spectrum estimation using the autocorrelation, covariance, or modified covariance algorithms involves finding the solution to a set of linear equations of the form

$$\mathbf{R}_x \mathbf{a}_M = \epsilon_M \mathbf{u}_1 \tag{8.179}$$

where $\mathbf{R}_x$ is an $(M+1) \times (M+1)$ autocorrelation matrix. From the solution to these equations,

$$\mathbf{a}_M = \epsilon_M \mathbf{R}_x^{-1} \mathbf{u}_1$$

an estimate of the spectrum is formed as follows:

$$\hat{P}_{AR}(e^{j\omega}) = \frac{|b(0)|^2}{|\mathbf{e}^H \mathbf{a}_M|^2}$$

where $b(0)$ is some appropriately chosen constant such as $|b(0)|^2 = \epsilon_M$. However, if $x(n)$ is known to consist of p complex exponentials in noise, then we may form a principal components solution to Eq. (8.179) as follows:

$$\mathbf{a}_{pc} = \epsilon_M \left(\sum_{i=1}^{p} \frac{1}{\lambda_i} \mathbf{v}_i \mathbf{v}_i^H \right) \mathbf{u}_1$$

or,

$$\mathbf{a}_{pc} = \epsilon_M \sum_{i=1}^{p} \frac{v_i^*(0)}{\lambda_i} \mathbf{v}_i = \epsilon_M \sum_{i=1}^{p} \alpha_i \mathbf{v}_i$$

where $v_i(0)$ is the first element of the normalized eigenvector $\mathbf{v}_i$ and $\alpha_i = v_i^*(0)/\lambda_i$. Therefore, if we set $|b(0)|^2 = \epsilon_M$, then the principal components autoregressive spectrum estimate becomes

$$\hat{P}_{PC-AR}(e^{j\omega}) = \frac{1}{\left| \sum_{i=1}^{p} \alpha_i \mathbf{e}^H \mathbf{v}_i \right|^2}$$

Note that as the number of autocorrelations is increased, only p principal eigenvectors and eigenvalues are used in $\hat{P}_{PC-AR}(e^{j\omega})$. This allows for an increase in the model order without a corresponding increase in the spurious peaks that are due to the noise eigenvectors of the autocorrelation matrix.

We conclude this section with a summary of the signal subspace frequency estimation algorithms, which is given in Table 8.11. As we have seen, each technique is based on using the p principal eigenvectors of the autocorrelation matrix. The techniques differ, however, in the weighting function that is applied to the eigenfilters $\mathbf{e}^H \mathbf{v}_k$ and in whether the linear combination of weighted eigenfilters is in the numerator or denominator of the spectrum estimate.

Table 8.11 ***Signal Subspace Methods***

Blackman-Tukey	$\hat{P}_{PC-BT}(e^{j\omega}) = \frac{1}{M}\sum_{i=1}^{p}\lambda_i \lvert\mathbf{e}^H\mathbf{v}_i\rvert^2$
Minimum variance	$\hat{P}_{PC-MV}(e^{j\omega}) = \frac{M}{\sum_{i=1}^{p}\frac{1}{\lambda_i}\lvert\mathbf{e}^H\mathbf{v}_i\rvert^2}$
Autoregressive	$\hat{P}_{PC-AR}(e^{j\omega}) = \frac{1}{\left\lvert\sum_{i=1}^{p}\alpha_i\mathbf{e}^H\mathbf{v}_i\right\rvert^2}$

8.8 SUMMARY

In this chapter, we considered many different approaches for estimating the power spectrum of a wide-sense stationary process. We began with a discussion of the nonparametric methods that are based on computing the discrete-time Fourier transform of an estimate of the autocorrelation sequence. The first of these was the periodogram, which is easily evaluated from the DFT of the given values of the process. Unfortunately, however, the periodogram is not a consistent estimate of the power spectrum. Therefore, we looked at several modifications of the periodogram to improve its statistical properties. These included applying a window to the data, periodogram averaging, and periodogram smoothing. Although periodogram averaging and periodogram smoothing provide a consistent estimate of the power spectrum, they generally do not work well for short data records, and are limited in their ability to resolve closely spaced narrowband processes when the number of data samples is limited. An advantage of these methods, on the other hand, is that they do not make any assumptions or place any constraints on the process and, therefore, may be used on any type of process.

Next, we derived the minimum variance method, which may be viewed as a data-adaptive modification to the periodogram. The basic idea of this approach is to design a filter bank of bandpass filters, measure the power in the processes that are produced at the output of each filter, and estimate the power spectrum by dividing this power estimate by the bandwidth of the filter. Generally, the minimum variance spectrum estimate provides higher resolution than the periodogram and Blackman-Tukey methods.

The primary limitation with the nonparametric methods of spectrum estimation is that the estimate of the spectrum is based on a windowed autocorrelation sequence. In an attempt to overcome this limitation and improve the resolution, we then derived the maximum entropy method, which estimates the spectrum using a maximum entropy extrapolation of a given partial autocorrelation sequence. In other words, given $r_x(k)$ for $|k| \leq p$, the values of $r_x(k)$ for $|k| > p$ are found that make the underlying process as white or as random as possible. What we discovered, however, is that this is equivalent to finding an all-pole model for the process that is consistent with the given autocorrelation sequence, and then computing the power spectrum from the all-pole model.

The next set of techniques that were discussed are the parametric methods of spectrum estimation. With a parametric approach, the first step is to select an appropriate model for the process. This selection may be based on a priori knowledge about how the process is generated or, perhaps, on experimental results indicating that a particular model "works well". Once a model has been selected, the next step is to estimate the model parameters from the given data. For example, if $x(n)$ is assumed to be an autoregressive process, then the covariance method or some other algorithm may be used to estimate the all-pole parameters. The final step is to estimate the power spectrum by incorporating the estimated parameters into the parametric form for the spectrum. Although it is possible to significantly improve the performance of the spectrum estimate with a parametric approach, it is important that the model that is used be consistent with the process that is being analyzed. Otherwise inaccurate or misleading spectrum estimates may result.

The final set of techniques that were discussed are those that assume a harmonic model for the process, i.e., that $x(n)$ is a sum of complex exponentials or sinusoids in white noise. For these processes, the goal is generally to estimate the frequencies of the complex exponentials and, possibly, to determine the powers. Two different approaches to the problem of frequency estimation were considered. The first involves defining a frequency estimation function that produces peaks at the frequencies of the complex exponentials. These frequency estimation functions are designed to take advantage of the fact that the signal and noise subspaces are orthogonal. The second set of approaches use a principal components analysis of the autocorrelation matrix. The resulting reduced-rank approximation to the autocorrelation matrix is then used in a spectrum estimation technique such as the minimum variance method or the maximum entropy method.

Although many different approaches to spectrum estimation have been presented in this chapter, the list is by no means complete. Many other methods have been proposed and, under certain conditions or set of assumptions, these approaches may be superior to the methods described here. For a more complete coverage of spectrum estimation, the reader may consult any one of a number of excellent books on the subject [10,16,25,28,34].

References

1. H. Akaike, "Fitting autoregressive models for prediction," *Ann. Inst. Statist. Math*, vol. 21, pp. 243–247, 1969.
2. H. Akaike, "A new look at the statistical model identification," *IEEE Trans. Autom. Control*, vol. AC-19, pp. 716–723, Dec. 1974.
3. N. O. Anderson, "Comments on the performance of the maximum entropy algorithm," *Proc. IEEE*, vol. 66, pp. 1581–1582, Nov. 1978.
4. R. Ash, *Information Theory*, Wiley Interscience, New York, 1965.
5. M. S. Bartlett, "Smoothing periodograms from time series with continuous spectra," *Nature* (London), vol. 161, pp. 686–687, May 1948.
6. R. B. Blackman and J. W. Tukey, *The Measurement of Power Spectra*, Dover, New York, 1958.
7. J. P. Burg, "Maximum entropy spectral analysis," Ph.D. thesis, Stanford University, Stanford, CA., May 1975.
8. J. Capon, "High-resolution frequency-wavenumber spectrum analysis," *Proc. IEEE*, vol. 57, pp. 1408–1418, Aug. 1969.
9. W. Y. Chen and G. R. Stegen, "Experiments with maximum entropy power spectra of sinusoids," *J. Geophysics Rev.*, vol. 79, pp. 3019–3022, July 1974.
10. D. G. Childers, Ed., *Modern Spectrum Analysis*, IEEE Press, New York, 1978.

11. T. M. Cover and J. A. Thomas, *Elements of Information Theory*, John Wiley & Sons, Inc., New York, 1991.
12. R. E. Davis and L. A. Regier, "Methods of estimating directional wave spectra from multi- element arrays," *Journal of Marine Research*, vol. 35, pp. 453–477, 1977.
13. C. Gueguen, Y. Grenier, and F. Giannella, "Factorial linear modeling, algorithms, and applications," *Proc. 1980 Int. Conf. Acoust., Speech, Signal Proc.*, pp. 618–621, 1980.
14. F. J. Harris, "On the use of windows for harmonic analysis with the discrete Fourier transform," *Proc. IEEE*, vol. 66, pp. 51–83, Jan. 1978.
15. M. H. Hayes and M. A. Clements, "An efficient algorithm for computing Pisarenko's harmonic decomposition using Levinson's recursion," *IEEE Trans. on Acoustics, Speech, Sig. Proc.*, vol. ASSP-34, no. 3, pp. 485–491, June 1986.
16. S. Haykin, Ed., *Nonlinear Methods of Spectral Analysis*, Springer-Verlag, New York, 1979.
17. S. Haykin, *Adaptive Filter Theory*, Prentice-Hall, Englewood Cliffs, NJ, 1986.
18. S. Haykin and S. Kesler, "Prediction-error filtering and maximum-entropy spectral estimation," Chapter 2 in *Nonlinear Methods of Spectral Analysis*, S. Haykin, Ed., Springer-Verlag, New York, 1979.
19. Y. Hu and S. Y. Kung, "Highly concurrent Toeplitz eigen-system solver for high-resolution spectral estimation," *Proc. 1983 Int. Conf. Acoust., Speech, Signal Proc.*, pp. 1422–1425, 1983.
20. D. H. Johnson and S. R. DeGraaf, "Improving the resolution of bearing in passive sonar arrays by eigenvalue analysis," *IEEE Trans. Acoust., Speech, Sig. Proc.*, vol. ASSP-30, no. 4, pp. 638–647, Aug. 1982.
21. R. H. Jones, "Autoregression order selection," *Geophysics*, vol. 41, pp. 771–773, Aug. 1976.
22. R. H. Jones, "Identification and autoregressive spectrum estimation," *IEEE Trans. Autom. Control*, vol. AC-19, pp. 894–897, Dec. 1974.
23. R. L. Kashyap, "Inconsistency of the AIC rule for estimating the order of autoregressive models," *IEEE Trans. Autom. Control*, vol. AC-25, pp. 996–998, Oct. 1980.
24. M. Kaveh and S. P. Bruzzone, "Order determination for autoregressive spectral estimation," *Record of the 1979 RADC Spectral Estimation Workshop*, pp. 139–145, 1979.
25. S. Kay, *Modern Spectrum Estimation: Theory and Applications*, Prentice-Hall, Englewood Cliffs, NJ, 1988.
26. S. Kay, "Noise compensation for autoregressive spectral estimates," *IEEE Trans. Acoust., Speech, Sig. Proc.*, vol. ASSP-28, pp. 292–303, June 1980.
27. S. Kay and S. L. Marple Jr., "Sources and remedies for spectral line splitting in autoregressive spectrum analysis," *Proc. 1979 Int. Conf. on Acoust., Speech, Sig. Proc.*, pp. 151–154, 1979.
28. S. B. Kesler, Ed., *Modern Spectrum Analysis, II*, IEEE Press, New York, 1986.
29. R. Kumaresan and D. W. Tufts, "Estimating the angles of arrival of multiple plane waves," *IEEE Trans. on Aerospace and Elec. Syst.*, vol. AES-19, vol. 1, pp. 134–139, Jan. 1983.
30. R. T. Lacoss, "Data adaptive spectral analysis methods," *Geophysics*, vol. 36, pp. 661–675, Aug. 1971.
31. M. A. Lagunas and A. Gasull-Llampalas, "An improved maximum likelihood method for power spectral density estimation," *IEEE Trans. Acoust., Speech, Sig. Proc.*, vol. ASSP-32, no. 1, pp. 170–172, Feb. 1984.
32. T. E. Landers and R. T. Lacoss, "Some geophysical applications of autoregressive spectral estimates," *IEEE Trans. Geosci. Electron.*, vol. GE-15, pp. 26–32, Jan. 1977.
33. H. L. Larson and B. O. Shubert, *Probabilistic Models in Engineering Sciences: Vol. I, Random Variables and Stochastic Processes*, John Wiley & Sons, New York, 1979.
34. S. L. Marple Jr., *Digital Spectral Analysis with Applications*, Prentice-Hall, Englewood Cliffs, NJ, 1987.
35. S. L. Marple Jr., "A new autoregressive spectrum analysis algorithm," *IEEE Trans. Acoust., Speech, Sig. Proc.*, vol. ASSP-28, pp. 441–454, Aug. 1980.

36. T. L. Marzetta and S. W. Lang, "New interpretations for the MLM and DASE spectral estimators," *Proc. 1983 Int. Conf. Acoust., Speech, Sig. Proc.*, pp. 844–846, April 1983.

37. A. H. Nuttall, "Spectral analysis of a univariate process with bad data points via maximum entropy and linear predictive techniques," Tech. Rep. TR-5303, Naval Underwater Systems Center, New London, CT., March 1976.

38. A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1989.

39. E. Parzen, "Some recent advances in time series modeling," *IEEE Trans. Autom. Control*, vol. AC-19, pp. 723–730, Dec. 1974.

40. A. Papoulis, *Probability, Random Variables, and Stochastic Processes*, McGraw-Hill, New York, 1984.

41. V. F. Pisarenko, "The retrieval of harmonics from a covariance function," *Geophysics J. Roy. Astron. Soc.*, vol. 33, pp. 347–366, 1973.

42. R. J. McAulay and T. F. Quatieri, "Speech analysis/synthesis based on a sinusoidal representation," *IEEE Trans. Acoust., Speech, Sig. Proc.*, vol. ASSP-34, no. 4, pp. 744–754, Aug. 1986.

43. L. R. Rabiner and R. W. Schafer, *Digital Processing of Speech Signals*, Prentice-Hall, Englewood Cliffs, NJ, 1978

44. I. S. Reed, "On a moment theorem for complex Gaussian processes," *Proc. IEEE Trans. Inf. Theory*, vol. IT-8, pp. 194–195, April 1962.

45. J. Rissanen, "Modeling by shortest data description," *Automatica*, vol. 14, pp. 465–471, 1978.

46. R .A. Roberts and C. T. Mullis, *Digital Signal Processing*, Addison Wesley, Reading, MA. 1987.

47. E. A. Robinson, "A historical perspective of spectrum estimation," *Proc. IEEE*, vol. 70, pp. 885–907, Sept., 1982.

48. R. Schmidt, "Multiple emitter location and signal parameter estimation," *Proc. RADC Spectrum Estimation Workshop*, pp. 243–258, 1979.

49. A. Schuster, "On the investigation of hidden periodicities with application to a supposed twenty-six day period of meteorological phenomena," *Terr. Magn.*, vol. 3, no. 1, pp. 13–41, March 1898.

50. G. Schwartz, "Estimating the dimension of a model," *Annals of Statistics*, vol. 6, pp. 461–464, 1978.

51. S. Shon and K. Mehrota, "Performance comparisons of autoregressive estimation methods," *Proc. 1984 Int. Conf. Acoust., Speech, Sig. Proc.*, pp. 14.3.1–14.3.4., 1984.

52. G. Strang, *Linear Algebra and its Applications*, Academic Press, New York, 1980.

53. D. N. Swingler, "A comparison between Burg's maximum entropy method and a non-recursive technique for the spectral analysis of deterministic signals," *J. Geophysics Res.*, vol. 84, pp. 679–685, Feb. 1979.

54. D. M. Thomas and M. H. Hayes, "A novel data-adaptive power spectrum estimation technique," *Proc. 1987 Int. Conf. on Acoustics, Speech, Sig. Proc.*, pp. 1589–1592, April 1987.

55. H. Tong, "Autoregressive model fitting with noisy data by Akaike's information criterion," *IEEE Trans. Inf. Theory*, vol. IT-21, pp. 476–480, July 1975.

56. T. J. Ulrych and M. Ooe, "Autoregressive and mixed autoregressive-moving average models," Chapter 3 in *Nonlinear Methods of Spectral Analysis*, S. Haykin, Ed., Springer-Verlag, New York, 1979.

57. T. J. Ulrych and R. W. Clayton, "Time series modeling and maximum entropy," *Phys. Earth Planet. Inter.*, vol. 12, pp. 188–200, Aug. 1976.

58. M. Wax and T. Kailath, "Detection of signals by information theoretic criteria," *IEEE Trans. Acoust., Speech, Sig. Proc.*, vol. ASSP-33, no. 2, pp. 387–392, April 1985.

59. P. D. Welch, "The use of fast Fourier transform for the estimation of power spectra: A method based on time averaging over short modified periodograms," *IEEE Trans. Audio and Electroacoust.*, vol. AU-15, pp. 70–73, June 1967.

8.9 PROBLEMS

8.1. Given $N = 10,000$ samples of a process $x(n)$, you are asked to compute the periodogram. However, with only a finite amount of memory resources, you are unable to compute a DFT any longer than 1024. Using these 10,000 samples, describe how you would be able to compute a periodogram that has a resolution of

$$\Delta\omega = 0.89\frac{2\pi}{10000}$$

Hint: Consider how the decimation-in-time FFT algorithm works.

8.2. A continuous-time signal $x_a(t)$ is bandlimited to 5 kHz, i.e., $x_a(t)$ has a spectrum $X_a(f)$ that is zero for $|f| > 5$ kHz. Only 10 seconds of the signal has been recorded and is available for processing. We would like to estimate the power spectrum of $x_a(t)$ using the available data in a radix-2 FFT algorithm, and it is required that the estimate have a resolution of at least 10 Hz. Suppose that we use Bartlett's method of periodogram averaging.

(a) If the data is sampled at the Nyquist rate, what is the minimum section length that you may use to get the desired resolution?

(b) Using the minimum section length determined in part (a), with 10 seconds of data, how many sections are available for averaging?

(c) How does your choice of the sampling rate affect the resolution and variance of your estimate? Are there any benefits to sampling above the Nyquist rate?

8.3. Bartlett's method is used to estimate the power spectrum of a process from a sequence of $N = 2000$ samples.

(a) What is the minimum length L that may be used for each sequence if we are to have a resolution of $\Delta f = 0.005$?

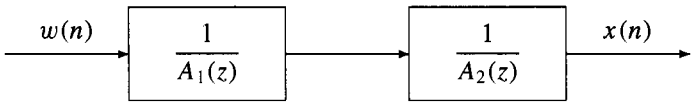
(b) Explain why it would not be advantageous to increase L beyond the value found in (a).

(c) The *quality factor* of a spectrum estimate is defined to be the inverse of the variability,

$$Q = 1/\mathcal{V}$$

Using Bartlett's method, what is the minimum number of data samples, N, that are necessary to achieve a resolution of $\Delta f = 0.005$, and a quality factor that is five times larger than that of the periodogram?

8.4. A random process $x(n)$ is generated by filtering unit variance white noise as shown in the figure below



where

$$A_1(z) = 1 + az^{-1} + 0.99z^{-2} \quad ; \quad A_2(z) = 1 - az^{-1} + 0.98z^{-2}$$

(a) Prepare a carefully labeled sketch of the power spectrum of $x(n)$ assuming that a is small, e.g., $0 < a < 0.1$. Pay careful attention to the location and amplitude of the two spectral peaks and the value of $P_x(e^{j\omega})$ at $\omega = \pi/2$.

(b) If $a = 0.1$, determine the section length L required to resolve the spectral peaks of $P_x(e^{j\omega})$ using Bartlett's method. For this value of L, find an approximate value for the bias of the estimate at the peaks of the spectrum. How is the bias related to the area of the spectral peaks?

(c) Consider the method of periodogram smoothing. How many lags of the autocorrelation must be used to obtain a resolution that is comparable to that of Bartlett's estimate considered in part (b)? How much data must be available if the variance of the estimate is to be comparable to that of a four-section Bartlett estimate?

8.5. Many commercial *Fourier analyzers* continuously update the estimate of the power spectrum of a process $x(n)$ by exponential averaging of periodograms as follows,

$$\hat{P}_i(e^{j\omega}) = \alpha \hat{P}_{i-1}(e^{j\omega}) + \frac{1-\alpha}{N}\left|\sum_{n=0}^{N-1} x_i(n)e^{-jn\omega}\right|^2$$

where $x_i(n) = x(n + Ni)$ is the ith sequence of N data values. This update equation is initialized with $\hat{P}_{-1}(e^{j\omega}) = 0$.

(a) Qualitatively describe the philosophy behind this method, and discuss how the value for the weighting factor α should be selected.

(b) Assuming that successive periodograms are uncorrelated and that $0 < \alpha < 1$, find the mean and variance of $\hat{P}_i(e^{j\omega})$ for a Gaussian random process.

(c) Repeat the analysis in part (b) if the periodograms are replaced with modified periodograms.

8.6. The minimum variance method of spectrum estimation constrains the bandpass filter $G_i(e^{j\omega})$ to have a gain of one at frequency $\omega = \omega_i$,

$$G_i(e^{j\omega_i}) = 1$$

Another approach is to constrain the filter to have unit energy over a frequency band that is centered at $\omega = \omega_i$ and has a bandwidth of Δ,

$$\frac{1}{\Delta}\int_{\omega_i-\Delta/2}^{\omega_i+\Delta/2} |G_i(e^{j\omega})|^2 \, d\omega = 1$$

With this constraint, the filter coefficients $\mathbf{g}_i = [g_i(0), g_i(1), \ldots, g_i(p)]^T$ that minimize the power in the filtered process,

$$E\{|y_i(n)|^2\} = \mathbf{g}_i^H \mathbf{R}_x \mathbf{g}_i$$

may be shown to be the solution to a generalized eigenvalue problem,

$$\mathbf{R}_x \mathbf{g}_i = \lambda(\omega_i, \Delta)\mathbf{T}\mathbf{g}_i$$

where $\mathbf{T}$ is a matrix whose elements depend upon ω_i and Δ. The spectrum estimate, referred to as the DASE estimate, is

$$\hat{P}_{DASE}(e^{j\omega}) = \lambda_{\min}(\omega_i, \Delta)$$

where $\lambda_{\min}(\omega_i, \Delta)$ is the minimum eigenvalue of the generalized eigenvalue problem.

(a) Perform the minimization of $E\{|y_i(n)|^2\}$ and determine the form of the matrix $\mathbf{T}$.

(b) What happens to the matrix $\mathbf{T}$ in the limit as $\Delta \rightarrow 0$? What does the power spectrum estimate correspond to in this case?

(c) Repeat part (b) for $\Delta = 2\pi$.

(d) Find the DASE estimate for white noise.

8.7. Let $x(n)$ be a random process consisting of a single complex exponential in white noise,

$$r_x(k) = Pe^{jk\omega_0} + \sigma_w^2\delta(k)$$

and let $\mathbf{g}_i$ be the minimum variance bandpass filter

$$\mathbf{g}_i = \frac{\mathbf{R}_x^{-1}\mathbf{e}_i}{\mathbf{e}_i^H\mathbf{R}_x^{-1}\mathbf{e}_i}$$

that has a center frequency ω_i with $G(e^{j\omega_i}) = 1$. Assuming that $\omega_i \neq \omega_0$, prove that $G_i(z)$ has a zero that approaches $z = e^{j\omega_0}$ as $\sigma_w^2 \longrightarrow 0$.

8.8. A random process is known to consist of a single sinusoid in white noise,

$$x(n) = A\cos(n\omega_0 + \phi) + w(n)$$

Thus, the autocorrelation sequence for $x(n)$ is

$$r_x(k) = \frac{1}{2}A^2\cos(k\omega_0) + \sigma_w^2\delta(k)$$

(a) If $\omega_0 = \pi/4$, $A = \sqrt{2}$, and $\sigma_w^2 = 1$, find the second-order MEM spectrum, $\hat{P}_{mem}(e^{j\omega})$.

(b) Determine the location of the poles of $\hat{P}_{mem}(z)$.

(c) Does the peak of $\hat{P}_{mem}(e^{j\omega})$ provide an accurate estimate of ω_0? How does this estimate of ω_0 compare to that obtained using the Pisarenko Harmonic decomposition?

8.9. Suppose that we have determined the following values for the autocorrelation sequence of a real-valued random process $x(n)$:

$$r_x(0) = 1 \quad ; \quad r_x(1) = a \quad ; \quad r_x(2) = 0$$

(a) Using the Blackman-Tukey method with a rectangular window, find and make a carefully labeled sketch of the estimated power spectrum, $\hat{P}_{BT}(e^{j\omega})$.

(b) Repeat part (a) for a second-order MEM spectrum estimate, $\hat{P}_{mem}(e^{j\omega})$.

(c) Repeat part (a) for a MV spectrum estimate, $\hat{P}_{MV}(e^{j\omega})$.

(d) What can you say about the autocorrelation sequences that correspond to the spectrum estimates $\hat{P}_{BT}(e^{j\omega})$, $\hat{P}_{mem}(e^{j\omega})$, and $\hat{P}_{MV}(e^{j\omega})$ found in parts (a)-(c)?

8.10. Given that the sixth-order minimum variance spectrum estimate of a process $x(n)$ is

$$\hat{P}_{MV}(e^{j\omega}) = \frac{1}{1 + a\cos 4\omega + 4a\cos 6\omega}$$

and the seventh-order estimate is

$$\hat{P}_{MV}(e^{j\omega}) = \frac{1}{1 - 2a\cos 2\omega - a\cos 7\omega}$$

find the seventh-order maximum entropy spectrum, $\hat{P}_{mem}(e^{j\omega})$.

8.11. The first-order ($p = 1$) minimum variance spectrum estimate of a random process is

$$\hat{P}_{MV}(e^{j\omega}) = \frac{8}{3 - \cos\omega}$$

(a) Find the autocorrelations, $r_x(0)$ and $r_x(1)$, that produced this spectrum estimate.

(b) In general, given the pth-order minimum variance estimate $\hat{P}_{MV}(e^{j\omega})$, is it possible to recover the values of the autocorrelation sequence that produce this estimate?

8.12. The second-order maximum entropy spectrum of a process $x(n)$ is

$$\hat{P}_{mem}(e^{j\omega}) = \frac{2}{\left|1 - 0.5e^{-j\omega} + 0.25e^{-2j\omega}\right|^2}$$

(a) What is the first-order maximum entropy spectrum?

(b) Find the second-order minimum variance spectrum estimate.

8.13. From measurements of a process $x(n)$, we estimate the following values for the autocorrelation sequence:

$$r_x(k) = \alpha^{|k|} \quad ; \quad |k| \leq M$$

where $|\alpha| < 1$. Estimate the power spectrum using

(a) The Blackman-Tukey method with a rectangular window.

(b) The minimum variance method.

(c) The maximum entropy method.

8.14. In Eq. (8.97), the entropy of a Gaussian random process was given as

$$H(x) = \frac{1}{2\pi} \int_{-\pi}^{\pi} \ln P_x(e^{j\omega}) d\omega$$

In this problem, we derive another expression for the entropy. Let $x(n)$ be a real-valued zero mean Gaussian random process, and let $\mathbf{x} = [x(0), x(1), \ldots, x(N-1)]^T$ be an N-dimensional Gaussian random vector that is formed from samples of this process. The probability density function for this random vector is

$$f_x(\mathbf{x}) = \frac{1}{(2\pi)^{N/2}(\det \mathbf{R}_x)^{1/2}} \exp\left\{-\tfrac{1}{2}\mathbf{x}^T \mathbf{R}_x^{-1} \mathbf{x}\right\}$$

where $\mathbf{R}_x$ is the autocorrelation matrix.

(a) The *average entropy* of a random vector $\mathbf{x}$ is defined as

$$H_N(\mathbf{x}) = -\frac{1}{N} \int f_x(\mathbf{x}) \ln f_x(\mathbf{x}) d\mathbf{x}$$

Show that the average entropy of a zero mean Gaussian random vector is

$$H_N(\mathbf{x}) = \frac{1}{2} \ln(2\pi e) + \frac{1}{2N} \ln \det \mathbf{R}_x$$

(b) Show that the average entropy of a Gaussian random vector may be written as

$$H_N(\mathbf{x}) = \frac{1}{2} \ln(2\pi e) + \frac{1}{N} \sum_{k=0}^{N-1} \ln \epsilon_k$$

where ϵ_k is the prediction error sequence that is generated by the Levinson-Durbin recursion from the autocorrelation sequence $r_x(k)$.

(c) The *entropy rate* of a process $x(n)$ is the limit, as $N \to \infty$, of the average entropy [40],

$$\bar{H}(\mathbf{x}) = \lim_{N \to \infty} H_N(x)$$

Given a partial autocorrelation sequence, $r_x(k)$, for $k = 0, 1, \ldots, N-1$, find the spectrum $P_x(e^{j\omega})$ that maximizes $\bar{H}(x)$ subject to the constraint that the spectrum is consistent with the given autocorrelations.

(d) Let $P_x(e^{j\omega})$ be the power spectrum of a wide-sense stationary process with an $N \times N$ autocorrelation matrix $\mathbf{R}_x$, and let $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_N \geq 0$ be the eigenvalues. *Szegö's theorem* states that if $g(\cdot)$ is a continuous real-valued function then

$$\lim_{N\to\infty} \frac{1}{N} \sum_{k=1}^{N} g(\lambda_k) = \frac{1}{2\pi} \int_{-\pi}^{\pi} g\left[P_x(e^{j\omega})\right] d\omega$$

Use Szegö's theorem to show that

$$H(x) = \frac{1}{2} \ln(2\pi e) + \bar{H}(x)$$

i.e., that the two entropy expressions are equal to within an additive constant.

8.15. In this problem, we examine how the entropy of a process changes with the addition of a harmonic process. Let $y(n)$ be a random process with a power spectrum

$$P_y(e^{j\omega}) = P_x(e^{j\omega}) + P_\epsilon(e^{j\omega})$$

where

$$P_\epsilon(e^{j\omega}) = \begin{cases} 1/\epsilon & ; \quad |\omega - \omega_0| < \epsilon \\ 0 & ; \quad \text{otherwise} \end{cases}$$

(a) Find the entropy of $y(n)$.

(b) What is the entropy of this process in the limit as $\epsilon \to 0$?

8.16. Given an autocorrelation sequence $r_x(k)$ for $k = 0, 1, \ldots, p$, the maximum entropy spectrum is

$$\hat{P}_{mem}(e^{j\omega}) = \frac{\epsilon_p}{\left|1 + \sum_{k=1}^{p} a_p(k) e^{-jk\omega}\right|^2}$$

where the coefficients $a_p(k)$ are the solution to the normal equations $\mathbf{R}_x \mathbf{a}_p = \epsilon_p \mathbf{u}_1$. If Γ_k are the reflection coefficients produced by the Levinson-Durbin recursion, show that the MEM spectrum may be upper and lower bounded in terms of Γ_k as follows,

$$r_x(0) \prod_{k=1}^{p} \frac{1 - |\Gamma_k|}{1 + |\Gamma_k|} \leq \hat{P}_{mem}(e^{j\omega}) \leq r_x(0) \prod_{k=1}^{p} \frac{1 + |\Gamma_k|}{1 - |\Gamma_k|}$$

Hint: Begin with the frequency domain version of the Levinson order-update equation and use the inequality,

$$\big||a| - |b|\big| \leq |a + b| \leq \big||a| + |b|\big|$$

8.17. Let $x(n)$ be a first-order Gaussian autoregressive process with power spectrum

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

where a and c are real numbers.

(a) With the constraint that the total power in the process is equal to one, find the value or values of a and c that maximize the entropy of $x(n)$.

(b) Repeat part (a) and find the value or values of a and c that minimize the entropy.

8.18. The estimated autocorrelation sequence of a random process $x(n)$ for lags $k = 0, 1, 2, 3, 4$ are

$$r_x(0) = 2 \quad ; \quad r_x(1) = 1 \quad ; \quad r_x(2) = 1 \quad ; \quad r_x(3) = 0.5 \quad ; \quad r_x(4) = 0$$

Estimate the power spectrum of $x(n)$ for each of the following cases.

(a) $x(n)$ is an AR(2) process.

(b) $x(n)$ is an MA(2) process.

(c) $x(n)$ is an ARMA(1,1) process.

(d) $x(n)$ contains a single sinusoid in white noise.

8.19. The first three values of the autocorrelation sequence for a process $x(n)$ are:

$$r_x(0) = 1 \quad ; \quad r_x(1) = 0 \quad ; \quad r_x(2) = -\alpha$$

where $0 < \alpha < 1$. The eigenvalues of the 3×3 autocorrelation matrix that is formed from these autocorrelations are $\lambda_1 = 1 + \alpha$, $\lambda_2 = 1$, and $\lambda_3 = 1 - \alpha$, and the corresponding eigenvectors are

$$\mathbf{v}_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} \quad ; \quad \mathbf{v}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \quad ; \quad \mathbf{v}_3 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

(a) Use the Blackman-Tukey method with a rectangular window to estimate the power spectrum of $x(n)$, and make a carefully labeled sketch of your estimate.

(b) Suppose that $x(n)$ is known to consist of two complex exponentials in white noise. Estimate the power spectrum of $x(n)$ and make a carefully labeled sketch of your estimate.

8.20. Suppose that we would like to estimate the power spectrum of an AR(2) process

$$x(n) = a(1)x(n-1) + a(2)x(n-2) + w(n)$$

where $w(n)$ is unit variance white noise. However, our measurements of $x(n)$ are noisy, and what we observe is the process

$$y(n) = x(n) + v(n)$$

where the measurement noise, $v(n)$, is uncorrelated with $x(n)$. It is known that $v(n)$ is a first-order moving average process,

$$v(n) = b(0)q(n) + b(1)q(n-1)$$

where $q(n)$ is white noise. Based on measurements of $v(n)$, the power spectrum of $v(n)$ is estimated to be

$$\hat{P}_v(e^{j\omega}) = 3 + 2\cos\omega$$

From $y(n)$ we estimate the following values of the autocorrelation sequence $r_y(k)$,

$$\hat{r}_y(0) = 5 \quad ; \quad \hat{r}_y(1) = 2 \quad ; \quad \hat{r}_y(2) = 0 \quad ; \quad \hat{r}_y(3) = -1 \quad ; \quad \hat{r}_y(4) = 0.5$$

Using all of the given information, estimate the power spectrum of $x(n)$ using the maximum entropy method.

8.21. Show that for $N \gg 1$, estimating the order of an autoregressive process by minimizing FPE(p) is equivalent to minimizing AIC(p). Hint: Show that for large N,

$$N \ln \text{FPE}(p) \approx \text{AIC}(p)$$

and use the fact that, if x is small, then $\ln(1 + x) \approx x$.

8.22. You are given the following values for the autocorrelation sequence of a wide-sense stationary process $x(n)$,

$$r_x(0) = 2 \quad ; \quad r_x(1) = \sqrt{3}/2 \quad ; \quad r_x(2) = 0.5$$

The eigenvalues of the 3×3 Toeplitz autocorrelation matrix are $\lambda_1 = 3.5$, $\lambda_2 = 1.5$, and $\lambda_3 = 1.0$ and the corresponding normalized eigenvectors are

$$\mathbf{v}_1 = \sqrt{2/5}\begin{bmatrix} \sqrt{3}/2 \\ 1 \\ \sqrt{3}/2 \end{bmatrix} \quad ; \quad \mathbf{v}_2 = \frac{1}{\sqrt{2}}\begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \quad ; \quad \mathbf{v}_3 = \frac{1}{\sqrt{5}}\begin{bmatrix} 1 \\ -\sqrt{3} \\ 1 \end{bmatrix}$$

It is known that $x(n)$ consists of a single sinusoid in white noise.

(a) Estimate the frequency of the sinusoid using the Blackman-Tukey method of frequency estimation.

(b) Use the MUSIC algorithm to estimate the frequency of the sinusoid.

(c) Repeat part (b) using the minimum norm algorithm.

8.23. The Pisarenko harmonic decomposition provides a way to estimate the frequencies of a sum of complex exponentials in white noise. As described in Sect. 8.6.2, the powers of the complex exponentials may be found by solving the set of linear equations given in Eq. (8.160). Another method that may be used is based on the orthogonality of the trigonometric sine and cosine functions. This orthogonality condition implies that

$$\det \begin{bmatrix} \sin\omega_1 & \sin\omega_2 & \cdots & \sin\omega_p \\ \sin 2\omega_1 & \sin 2\omega_2 & \cdots & \sin 2\omega_p \\ \vdots & \vdots & & \vdots \\ \sin p\omega_1 & \sin p\omega_2 & \cdots & \sin p\omega_p \end{bmatrix} \neq 0$$

and

$$\det \begin{bmatrix} 1 & 1 & \cdots & 1 \\ \cos\omega_1 & \cos\omega_2 & \cdots & \cos\omega_p \\ \vdots & \vdots & & \vdots \\ \cos(p-1)\omega_1 & \cos(p-1)\omega_2 & \cdots & \cos(p-1)\omega_p \end{bmatrix} \neq 0$$

provided $0 < \omega_i < \pi$ and $\omega_i \neq \omega_j$.

(a) Given the autocorrelation sequence of a pth-order harmonic process,

$$r_x(k) = \sum_{i=1}^{p} P_i e^{jk\omega_i} + \sigma_w^2 \delta(k)$$

evaluate the imaginary part of $r_x(k)$ and use the orthogonality of the sine functions to derive a set of linear equations that may be solved to find the signal powers P_i.

(b) How would you modify this algorithm if some of the frequencies were equal to zero or π?

(c) How would you modify this approach for a sum of sinusoids in white noise?

8.24. The Pisarenko harmonic decomposition was derived for a process that consists of a sum of complex exponentials in white noise. In this problem we generalize the decomposition to nonwhite noise. To accomplish this, we begin with an alternate derivation of the Pisarenko decomposition for white noise. Let

$$x(n) = \sum_{k=1}^{p} A_k e^{jn\omega_k} + w(n)$$

where $w(n)$ is noise that is uncorrelated with the complex exponentials.

(a) If $w(n)$ is white noise then, as we saw in Eq. (8.149), the autocorrelation matrix for $x(n)$ may be written as

$$\mathbf{R}_x = \mathbf{E}\mathbf{P}\mathbf{E}^H + \sigma_w^2\mathbf{I}$$

where $\mathbf{E}$ is a matrix of complex exponentials and $\mathbf{P}$ is a diagonal matrix of signal powers. If $x(n)$ is filtered with a pth-order FIR filter $\mathbf{a} = [a(0), a(1), \ldots, a(p)]^T$, then the power in the output process is

$$\xi = E\{|y(n)|^2\} = \mathbf{a}^H\mathbf{R}_x\mathbf{a}$$

If $\mathbf{a}$ is constrained to have unit norm, $\mathbf{a}^H\mathbf{a} = 1$, show that the filter that minimizes ξ has p zeros on the unit circle at the frequencies ω_k of the complex exponentials, and show that the minimum value of ξ is equal to σ_w^2.

(b) Now assume that $w(n)$ has an arbitrary power spectrum, $P_w(e^{j\omega})$. If the autocorrelation matrix of $w(n)$ is $\sigma_w^2\mathbf{Q}$, then the autocorrelation matrix for $x(n)$ becomes

$$\mathbf{R}_x = \mathbf{E}\mathbf{P}\mathbf{E}^H + \sigma_w^2\mathbf{Q}$$

Suppose that $x(n)$ is filtered with a pth-order FIR filter $\mathbf{a} = [a(0), a(1), \ldots, a(p)]^T$ that is normalized so that

$$\mathbf{a}^H\mathbf{Q}\mathbf{a} = 1$$

Show that the filter that minimizes the power in the filtered process has p zeros on the unit circle at the frequencies ω_k of the complex exponentials, and that the minimum value is equal to σ_w^2.

(c) Show that minimizing $\xi = \mathbf{a}^H\mathbf{R}_x\mathbf{a}$ subject to the constraint $\mathbf{a}^H\mathbf{Q}\mathbf{a} = 1$ is equivalent to solving the generalized eigenvalue problem

$$\mathbf{R}_x\mathbf{a} = \lambda\mathbf{Q}\mathbf{a}$$

for the minimum eigenvalue and eigenvector. Thus, the frequencies of the complex exponentials correspond to the roots of the polynomial that is formed from the minimum eigenvector

$$V_{\min}(z) = \sum_{k=0}^{p} v_{\min}(k)z^{-k}$$

and σ_w^2 corresponds to the minimum eigenvalue.

(d) A random process consists of single sinusoid in nonwhite noise,

$$x(n) = A\sin(n\omega_0 + \phi) + w(n)$$

The first three values of the autocorrelation sequence for $x(n)$ are

$$\mathbf{r}_x = [9.515, 7.758, 6.472]^T$$

It is known that the additive noise $w(n)$ is a moving average process that is generated by filtering white noise $v(n)$ as follows

$$w(n) = v(n) + 0.1v(n-1)$$

However, the variance of $v(n)$ is unknown. Find the frequency ω_0 and the power, $P = \frac{1}{2}A^2$, of the sinusoid.

8.25. A random process is known to consist of a single sinusoid in white noise

$$x(n) = A\sin(n\omega_0 + \phi) + w(n)$$

where the variance of $w(n)$ is σ_w^2.

(a) Suppose that the first three values of the autocorrelation sequence are estimated and found to be

$$r_x(0) = 1 \quad ; \quad r_x(1) = \beta \quad ; \quad r_x(2) = 0$$

Find and prepare a carefully labeled sketch of the spectrum estimate that is formed using the Blackman-Tukey method with a rectangular window.

(b) With the autocorrelations given in part (a), use the Pisarenko harmonic decomposition to estimate the variance of the white noise, σ_w^2, the frequency of the sinusoid, ω_0, and the sinusoid power, $P = \frac{1}{2}A^2$. How does your estimate of the white noise power and the sinusoid frequency depend upon β? Does the sinusoid power depend upon β?

(c) Suppose that we compute the periodogram $\hat{P}_{per}(e^{j\omega})$ using N samples of $x(n)$. Find and prepare a carefully labeled sketch of the expected value of this spectrum estimate. Is this estimate biased? Is it consistent?

(d) Using the autocorrelations given in part (a), find the second-order MEM power spectrum.

8.26. A random process may be classified in terms of the properties of the prediction error sequence ϵ_k that is produced when fitting an all-pole model to the process. Listed below are five different classifications for the error sequence:

1. $\epsilon_k = c > 0$ for all $k \geq 0$.
2. $\epsilon_k = c > 0$ for all $k \geq k_0$ for some $k_0 > 0$.
3. $\epsilon_k \rightarrow c$ as $k \rightarrow \infty$ where $c > 0$.
4. $\epsilon_k \rightarrow 0$ as $k \rightarrow \infty$.
5. $\epsilon_k = 0$ for all $k \geq k_0$ for some $k_0 > 0$.

For each of these classifications, describe as completely as possible the characteristics that may be attributed to the process and its power spectrum.

8.27. In the MUSIC algorithm, finding the peaks of the frequency estimation function

$$\hat{P}_{MU}(e^{j\omega}) = \frac{1}{\sum_{i=p+1}^{M} |\mathbf{e}^H \mathbf{v}_i|^2}$$

is equivalent to finding the minima of the denominator. Show that finding the *minima* of the denominator is equivalent to finding the *maxima* of

$$\sum_{i=1}^{p} |\mathbf{e}^H \mathbf{v}_i|^2$$

Hint: Use the fact that

$$\mathbf{I} = \sum_{i=1}^{M} \mathbf{v}_i \mathbf{v}_i^H$$

8.28. The 3×3 autocorrelation matrix of a harmonic process is

$$\mathbf{R}_x = \begin{bmatrix} 3 & -j & -1 \\ j & 3 & -j \\ -1 & j & 3 \end{bmatrix}$$

(a) Using the Pisarenko harmonic decomposition, find the complex exponential frequencies and the variance of the white noise.

(b) Repeat part (a) using the MUSIC algorithm, the eigenvector method, and the minimum norm method.

8.29. In this problem we prove that the spurious roots in the minimum norm method lie inside the unit circle. Let $x(n)$ be a random process that is a sum of p complex exponentials in white noise, and let $\mathbf{a}$ be an M-dimensional vector that lies in the noise subspace. The z-transform of $\mathbf{a}$ may be factored as follows

$$A(z) = A_0(z)A_1(z)$$

where

$$A_0(z) = \prod_{k=1}^{p} (1 - e^{j\omega_k} z^{-1})$$

is a monic polynomial that has p roots on the unit circle at the frequencies of the complex exponentials in $x(n)$, and $A_1(z)$ is a polynomial that contains the $M - p - 1$ spurious roots.

(a) Show that minimizing $\|\mathbf{a}\|^2$ is equivalent to minimizing

$$\frac{1}{2\pi} \int_{-\pi}^{\pi} |A(e^{j\omega})|^2 d\omega = \frac{1}{2\pi} \int_{-\pi}^{\pi} |A_0(e^{j\omega})|^2 |A_1(e^{j\omega})|^2 d\omega$$

where $A_0(e^{j\omega})$ is fixed and $A_1(e^{j\omega})$ is monic.

(b) From the results of part (a), show that minimizing $\|\mathbf{a}\|^2$ is thus equivalent to using the autocorrelation method to find the prediction error filter $A_1(z)$ for the signal whose z-transform is $A_0(z)$.

(c) From (c), argue that $A_1(z)$ must therefore have all of its roots inside the unit circle.

8.30. In the minimum norm method, the spurious zeros in the polynomial $A(z)$ are separated from those that lie on the unit circle by forcing the spurious roots to lie inside the unit circle. In some applications of eigenvector methods, such as system identification, some of the desired zeros may lie inside the unit circle. In this case, the desired roots cannot be distinguished from the spurious roots. The minimum norm method may be modified, however, to force the spurious zeros to lie *outside* the unit circle. This is done by constraining the *last* element of the vector $\mathbf{a}$ to have a value of one, i.e.,

$$\mathbf{a}^H \mathbf{u}_M = 1$$

where $\mathbf{u}_M = [0, 0, \ldots, 0, 1]^T$ is a unit vector with the last element equal to one.

(a) Derive the *modified minimum norm* algorithm that uses the constant that $\mathbf{a}^4 \mathbf{u}_M = 1$ instead of $\mathbf{a}^4 \mathbf{u}_1 = 1$ as in the minimum norm algorithm.

(b) The 3×3 autocorrelation matrix for a single complex exponential in white noise is

$$\mathbf{R}_x = \begin{bmatrix} 2 & 1-j & -j\sqrt{2} \\ 1+j & 2 & -j \\ j\sqrt{2} & j & 2 \end{bmatrix}$$

Find the frequency of the complex exponential and the locations of the spurious roots in the minimum norm frequency estimation function.

(c) Repeat part (b) for the modified minimum norm method.

Computer Exercises

C8.1. Consider the broadband AR(4) process $x(n)$ that is produced by filtering unit variance white Gaussian noise with the filter

$$H(z) = \frac{1}{(1 - 0.5z^{-1} + 0.5z^{-2})(1 + 0.5z^{-2})}$$

(a) Generate $N = 256$ samples of this process and estimate the power spectrum using the autocorrelation method with $p = 4$. Make a plot of the estimate and compare it to the true power spectrum.

(b) Repeat part (a) for 20 different realizations of the process $x(n)$. Generate an overlay plot of the 20 estimates and plot the ensemble average. Comment on the variance of the estimate and on how accurately the autocorrelation method is able to estimate the power spectrum.

(c) Repeat part (b) using model orders of $p = 6$, 8, and 12. Describe what happens when the model order becomes too large.

(d) Repeat parts (b) and (c) for the covariance, modified covariance, and Burg algorithms. Which approach seems to be the best for a broadband autoregressive process?

C8.2. Repeat Problem C8.1 for the narrowband AR(4) process that is generated by filtering unit variance white noise with

$$H(z) = \frac{1}{(1 - 1.585z^{-1} + 0.96z^{-2})(1 - 1.152z^{-1} + 0.96z^{-2})}$$

C8.3. Consider the process

$$y(n) = x(n) + w(n)$$

where $w(n)$ is white Gaussian noise with a variance σ_w^2 and $x(n)$ is an AR(2) process that is generated by filtering unit variance white noise with the filter

$$H(z) = \frac{1}{1 - 1.585z^{-1} + 0.96z^{-2}}$$

(a) Plot the power spectrum of $x(n)$ and $y(n)$.

(b) For $\sigma_w^2 = 0.5, 1, 2, 5$, generate $N = 100$ samples of the process $y(n)$, and estimate the power spectrum of $x(n)$ from $y(n)$ using the maximum entropy method with $p = 2$. What is the effect of the noise $w(n)$ on the accuracy of the spectrum estimate?

(c) Repeat part (b) using the maximum entropy method with $p = 5$. Describe your observations.

(d) Since the autocorrelation sequence of $y(n)$ is

$$r_y(k) = r_x(k) + \sigma_w^2 \delta(k)$$

investigate what happens to your estimate in (c) if the autocorrelation sequence is modified by subtracting σ_w^2 from $r_y(0)$ before the maximum entropy spectrum is computed. Does this improve the spectrum estimate?

C8.4. In this exercise, we look at what happens if an autoregressive spectrum estimation technique is used on a moving average process.

(a) Let $x(n)$ be the second-order moving average process that is formed by filtering unit variance white Gaussian noise $w(n)$ as follows,

$$x(n) = w(n) - w(n-2)$$

Examine the spectrum estimates that are produced using an autoregressive technique such as MEM or the covariance method, and discuss your findings. What happens as you let the model order become large?

(b) Repeat part (a) for the MA(3) process that is formed by filtering unit variance white Gaussian noise with the filter

$$H(z) = (1 - 0.98z^{-1})(1 - 0.96z^{-2})$$

C8.5. Write a MATLAB program to estimate the order of an all-pole random process using the Akaike final prediction error, the minimum descriptor length, the Akaike information criterion, and Parzen's CAT. Evaluate the accuracy of the estimates that are produced with these methods if the modeling errors $\mathcal{E}_p$ are derived using the autocorrelation method. Consider both broadband and narrowband AR processes. How much do the estimates change if $\mathcal{E}_p$ is replaced with the Burg error, $\mathcal{E}_p^B$? Discuss your findings.

C8.6. Consider the process $x(n)$ consisting of two sinusoids in noise,

$$x(n) = 2\cos(n\omega_1 + \phi_1) + 2\cos(n\omega_2 + \phi_2) + w(n)$$

where ϕ_1 and ϕ_2 are uncorrelated random variables that are uniformly distributed over the interval $[0, 2\pi]$, and $w(n)$ is a fourth-order moving average process that is generated by filtering unit variance white noise with a linear shift-invariant filter that has a system function

$$H(z) = (1 - z^{-1})(1 + z^{-1})^3$$

Let $\omega_1 = \pi/2$ and $\omega_2 = 1.1\pi/2$.

(a) Plot the power spectrum of $x(n)$.

(b) Find the variance of the moving average process and compare the power in $w(n)$ to the power in each of the sinusoids.

(c) How many values of $x(n)$ are necessary in order to resolve the frequencies of the two sinusoids using the periodogram?

(d) Compare the expected value of the periodogram at the sinusoid frequencies ω_1 and ω_2 with the power spectrum of the noise $P_w(e^{j\omega})$. What record length N is required in order for the expected value of the periodogram at w_1 and w_2 to be twice the value of $P_w(e^{j\omega})$?

(e) Generate a sample realization of $x(n)$ of length $N = 256$ and plot the periodogram. Are the two sinusoidal frequencies clearly visible in the spectrum estimate? Repeat for 20 different realizations of the process and discuss your findings.

(f) Estimate the expected value of the periodogram by averaging the periodograms that are formed from an ensemble of 50 realizations of $x(n)$. Plot the estimate and comment on its resolution and on how closely it estimates the moving average part of the spectrum.

(g) Repeat parts (e) and (f) using Bartlett's method with $K = 2, 4$, and 8 sections.

C8.7. Let $x(n)$ be the process defined in Problem C8.6 consisting of a fourth-order moving average process plus two sinusoids.

(a) Generate a sample realization of $x(n)$ of length $N = 256$, and estimate the spectrum using the maximum entropy method with $p = 4$, 8, 16, and 32. Repeat for 20 different realizations of the process, and generate an overlay plot of the estimates along with the ensemble average. Discuss your findings.

(b) Repeat part (a) using the Burg algorithm and the modified covariance method.

C8.8. One of the methods we have seen for estimating the numerator and denominator coefficients of an ARMA process is iterative prefiltering. In this exercise we consider the use of iterative prefiltering for spectrum estimation.

(a) Generate 100 samples of an ARMA(2,2) process $x(n)$ by filtering unit variance white Gaussian noise with a filter that has a system function of the form

$$H(z) = \frac{1 + 1.5z^{-1} + z^{-2}}{1 + 0.9z^{-2}}$$

Using the method of iterative prefiltering, find the second-order ARMA model for $x(n)$, i.e., $p = q = 2$. Make a plot of the power spectrum that is formed from this model, and compare it to the true power spectrum.

(b) Repeat your experiment in part (a) for 20 different realizations of $x(n)$. How close are the estimated model coefficients to the true model, on the average? How accurate are the estimates of the power spectrum that are generated from these models? What happens if the process is over-modeled by setting $p = q = 3$? What about $p = q = 5$?

C8.9. Let $P_x(e^{j\omega})$ be the power spectrum of a process that has an autocorrelation sequence $r_x(k)$, and let λ_k be the eigenvalues of the $N \times N$ autocorrelation matrix $\mathbf{R}_x$.

(a) Use Szegö's theorem (see Prob. 8.14) to estimate the eigenvalues of the 128 × 128 autocorrelation matrix of the lowpass random process $x(n)$ that has a power spectrum

$$P_x(e^{j\omega}) = \begin{cases} 1 & ; \quad |\omega| < \pi/2 \\ 0 & ; \quad \pi/2 \leq |\omega| \leq \pi \end{cases}$$

(b) Generate the 128 × 128 autocorrelation matrix for this process, find the eigenvalues, and plot them in increasing order. Given the autocorrelation sequence that is stored in the vector `r`, note that this plot may be generated with the single MATLAB command

```
plot(sort(eig(toeplitz(r)))).
```

Are your results consistent with the estimate found in part (a)?

(c) Repeat parts (a) and (b) for the harmonic process

$$x(n) = A\cos(n\omega_0 + \phi) + w(n)$$

where $w(n)$ is unit variance white noise.

C8.10. It has been said that the minimum variance spectrum estimate exhibits less variance than an AR spectrum estimate for long data records [30]. To investigate this claim, consider the autocorrelation sequence

$$r_x(k) = \cos(0.35\pi k) + \delta(k) + w(k) \quad ; \quad k = 0, 1, \ldots, p$$

where $w(k)$ is uniformly distributed white noise with zero mean and variance σ_w^2. This sequence represents a noisy estimate of the autocorrelation sequence of a random phase sinusoid of frequency $\omega = 0.35\pi$ in unit variance white noise.

(a) Compute the minimum variance spectrum estimate with $p = 10$ and $\sigma_w^2 = 0$.

(b) Generate an overlay plot of 25 minimum variance spectrum estimates with $\sigma_w^2 = 0.1$ and compare these estimates to that generated in part (a).

(c) Repeat parts (a) and (b) using the maximum entropy method and compare your results with the minimum variance estimates. Is the claim substantiated? Suppose the Burg algorithm is used instead of MEM. Are your results any different?

C8.11. In the minimum variance spectrum estimate method, the bandwidths of the bandpass filters are the same for each filter in the filter bank, $\Delta\omega = 2\pi/p$. However, since each filter $\mathbf{g}_i$ has a bandwidth that depends, in general, on the center frequency of the filter, it may be possible to *improve* the power estimate by modifying the definition for the filter bandwidth. For example, consider the "equivalent filter bandwidth," which is defined by

$$\Delta = \mathbf{g}_i^H \mathbf{g}_i$$

(a) Show that the resulting *modified MV spectrum* is given by

$$\hat{P}_x(e^{j\omega}) = \frac{\mathbf{e}^H \mathbf{R}_x^{-1} \mathbf{e}}{\mathbf{e}^H \mathbf{R}_x^{-2} \mathbf{e}}$$

(b) What is the modified MV estimate of white Gaussian noise that has a variance of σ_v^2.

(c) Modify the m-file `minvar.m` so that it finds the modified minimum variance spectrum estimate of a random process $x(n)$.

(d) Let $x(n)$ be a harmonic process consisting of a sum of two sinusoids in white noise. Let the sinusoid frequencies be $\omega_1 = 0.2\pi$ and $\omega_2 = 0.25\pi$, and assume that the

signal-to-noise ratio for each sinusoid is 10 dB, i.e.,

$$20\log_{10}\frac{P}{\sigma_w} = 10$$

where P is the sinusoid power and σ_w^2 is the variance of the white noise. Find the autocorrelation sequence of this process and compute the 10th-order modified MV spectrum estimate. Make a plot of your estimate and compare it to a 10th-order minimum variance estimate, and a 10th-order MEM estimate.

(e) Make a plot of the bandwidths of the bandpass filters versus ω that are used in the modified MV spectrum estimate generated in (d) and compare them to the fixed bandwidth $\Delta\omega = 2\pi/p$ used in the MV spectrum estimate.

(f) Create an ensemble of 25 different harmonic processes of the form given in (d) and generate an overlay plot of the modified MV spectrum estimates. Repeat for the MV spectrum estimate and comment on any differences that you observe.

C8.12. Let $x(n)$ be the harmonic process

$$x(n) = \sum_{i=1}^{3} A_i e^{jn\omega_i} + w(n)$$

where $w(n)$ is unit variance white Gaussian noise. Let $A_1 = 4e^{j\phi_1}$, $A_2 = 3e^{j\phi_2}$, and $A_3 = e^{j\phi_3}$ where ϕ_i are uncorrelated random variables that are uniformly distributed between $-\pi$ and π. In addition, let $\omega_1 = 0.4\pi$, $\omega_2 = 0.45\pi$, and $\omega_3 = 0.8\pi$.

(a) Assuming that it is known that $x(n)$ contains three complex exponentials, use the Pisarenko harmonic decomposition to estimate the frequencies, and discuss the accuracy of your estimates. Repeat for 20 different realizations of $x(n)$. How accurate are the frequency estimates, on the average? How much variance is there in the frequency estimates? What happens if you overestimate the number of complex exponentials? What about if the number of complex exponentials is underestimated?

(b) Write an m-file to estimate the powers in the complex exponentials. Using this m-file, estimate the powers using the frequency estimates derived in part (a). Repeat the estimation of the powers using the correct frequencies.

(c) Repeat part (a) using the MUSIC algorithm, the eigenvector method, and the minimum norm algorithm on 20 different realizations of $x(n)$. Compare the accuracy of the estimates that are produced with each method.

C8.13. If a random process consists of a sum of complex exponentials in white noise, one way to estimate the number of complex exponentials is to perform an eigendecomposition of the autocorrelation matrix and examine the multiplicity of the minimum eigenvalue.

(a) Generate $N = 100$ samples of the random process

$$x(n) = \sum_{i=1}^{3} A_i e^{jn\omega_i} + w(n)$$

where $w(n)$ is white Gaussian noise with a variance σ_w^2. Let $|A_i| = 2$ for $i = 1, 2, 3$, and let the phase of A be a random variable that is uniformly distributed between $-\pi$ and π. If $\omega_1 = 0.4\pi$, $\omega_2 = 0.5\pi$, and $\omega_3 = 1.2\pi$, find the eigenvalues of the 6×6 autocorrelation matrix. Is the number of complex exponentials evident from the eigenvalues?

(b) Repeat part (a) for autocorrelation matrices of size 15×15 and 30×30.

(c) Does the variance of the white noise have any effect on the ability to estimate the number of complex exponentials from the eigenvalues?

(d) Let

$$\alpha(k) = \left[\prod_{i=k+1}^{M} \lambda_i \right]^{1/(M-k)}$$

and let

$$\beta(k) = \frac{1}{M-k} \sum_{i=k+1}^{M} \lambda_i$$

where $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_M$ are the eigenvalues of the $M \times M$ autocorrelation matrix. Thus, $\alpha(k)$ is the harmonic mean of the noise eigenvalues and $\beta(k)$ is the arithmetic mean. The Akaike information criterion for estimating the number of complex exponentials is

$$\text{AIC}(k) = -N(M-k) \log \frac{\alpha(k)}{\beta(k)} + k(2M-k)$$

where N is the length of the data record. An estimate of the number of complex exponentials in a process $x(n)$ is the value of k that minimizes AIC(k). Determine the accuracy of the AIC in estimating the number of complex exponentials by applying it to a number of different harmonic processes.

ADAPTIVE FILTERING

9

9.1 INTRODUCTION

In the previous five chapters, we have considered a variety of different problems including signal modeling, Wiener filtering, and spectrum estimation. In each case, we made an important assumption that the signals that were being analyzed were stationary. In signal modeling, for example, it was assumed that the signal to be modeled could be approximated as the output of a linear *shift-invariant* filter $h(n)$ with an input that is either a unit sample, in the case of deterministic signal modeling, or stationary white noise, in the case of stochastic signal modeling. In Chapter 7, we looked at the problem of designing a linear *shift-invariant* filter that would produce the minimum mean-square estimate of a wide-sense stationary process $d(n)$ and in Chapter 8 we considered the problem of estimating the power spectrum $P_x(e^{j\omega})$ of a wide-sense stationary process $x(n)$. Unfortunately, since the signals that arise in almost every application will be nonstationary, the approaches and techniques that we have been considering thus far would not be appropriate. One way to circumvent this difficulty would be to process these nonstationary processes in blocks, over intervals for which the process may be assumed to be approximately stationary. This approach, however, is limited in its effectiveness for several reasons. First, for rapidly varying processes, the interval over which a process may be assumed to be stationary may be too small to allow for sufficient accuracy or resolution in the estimation of the relevant parameters. Second, this approach would not easily accommodate step changes within the analysis intervals. Third, and perhaps most important, this solution imposes an incorrect model on the data, i.e., piecewise stationary. Therefore, a better approach would be start over and begin with a nonstationarity assumption at the outset.

In order to motivate the approach that we will be considering in this chapter, let us reconsider the Wiener filtering problem within the context of nonstationary processes. Specifically, let $w(n)$ denote the unit sample response of the FIR Wiener filter that produces the minimum mean-square estimate of a desired process $d(n)$,

$$\hat{d}(n) = \sum_{k=0}^{p} w(k)x(n-k) \tag{9.1}$$

As we saw in Chap. 7, if $x(n)$ and $d(n)$ are jointly wide-sense stationary processes, with $e(n) = d(n) - \hat{d}(n)$, then the filter coefficients that minimize the mean-square error

$E\{|e(n)|^2\}$ are found by solving the Wiener-Hopf equations

$$\mathbf{R}_x\mathbf{w} = \mathbf{r}_{dx} \tag{9.2}$$

However, if $d(n)$ and $x(n)$ are nonstationary, then the filter coefficients that minimize $E\{|e(n)|^2\}$ will depend on n, and the filter will be shift-varying, i.e.,

$$\hat{d}(n) = \sum_{k=0}^{p} w_n(k)x(n-k) \tag{9.3}$$

where $w_n(k)$ is the value of the kth filter coefficient at time n. Using vector notation, this estimate may be expressed as

$$\hat{d}(n) = \mathbf{w}_n^T\mathbf{x}(n)$$

where

$$\mathbf{w}_n = \Big[w_n(0),\ w_n(1),\ \ldots,\ w_n(p)\Big]^T$$

is the vector of filter coefficients at time n, and

$$\mathbf{x}(n) = \Big[x(n),\ x(n-1),\ \ldots,\ x(n-p)\Big]^T$$

In many respects, the design of a shift-varying (adaptive) filter is much more difficult than the design of a (shift-invariant) Wiener filter since, for each value of n, it is necessary to find the set of optimum filter coefficients, $w_n(k)$, for $k = 0, 1, \ldots, p$. However, the problem may be simplified considerably if we relax the requirement that $\mathbf{w}_n$ minimize the mean-square error at each time n and consider, instead, a coefficient update equation of the form

$$\boxed{\mathbf{w}_{n+1} = \mathbf{w}_n + \Delta\mathbf{w}_n} \tag{9.4}$$

where $\Delta\mathbf{w}_n$ is a *correction* that is applied to the filter coefficients $\mathbf{w}_n$ at time n to form a new set of coefficients, $\mathbf{w}_{n+1}$, at time $n+1$. This update equation is the heart of the adaptive filters that we will be designing in this chapter. As illustrated in Fig. 9.1, the design of an adaptive filter involves defining how this correction is to be formed. Even for stationary processes, there are several reasons why we may prefer to implement a time-invariant Wiener filter using Eq. (9.4). First, if the order of the filter p is large, then it may be difficult or impractical to solve the Wiener-Hopf equations directly. Second, if $\mathbf{R}_x$ is ill-conditioned (almost singular) then the solution to the Wiener-Hopf equations will be numerically sensitive to round-off errors and finite precision effects. Finally, and perhaps most importantly, is the fact that solving the Wiener-Hopf equations requires that the autocorrelation $r_x(k)$ and the cross-correlation $r_{dx}(k)$ be known. Since these ensemble averages are typically unknown, then it is necessary to estimate them from measurements of the processes. Although we could use estimates such as

$$\hat{r}_x(k) = \frac{1}{N}\sum_{n=0}^{N-1} x(n)x^*(n-k)$$

$$\hat{r}_{dx}(k) = \frac{1}{N}\sum_{n=0}^{N-1} d(n)x^*(n-k) \tag{9.5}$$

doing so would result in a delay of N samples. Even more importantly, in an environment for which the ensemble averages are changing in time, these estimates would need to be updated continuously.

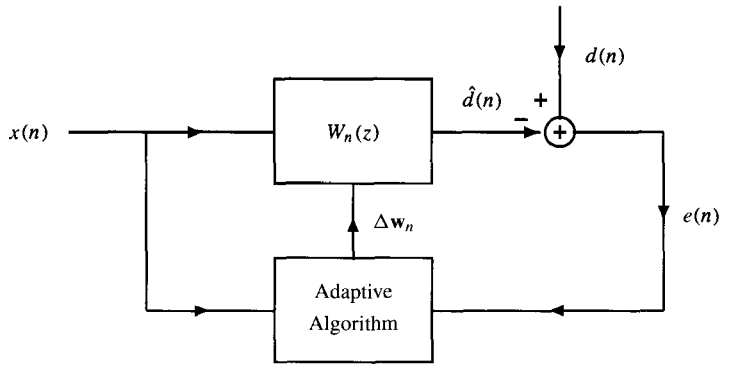
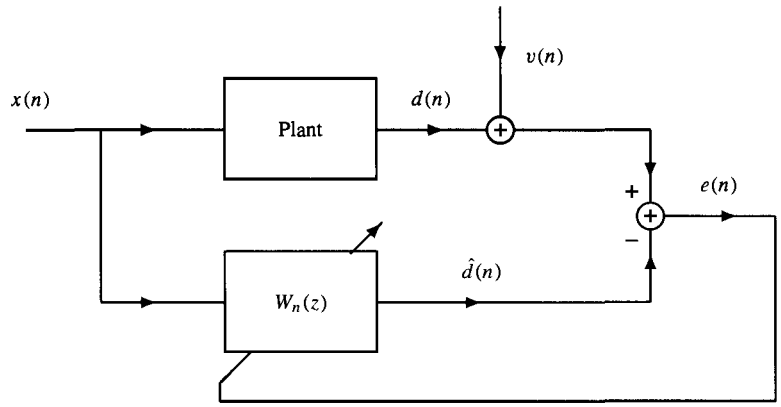


Figure 9.1 *Block diagram of an adaptive filter consisting of a shift-varying filter $W_n(z)$ and an adaptive algorithm for updating the filter coefficients $w_n(k)$.*

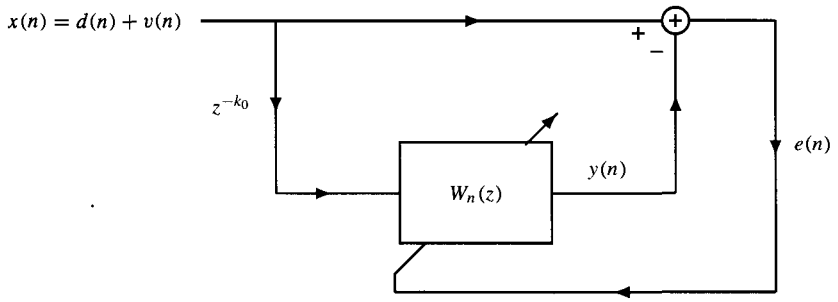
The key component of an adaptive filter is the set of rules, or algorithm, that defines how the correction $\Delta\mathbf{w}_n$ is to be formed. Although it is not yet clear what this correction should be, what is clear is that the sequence of corrections should decrease the mean-square error. In fact, whatever algorithm is used, the adaptive filter should have the following properties:

1. In a stationary environment, the adaptive filter should produce a sequence of corrections $\Delta\mathbf{w}_n$ in such a way that $\mathbf{w}_n$ converges to the solution to the Wiener-Hopf equations,
$$\lim_{n\to\infty} \mathbf{w}_n = \mathbf{R}_x^{-1}\mathbf{r}_{dx}$$
2. It should not be necessary to know the signal statistics $r_x(k)$ and $r_{dx}(k)$ in order to compute $\Delta\mathbf{w}_n$. The estimation of these statistics should be "built into" the adaptive filter.
3. For nonstationary signals, the filter should be able to adapt to the changing statistics and "track" the solution as it evolves in time.

An important issue in the implementation of an adaptive filter that is apparent from Fig. 9.1 is the requirement that the error signal, $e(n)$, be available to the adaptive algorithm. Since it is $e(n)$ that allows the filter to *measure* its performance and determine how the filter coefficients should be modified, without $e(n)$ the filter would not be able to adapt. In some applications, acquiring $d(n)$ is straightforward, which makes the evaluation of $e(n)$ easy. Consider, for example, the problem of system identification shown in Fig. 9.2*a* in which a plant produces an output $d(n)$ in response to a known input $x(n)$. The goal is to develop a model for the plant, $W_n(z)$, that produces a response $\hat{d}(n)$ that is as close as possible to $d(n)$. To account for observation noise in the measurement of the plant output, an additive noise source $v(n)$ is shown in the figure. In other applications, the acquisition of $d(n)$ is not as straightforward and some ingenuity is required. For example, consider the problem of interference cancellation in which a signal $d(n)$ is observed in the presence of an interfering signal $v(n)$,
$$x(n) = d(n) + v(n)$$
Since $d(n)$ is unknown, the error sequence cannot be generated directly. In some circumstances, however, the system shown in Fig. 9.2*b* will generate a sequence that may be used by the adaptive filter to estimate $d(n)$. Specifically, suppose that $d(n)$ and $v(n)$ are



(a) System identification



(b) Noise cancellation

Figure 9.2 *Two adaptive filtering applications that illustrate how an error sequence, $e(n)$, may be generated.*

real-valued, uncorrelated zero mean processes. In addition, suppose that $d(n)$ is a narrowband process and that $v(n)$ is a broadband process with an autocorrelation sequence that is approximately zero for lags $k \geq k_0$. The mean-square error that is formed by taking the difference between $x(n)$ and the output of the adaptive filter, $y(n)$, may be expressed as follows:

$$\begin{aligned} E\{e^2(n)\} &= E\left\{\left[d(n) + v(n) - y(n)\right]^2\right\} \\ &= E\left\{v^2(n)\right\} + E\left\{\left[d(n) - y(n)\right]^2\right\} + 2E\left\{v(n)\left[d(n) - y(n)\right]\right\} \end{aligned} \tag{9.6}$$

Since $v(n)$ and $d(n)$ are uncorrelated, then $E\{v(n)d(n)\} = 0$ and the last term becomes

$$2E\left\{v(n)\left[d(n) - y(n)\right]\right\} = -2E\left\{v(n)y(n)\right\}$$

In addition, since the input to the adaptive filter is $x(n - k_0)$, then the output $y(n)$ is

$$y(n) = \sum_{k=0}^{p} w_n(k)x(n - k_0 - k) = \sum_{k=0}^{p} w_n(k)\left[d(n - k_0 - k) + v(n - k_0 - k)\right]$$

Thus,

$$E\{v(n)y(n)\} = \sum_{k=0}^{p} w_n(k)\Big[E\{v(n)d(n-k_0-k)\} + E\{v(n)v(n-k_0-k)\}\Big]$$

Finally, since $v(n)$ is uncorrelated with $d(n)$ as well as with $v(n-k_0-k)$, then $E\{v(n)y(n)\} = 0$ and the mean-square error becomes

$$E\{e^2(n)\} = E\{v^2(n)\} + E\{[d(n) - y(n)]^2\}$$

Therefore, minimizing $E\{e^2(n)\}$ is equivalent to minimizing $E\{[d(n)-y(n)]^2\}$, the mean-square error between $d(n)$ and the output of the adaptive filter, $y(n)$. Thus, the output of the adaptive filter is the minimum mean-square estimate of $d(n)$.

In this chapter, we will consider a variety of different methods for designing and implementing adaptive filters. As we will see, the efficiency of the adaptive filter and its performance in estimating $d(n)$ will depend on a number of factors including the type of filter (FIR or IIR), the filter structure (direct form, parallel, lattice, etc.), and the way in which the performance measure is defined (mean-square error, least squares error). The organization of this chapter is as follows. In Section 9.2, we begin with the development of the FIR adaptive filters that are based on the method of steepest descent. Of primary interest will be the LMS adaptive filter. We will consider direct form as well as lattice filter structures. In Section 9.3, we will look briefly at the design of adaptive IIR filters. Being more difficult to characterize in terms of their properties and performance, the focus will be on the LMS adaptive recursive filter. Finally, in Section 9.4, the Recursive Least Squares (RLS) algorithm will be developed. There is a wide variety of applications in which adaptive filters have been successfully used such as linear prediction, echo cancellation, channel equalization, interference cancellation, adaptive notch filtering, adaptive control, system identification, and array processing. Therefore, as we progress through this chapter and look at specific adaptive filtering algorithms, we will briefly introduce some of these applications.

9.2 FIR ADAPTIVE FILTERS

In this section, we begin our study of adaptive filters by looking at the design of FIR (non-recursive) adaptive filters. In contrast to IIR or recursive adaptive filters, FIR filters are routinely used in adaptive filtering applications that range from adaptive equalizers in digital communication systems [26] to adaptive noise control systems [14]. There are several reasons for the popularity of FIR adaptive filters. First, stability is easily controlled by ensuring that the filter coefficients are bounded. Second, there are simple and efficient algorithms for adjusting the filter coefficients. Third, the performance of these algorithms is well understood in terms of their convergence and stability. Finally, FIR adaptive filters very often perform well enough to satisfy the design criteria.

An FIR adaptive filter for estimating a desired signal $d(n)$ from a related signal $x(n)$, as illustrated in Fig. 9.3, is

$$\hat{d}(n) = \sum_{k=0}^{p} w_n(k)x(n-k) = \mathbf{w}_n^T\mathbf{x}(n)$$

Here it is assumed that $x(n)$ and $d(n)$ are nonstationary random process and the goal is to

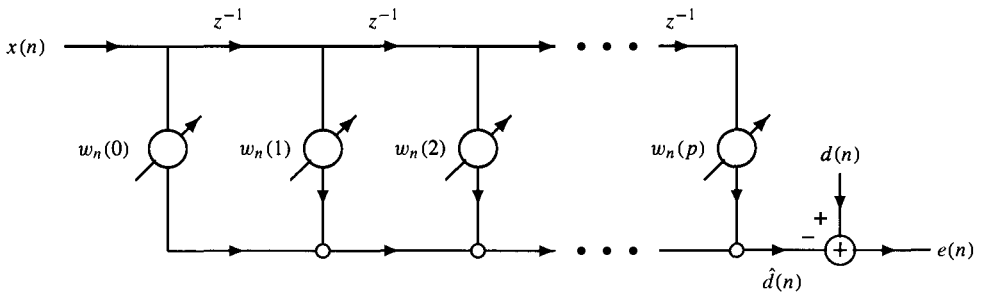


Figure 9.3 *A direct-form FIR adaptive filter.*

find the coefficient vector $\mathbf{w}_n$ at time n that minimizes the mean-square error,

$$\xi(n) = E\{|e(n)|^2\}$$

where

$$e(n) = d(n) - \hat{d}(n) = d(n) - \mathbf{w}_n^T \mathbf{x}(n) \tag{9.7}$$

As in the derivation of the FIR Wiener filter in Section 7.2, the solution to this minimization problem may be found by setting the derivative of $\xi(n)$ with respect to $w_n^*(k)$ equal to zero for $k = 0, 1, \ldots, p$. The result is

$$E\{e(n)x^*(n-k)\} = 0 \quad ; \quad k = 0, 1, \ldots, p \tag{9.8}$$

Substituting Eq. (9.7) into Eq. (9.8) we have

$$E\left\{\left[d(n) - \sum_{l=0}^{p} w_n(l)x(n-l)\right]x^*(n-k)\right\} = 0 \quad ; \quad k = 0, 1, \ldots, p$$

which, after rearranging terms, becomes

$$\sum_{l=0}^{p} w_n(l)E\{x(n-l)x^*(n-k)\} = E\{d(n)x^*(n-k)\} \quad ; \quad k = 0, 1, \ldots, p \tag{9.9}$$

Equation (9.9) is a set of $p+1$ linear equations in the $p+1$ unknowns $w_n(l)$. However, unlike the case of an FIR Wiener filter where it was assumed that $x(n)$ and $d(n)$ are jointly WSS, the solution to these equations depends on n. We may express these equations in vector form as follows:

$$\boxed{\mathbf{R}_x(n)\mathbf{w}_n = \mathbf{r}_{dx}(n)} \tag{9.10}$$

where

$$\mathbf{R}_x(n) = \begin{bmatrix} E\{x(n)x^*(n)\} & E\{x(n-1)x^*(n)\} & \cdots & E\{x(n-p)x^*(n)\} \\ E\{x(n)x^*(n-1)\} & E\{x(n-1)x^*(n-1)\} & \cdots & E\{x(n-p)x^*(n-1)\} \\ \vdots & \vdots & & \vdots \\ E\{x(n)x^*(n-p)\} & E\{x(n-1)x^*(n-p)\} & \cdots & E\{x(n-p)x^*(n-p)\} \end{bmatrix}$$

is a $(p+1) \times (p+1)$ Hermitian matrix of autocorrelations and

$$\mathbf{r}_{dx}(n) = \left[E\{d(n)x^*(n)\},\ E\{d(n)x^*(n-1)\},\ \cdots,\ E\{d(n)x^*(n-p)\}\right]^T \tag{9.11}$$

is a vector of cross-correlations between $d(n)$ and $x(n)$. Note that in the case of jointly WSS processes, Eq. (9.10) reduces to the Wiener-Hopf equations, and the solution $\mathbf{w}_n$ becomes independent of time. Instead of solving Eq. (9.10) for each value of n, which would be impractical in most real-time implementations, in the following section, we consider an iterative approach that is based on the method of steepest descent.

9.2.1 The Steepest Descent Adaptive Filter

In designing an FIR adaptive filter, the goal is to find the vector $\mathbf{w}_n$ at time n that minimizes the quadratic function

$$\xi(n) = E\{|e(n)|^2\}$$

Although the vector that minimizes $\xi(n)$ may be found by setting the derivatives of $\xi(n)$ with respect to $w^*(k)$ equal to zero, another approach is to *search* for the solution using the method of steepest descent. The *method of steepest descent* is an iterative procedure that has been used to find extrema of nonlinear functions since before the time of Newton. The basic idea of this method is as follows. Let $\mathbf{w}_n$ be an estimate of the vector that minimizes the mean-square error $\xi(n)$ at time n. At time $n+1$ a new estimate is formed by adding a correction to $\mathbf{w}_n$ that is designed to bring $\mathbf{w}_n$ closer to the desired solution. The correction involves taking a step of size μ in the direction of *maximum descent* down the quadratic error surface. For example, shown in Fig. 9.4*a* is a three-dimensional plot of a quadratic function of two real-valued coefficients, $w(0)$ and $w(1)$, given by[1]

$$\xi(n) = 6 - 6w(0) - 4w(1) + 6[w^2(0) + w^2(1)] + 6w(0)w(1) \tag{9.12}$$

Note that the contours of constant error, when projected onto the $w(0)$-$w(1)$ plane, form a set of concentric ellipses. The direction of steepest descent at any point in the plane is the direction that a marble would take if it were placed on the inside of this quadratic *bowl*. Mathematically, this direction is given by the *gradient*, which is the vector of partial

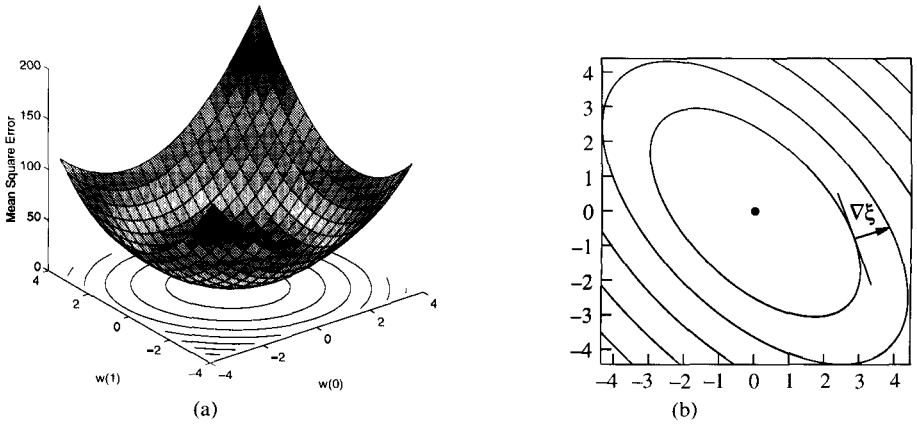


Figure 9.4 *(a) A quadratic function of two weights and (b) the contours of constant error. The gradient vector, which points in the direction of maximum increase in ξ, is orthogonal to the line that is tangent to the contour as illustrated in (b).*

[1] Although this error does not depend on n, we still denote it by $\xi(n)$.

derivatives of $\xi(n)$ with respect to the coefficients $w(k)$. For the quadratic function in Eq. (9.12), the gradient vector is

$$\nabla\xi(n) = \begin{bmatrix} \dfrac{\partial\xi(n)}{\partial w(0)} \\ \dfrac{\partial\xi(n)}{\partial w(1)} \end{bmatrix} = \begin{bmatrix} 12w(0) + 6w(1) - 6 \\ 12w(1) + 6w(0) - 4 \end{bmatrix}$$

As shown in Fig. 9.4*b*, for any vector $\mathbf{w}$, the gradient is orthogonal to the line that is tangent to the contour of constant error at $\mathbf{w}$. However, since the gradient vector points in the direction of *steepest ascent*, the direction of *steepest descent* points in the negative gradient direction. Thus, the update equation for $\mathbf{w}_n$ is

$$\boxed{\mathbf{w}_{n+1} = \mathbf{w}_n - \mu\nabla\xi(n)}$$

The step size μ affects the rate at which the weight vector moves down the quadratic surface and must be a positive number (a negative value for μ would move the weight vector up the quadratic surface in the direction of maximum ascent and would result in an increase in the error). For very small values of μ, the correction to $\mathbf{w}_n$ is small and the movement down the quadratic surface is slow and, as μ is increased, the rate of descent increases. However, there is an upper limit on how large the step size may be. For values of μ that exceed this limit, the trajectory of $\mathbf{w}_n$ becomes unstable and unbounded. The steepest descent algorithm may be summarized as follows:

1. Initialize the steepest descent algorithm with an initial estimate, $\mathbf{w}_0$, of the optimum weight vector $\mathbf{w}$.
2. Evaluate the gradient of $\xi(n)$ at the current estimate, $\mathbf{w}_n$, of the optimum weight vector.
3. Update the estimate at time n by adding a correction that is formed by taking a step of size μ in the negative gradient direction

$$\mathbf{w}_{n+1} = \mathbf{w}_n - \mu\nabla\xi(n)$$

4. Go back to (2) and repeat the process.

Let us now evaluate the gradient vector $\nabla\xi(n)$. Assuming that $\mathbf{w}$ is complex, the gradient is the derivative of $E\{|e(n)|^2\}$ with respect to $\mathbf{w}^*$. With

$$\nabla\xi(n) = \nabla E\{|e(n)|^2\} = E\{\nabla|e(n)|^2\} = E\{e(n)\nabla e^*(n)\}$$

and

$$\nabla e^*(n) = -\mathbf{x}^*(n)$$

it follows that

$$\nabla\xi(n) = -E\{e(n)\mathbf{x}^*(n)\}$$

Thus, with a step size of μ, the steepest descent algorithm becomes

$$\boxed{\mathbf{w}_{n+1} = \mathbf{w}_n + \mu E\{e(n)\mathbf{x}^*(n)\}} \tag{9.13}$$

To see how this steepest descent update equation for $\mathbf{w}_n$ performs, let us consider what

happens in the case of stationary processes. If $x(n)$ and $d(n)$ are jointly WSS then

$$\begin{aligned} E\{e(n)\mathbf{x}^*(n)\} &= E\{d(n)\mathbf{x}^*(n)\} - E\{\mathbf{w}_n^T\mathbf{x}(n)\mathbf{x}^*(n)\} \\ &= \mathbf{r}_{dx} - \mathbf{R}_x\mathbf{w}_n \end{aligned}$$

and the steepest descent algorithm becomes

$$\boxed{\mathbf{w}_{n+1} = \mathbf{w}_n + \mu(\mathbf{r}_{dx} - \mathbf{R}_x\mathbf{w}_n)} \tag{9.14}$$

Note that if $\mathbf{w}_n$ is the solution to the Wiener-Hopf equations, $\mathbf{w}_n = \mathbf{R}_x^{-1}\mathbf{r}_{dx}$, then the correction term is zero and $\mathbf{w}_{n+1} = \mathbf{w}_n$ for all n. Of greater interest, however, is how the weights evolve in time, beginning with an arbitrary initial weight vector $\mathbf{w}_0$. The following property defines what is required for $\mathbf{w}_n$ to converge to $\mathbf{w}$.

Property 1. For jointly wide-sense stationary processes, $d(n)$ and $x(n)$, the steepest descent adaptive filter converges to the solution to the Wiener-Hopf equations

$$\lim_{n\to\infty} \mathbf{w}_n = \mathbf{R}_x^{-1}\mathbf{r}_{dx}$$

if the step size satisfies the condition

$$0 < \mu < \frac{2}{\lambda_{\max}} \tag{9.15}$$

where $\lambda_{\max}$ is the maximum eigenvalue of the autocorrelation matrix $\mathbf{R}_x$.

To establish this property, we begin by rewriting Eq. (9.14) as follows:

$$\mathbf{w}_{n+1} = (\mathbf{I} - \mu\mathbf{R}_x)\mathbf{w}_n + \mu\mathbf{r}_{dx} \tag{9.16}$$

Subtracting $\mathbf{w}$ from both sides of this equation and using the fact that $\mathbf{r}_{dx} = \mathbf{R}_x\mathbf{w}$ we have

$$\mathbf{w}_{n+1} - \mathbf{w} = (\mathbf{I} - \mu\mathbf{R}_x)\mathbf{w}_n + \mu\mathbf{R}_x\mathbf{w} - \mathbf{w} = [\mathbf{I} - \mu\mathbf{R}_x](\mathbf{w}_n - \mathbf{w}) \tag{9.17}$$

If we let $\mathbf{c}_n$ be the *weight error vector*,

$$\boxed{\mathbf{c}_n = \mathbf{w}_n - \mathbf{w}} \tag{9.18}$$

then Eq. (9.17) becomes

$$\mathbf{c}_{n+1} = (\mathbf{I} - \mu\mathbf{R}_x)\mathbf{c}_n \tag{9.19}$$

Note that, unless $\mathbf{R}_x$ is a diagonal matrix, there will be cross-coupling between the coefficients of the weight error vector. However, we may decouple these coefficients by diagonalizing the autocorrelation matrix as follows. Using the spectral theorem (p. 44) the autocorrelation matrix may be factored as

$$\mathbf{R}_x = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^H$$

where $\mathbf{\Lambda}$ is a diagonal matrix containing the eigenvalues of $\mathbf{R}_x$, and $\mathbf{V}$ is a matrix whose columns are the eigenvectors of $\mathbf{R}_x$. Since $\mathbf{R}_x$ is Hermitian and nonnegative definite, the eigenvalues are real and non-negative, $\lambda_k \geq 0$, and the eigenvectors may be chosen to be

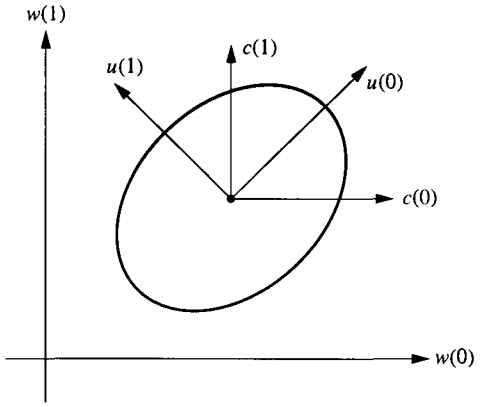


Figure 9.5 *Illustration of the relationships between the vectors* **w**, **c**, *and* **u**.

orthonormal, $\mathbf{V}\mathbf{V}^H = \mathbf{I}$, i.e., $\mathbf{V}$ is *unitary*. Incorporating this factorization into Eq. (9.19) leads to

$$\mathbf{c}_{n+1} = (\mathbf{I} - \mu\mathbf{V}\mathbf{\Lambda}\mathbf{V}^H)\mathbf{c}_n$$

Using the unitary property of $\mathbf{V}$ we have

$$\mathbf{c}_{n+1} = (\mathbf{V}\mathbf{V}^H - \mu\mathbf{V}\mathbf{\Lambda}\mathbf{V}^H)\mathbf{c}_n = \mathbf{V}(\mathbf{I} - \mu\mathbf{\Lambda})\mathbf{V}^H\mathbf{c}_n$$

Multiplying both sides of the equation by $\mathbf{V}^H$ gives

$$\mathbf{V}^H\mathbf{c}_{n+1} = (\mathbf{I} - \mu\mathbf{\Lambda})\mathbf{V}^H\mathbf{c}_n \tag{9.20}$$

If we define

$$\mathbf{u}_n = \mathbf{V}^H\mathbf{c}_n \tag{9.21}$$

then Eq. (9.20) becomes

$$\mathbf{u}_{n+1} = (\mathbf{I} - \mu\mathbf{\Lambda})\mathbf{u}_n$$

As illustrated in Fig. 9.5, Eq. (9.21) represents a rotation of the coordinate system for the weight error vector $\mathbf{c}_n$ with the new axes aligned with the eigenvectors $\mathbf{v}_k$ of the autocorrelation matrix. With an initial weight vector $\mathbf{u}_0$, it follows that

$$\mathbf{u}_n = (\mathbf{I} - \mu\mathbf{\Lambda})^n\mathbf{u}_0 \tag{9.22}$$

Since $(\mathbf{I} - \mu\mathbf{\Lambda})$ is a diagonal matrix then the kth component of $\mathbf{u}_n$ may be expressed as

$$u_n(k) = (1 - \mu\lambda_k)^n u_0(k) \tag{9.23}$$

In order for $\mathbf{w}_n$ to converge to $\mathbf{w}$ it is necessary that the weight error vector $\mathbf{c}_n$ converge to zero and, therefore, that $\mathbf{u}_n = \mathbf{V}^H\mathbf{c}_n$ converge to zero. This will occur for any $\mathbf{u}_0$ if and only if

$$|1 - \mu\lambda_k| < 1 \quad ; \quad k = 0, 1, \ldots, p$$

which places the following restriction on the step size μ

$$0 < \mu < \frac{2}{\lambda_{\max}}$$

as was to be shown. ■

Having found an expression for the evolution of the vector $\mathbf{u}_n$, we may derive an expression for the evolution of the weight vector $\mathbf{w}_n$. With

$$\mathbf{w}_n = \mathbf{w} + \mathbf{c}_n = \mathbf{w} + \mathbf{V}\mathbf{u}_n = \mathbf{w} + [\mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_p] \begin{bmatrix} u_n(0) \\ u_n(1) \\ \vdots \\ u_n(p) \end{bmatrix}$$

using Eq. (9.23) we have

$$\mathbf{w}_n = \mathbf{w} + \sum_{k=0}^{p} u_n(k)\mathbf{v}_k = \mathbf{w} + \sum_{k=0}^{p} (1 - \mu\lambda_k)^n u_0(k)\mathbf{v}_k$$

Since $\mathbf{w}_n$ is a linear combination of the eigenvectors $\mathbf{v}_n$, referred to as the *modes* of the filter, then $\mathbf{w}_n$ will converge no faster than the slowest decaying mode. With each mode decaying as $(1 - \mu\lambda_k)^n$, we may define the time constant τ_k to be the time required for the kth mode to reach $1/e$ of its initial value:

$$(1 - \mu\lambda_k)^{\tau_k} = 1/e$$

Taking logarithms we have

$$\boxed{\tau_k = -\frac{1}{\ln(1 - \mu\lambda_k)}} \tag{9.24}$$

If μ is small enough so that $\mu\lambda_k \ll 1$, then the time constant may be approximated by

$$\tau_k \approx \frac{1}{\mu\lambda_k}$$

Defining the overall time constant to be the time that it takes for the slowest decaying mode to converge to $1/e$ of its initial value we have

$$\boxed{\tau = \max\{\tau_k\} \approx \frac{1}{\mu\lambda_{\min}}} \tag{9.25}$$

Since Property 1 places an upper bound of $2/\lambda_{\max}$ on the step size if $\mathbf{w}_n$ is to converge to $\mathbf{w}$, let us write μ as follows

$$\mu = \alpha \frac{2}{\lambda_{\max}}$$

where α is a normalized step size with $0 < \alpha < 1$. In terms of α, the time constant becomes

$$\tau \approx \frac{1}{2\alpha} \frac{\lambda_{\max}}{\lambda_{\min}} = \frac{1}{2\alpha} \chi$$

where $\chi = \lambda_{\max}/\lambda_{\min}$ is the condition number of the autocorrelation matrix. Thus, the rate of convergence is determined by the eigenvalue spread. The reason for this dependence may be explained geometrically as illustrated in Fig. 9.6. In Fig. 9.6*a* are the error contours for a two-dimensional adaptive filter with $\lambda_1 = \lambda_2 = 1$, i.e., $\chi = 1$, along with the trajectory of $\mathbf{w}_n$. Note that the contours are circles and that, at any point, the direction of steepest descent points toward the minimum of the quadratic function. Fig. 9.6*b*, on the other hand, shows the error contours when $\lambda_1 = 3$ and $\lambda_2 = 1$, i.e., $\chi = 3$, along with the trajectory of $\mathbf{w}_n$.

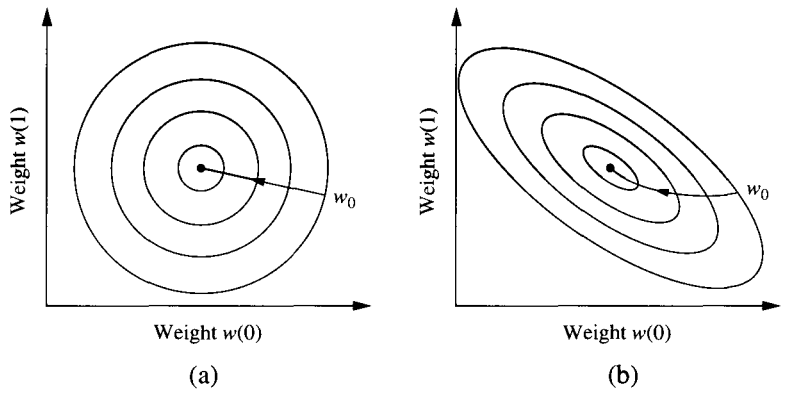


Figure 9.6 *The effect of the condition number on the convergence rate of the steepest descent algorithm. (a) With $\chi = 1$ the direction of steepest descent points toward the minimum of the error surface. (b) With $\chi = 3$ the direction of steepest descent does not, in general, point toward the minimum.*

Note that, unlike the case when $\chi = 1$, the direction of steepest descent is typically not pointing toward the minimum of the quadratic function.

Another important and useful measure of performance is the behavior of the mean-square error as a function of n. From Eq. (7.12) we see that for jointly wide-sense stationary processes the minimum mean-square error is

$$\xi_{\min} = r_d(0) - \mathbf{r}_{dx}^H \mathbf{w} \tag{9.26}$$

For an arbitrary weight vector, $\mathbf{w}_n$, the mean-square error is

$$\begin{aligned} \xi(n) &= E\{|e(n)|^2\} = E\{|d(n) - \mathbf{w}_n^T \mathbf{x}(n)|^2\} \\ &= r_d(0) - \mathbf{r}_{dx}^H \mathbf{w}_n - \mathbf{w}_n^H \mathbf{r}_{dx} + \mathbf{w}_n^H \mathbf{R}_x \mathbf{w}_n \end{aligned} \tag{9.27}$$

Expressing $\mathbf{w}_n$ in terms of the weight error vector $\mathbf{c}_n$ this becomes

$$\xi(n) = r_d(0) - \mathbf{r}_{dx}^H(\mathbf{w} + \mathbf{c}_n) - (\mathbf{w} + \mathbf{c}_n)^H \mathbf{r}_{dx} + (\mathbf{w} + \mathbf{c}_n)^H \mathbf{R}_x(\mathbf{w} + \mathbf{c}_n)$$

Expanding the products and using the fact that $\mathbf{R}_x \mathbf{w} = \mathbf{r}_{dx}$ we have

$$\xi(n) = r_d(0) - \mathbf{r}_{dx}^H \mathbf{w} + \mathbf{c}_n^H \mathbf{R}_x \mathbf{c}_n \tag{9.28}$$

Note that the first two terms are equal to the minimum error, $\xi_{\min}$. Therefore, the error at time n is

$$\xi(n) = \xi_{\min} + \mathbf{c}_n^H \mathbf{R}_x \mathbf{c}_n$$

With the decomposition $\mathbf{R}_x = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^H$ and the definition for $\mathbf{u}_n$ given in Eq. (9.21), we have

$$\boxed{\xi(n) = \xi_{\min} + \mathbf{u}_n^H \mathbf{\Lambda}_x \mathbf{u}_n} \tag{9.29}$$

Expanding the quadratic form and using the expression for $u_n(k)$ given in Eq. (9.23) yields

$$\xi(n) = \xi_{\min} + \sum_{k=0}^{p} \lambda_k |u_n(k)|^2 = \xi_{\min} + \sum_{k=0}^{p} \lambda_k (1 - \mu\lambda_k)^{2n} |u_0(k)|^2 \tag{9.30}$$

Thus, if the step size μ satisfies the condition for convergence given in Eq. (9.15), then $\xi(n)$ decays exponentially to $\xi_{\min}$. A plot of $\xi(n)$ versus n is referred to as the *learning curve* and indicates how rapidly the adaptive filter *learns* the solution to the Wiener-Hopf equations.

Although for stationary processes the steepest descent adaptive filter converges to the solution to the Wiener-Hopf equations when $\mu < 2/\lambda_{\max}$, this algorithm is primarily of theoretical interest and finds little use in adaptive filtering applications. The reason for this is that, in order to compute the gradient vector, it is necessary that $E\{e(n)\mathbf{x}^*(n)\}$ be known. For stationary processes this requires that the autocorrelation matrix of $x(n)$, and the cross-correlation between $d(n)$ and $\mathbf{x}(n)$ be known. In most applications, these ensemble averages are unknown and must be estimated from the data. In the next section, we look at the LMS algorithm, which incorporates an *estimate* of the expectation $E\{e(n)\mathbf{x}^*(n)\}$ into the adaptive algorithm.

9.2.2 The LMS Algorithm

In the previous section, we developed the steepest descent adaptive filter, which has a weight-vector update equation given by

$$\mathbf{w}_{n+1} = \mathbf{w}_n + \mu E\{e(n)\mathbf{x}^*(n)\} \tag{9.31}$$

A practical limitation with this algorithm is that the expectation $E\{e(n)\mathbf{x}^*(n)\}$ is generally unknown. Therefore, it must be replaced with an estimate such as the sample mean

$$\hat{E}\{e(n)\mathbf{x}^*(n)\} = \frac{1}{L}\sum_{l=0}^{L-1} e(n-l)\mathbf{x}^*(n-l) \tag{9.32}$$

Incorporating this estimate into the steepest descent algorithm, the update for $\mathbf{w}_n$ becomes

$$\mathbf{w}_{n+1} = \mathbf{w}_n + \frac{\mu}{L}\sum_{l=0}^{L-1} e(n-l)\mathbf{x}^*(n-l) \tag{9.33}$$

A special case of Eq. (9.33) occurs if we use a one-point sample mean ($L = 1$),

$$\hat{E}\{e(n)\mathbf{x}^*(n)\} = e(n)\mathbf{x}^*(n) \tag{9.34}$$

In this case, the weight vector update equation assumes a particularly simple form

$$\boxed{\mathbf{w}_{n+1} = \mathbf{w}_n + \mu e(n)\mathbf{x}^*(n)} \tag{9.35}$$

and is known as the *LMS algorithm* [36]. The simplicity of the algorithm comes from the fact that the update for the kth coefficient,

$$w_{n+1}(k) = w_n(k) + \mu e(n)x^*(n-k)$$

requires only one multiplication and one addition (the value for $\mu e(n)$ need only be computed once and may be used for all of the coefficients). Therefore, an LMS adaptive filter having $p+1$ coefficients requires $p+1$ multiplications and $(p+1)$ additions to update the filter coefficients. In addition, one addition is necessary to compute the error $e(n) = d(n) - y(n)$ and one multiplication is needed to form the product $\mu e(n)$. Finally, $p+1$ multiplications and p additions are necessary to calculate the output, $y(n)$, of the adaptive filter. Thus, a total of $2p+3$ multiplications and $2p+2$ additions per output point are required. The complete LMS algorithm is summarized in Table 9.1 and a MATLAB program is given in

Table 9.1 *The LMS Algorithm for a pth-Order FIR Adaptive Filter*

Parameters:	p = Filter order
	μ = Step size
Initialization:	$\mathbf{w}_0 = \mathbf{0}$
Computation:	For $n = 0, 1, 2, \ldots$
	(a) $y(n) = \mathbf{w}_n^T \mathbf{x}(n)$
	(b) $e(n) = d(n) - y(n)$
	(c) $\mathbf{w}_{n+1} = \mathbf{w}_n + \mu e(n)\mathbf{x}^*(n)$

The LMS Algorithm

```
function [A,E] = lms(x,d,mu,nord,a0)
%
X=convm(x,nord);
[M,N] = size(X);
if nargin < 5,   a0 = zeros(1,N);   end
a0 = a0(:).';
E(1) = d(1) - a0*X(1,:).';
A(1,:) = a0 + mu*E(1)*conj(X(1,:));
if M>1
for k=2:M-nord+1;
    E(k) = d(k) - A(k-1,:)*X(k,:).';
    A(k,:) = A(k-1,:) + mu*E(k)*conj(X(k,:));
    end;
end;
```

Figure 9.7 *A* MATLAB *program to estimate a process $d(n)$ from a related process $x(n)$ using the LMS algorithm.*

Fig. 9.7. Although based on a very crude estimate of $E\{e(n)\mathbf{x}^*(n)\}$, we will see that the LMS adaptive filter often performs well enough to be used successfully in a number of applications. In the following section, we consider the convergence of the LMS adaptive filter.

9.2.3 Convergence of the LMS Algorithm

In estimating the ensemble average $E\{e(n)\mathbf{x}^*(n)\}$ with a one-point sample average $e(n)\mathbf{x}^*(n)$, the LMS algorithm replaces the gradient in the steepest descent algorithm,

$$\nabla\xi(n) = -E\{e(n)\mathbf{x}^*(n)\}$$

with an estimated gradient

$$\hat{\nabla}\xi(n) = -e(n)\mathbf{x}^*(n) \tag{9.36}$$

When this is done, the correction that is applied to $\mathbf{w}_n$ is generally not aligned with the

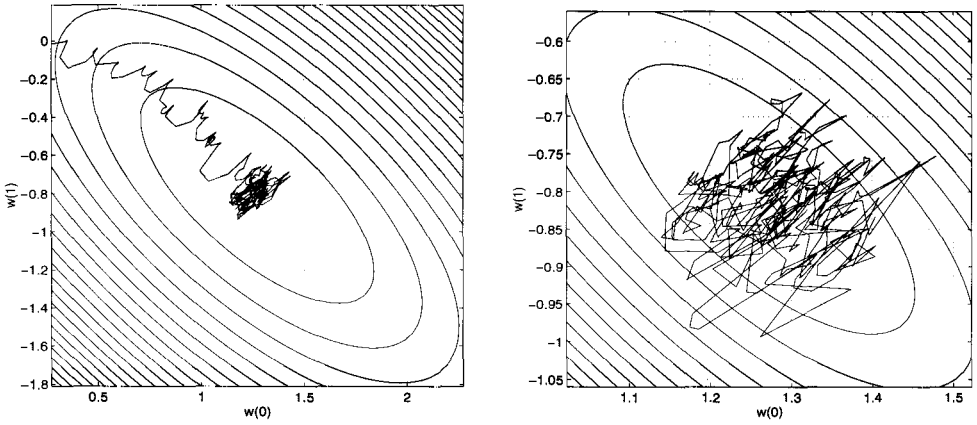


Figure 9.8 *Weight trajectories for a two-coefficient LMS adaptive filter. (a) The trajectory of* $\mathbf{w}_n$ *for 500 iterations with* $\mathbf{w}_0 = \mathbf{0}$. *The solution to the Wiener-Hopf equations is at the center of the ellipses. (b) The trajectory with* $\mathbf{w}_0 = \mathbf{R}_x^{-1}\mathbf{r}_{dx}$.

direction of steepest descent. However, since the gradient estimate is unbiased,

$$E\left\{\hat{\nabla}\xi(n)\right\} = -E\left\{e(n)\mathbf{x}^*(n)\right\} = \nabla\xi(n)$$

then the correction that is applied is, on the average, in the direction of steepest descent. An illustration of the behavior that is typically observed with the LMS algorithm for stationary processes is shown in Fig. 9.8. In Fig. 9.8*a*, with the weight vector initialized to $\mathbf{w}_0 = \mathbf{0}$, we see that the sequence of weights generally moves toward the solution to the Wiener-Hopf equations, $\mathbf{w} = \mathbf{R}_x^{-1}\mathbf{r}_{dx}$. However, as illustrated in Fig. 9.8*b*, if the weight vector is initialized with the solution to the Wiener-Hopf equations then, although the gradient at this point is zero, since a gradient estimate is used, $\mathbf{w}_n$ moves randomly within a neighborhood of this solution. In order to explain these observations, we will now consider the convergence of the LMS adaptive filter.

Since $\mathbf{w}_n$ is a vector of random variables, the convergence of the LMS algorithm must be considered within a statistical framework. Therefore, we will begin by assuming that $x(n)$ and $d(n)$ are jointly wide-sense stationary processes, and will determine when the coefficients $\mathbf{w}_n$ converge in the mean to $\mathbf{w} = \mathbf{R}_x^{-1}\mathbf{r}_{dx}$, i.e.,

$$\lim_{n\to\infty} E\left\{\mathbf{w}_n\right\} = \mathbf{w} = \mathbf{R}_x^{-1}\mathbf{r}_{dx}$$

We begin by substituting Eq. (9.7) into the LMS coefficient update equation as follows:

$$\mathbf{w}_{n+1} = \mathbf{w}_n + \mu\left[d(n) - \mathbf{w}_n^T\mathbf{x}(n)\right]\mathbf{x}^*(n)$$

Taking the expected value, we have

$$E\left\{\mathbf{w}_{n+1}\right\} = E\left\{\mathbf{w}_n\right\} + \mu E\left\{d(n)\mathbf{x}^*(n)\right\} - \mu E\left\{\mathbf{x}^*(n)\mathbf{x}^T(n)\mathbf{w}_n\right\} \tag{9.37}$$

Although the last term in Eq. (9.37) is not easy to evaluate, it may be simplified considerably if we make the following *independence assumption* [20]:

Independence Assumption. The data $\mathbf{x}(n)$ and the LMS weight vector $\mathbf{w}_n$ are statistically independent.[2]

With this assumption, Eq. (9.37) becomes

$$\begin{aligned} E\{\mathbf{w}_{n+1}\} &= E\{\mathbf{w}_n\} + \mu E\{d(n)\mathbf{x}^*(n)\} - \mu E\{\mathbf{x}^*(n)\mathbf{x}^T(n)\}E\{\mathbf{w}_n\} \\ &= (\mathbf{I} - \mu\mathbf{R}_x)E\{\mathbf{w}_n\} + \mu\mathbf{r}_{dx} \end{aligned} \tag{9.38}$$

which is the same as Eq. (9.16) for the weight vector in the steepest descent algorithm. Therefore, the analysis for the steepest descent algorithm is applicable to $E\{\mathbf{w}_{n+1}\}$. In particular, it follows from Eq. (9.22) that

$$E\{\mathbf{u}_n\} = (\mathbf{I} - \mu\mathbf{\Lambda})^n\mathbf{u}_0 \tag{9.39}$$

where

$$\mathbf{u}_n = \mathbf{V}^H[\mathbf{w}_n - \mathbf{w}]$$

Since $\mathbf{w}_n$ will converge in the mean to $\mathbf{w}$ if $E\{\mathbf{u}_n\}$ converges to zero, then we have the following property:

Property 2. For jointly wide-sense stationary processes, the LMS algorithm *converges in the mean* if

$$0 < \mu < \frac{2}{\lambda_{\max}} \tag{9.40}$$

and the independence assumption is satisfied.

Although Eq. (9.40) places a bound on the step size for convergence in the mean, this bound is of limited use for two reasons. First, it is generally acknowledged that the upper bound is too large to ensure stability of the LMS algorithm since it is not sufficient to guarantee that the coefficient vector will remain bounded for all n. For example, although this bound ensures that $E\{\mathbf{w}_n\}$ converges, it places no constraints on how large the variance of $\mathbf{w}_n$ may become. Second, since the upper bound is expressed in terms of the largest eigenvalue of $\mathbf{R}_x$, using this bound requires that $\mathbf{R}_x$ be known. If this matrix is unknown, then it becomes necessary to estimate $\lambda_{\max}$. One way around this difficulty is to use the fact that $\lambda_{\max}$ may be upper bounded by the trace of $\mathbf{R}_x$,

$$\lambda_{\max} \leq \sum_{k=0}^{p} \lambda_k = \text{tr}(\mathbf{R}_x)$$

Therefore, if $x(n)$ is wide-sense stationary, then $\mathbf{R}_x$ is Toeplitz and the trace becomes

$$\text{tr}(\mathbf{R}_x) = (p+1)r_x(0) = (p+1)E\{|x(n)|^2\}$$

[2]Since $\mathbf{w}_n$ depends on the previous input vectors $\mathbf{x}(n-1), \mathbf{x}(n-2), \ldots$, this assumption can only be approximately true. However, experience with the LMS algorithm shows that this assumption leads to convergence properties that are generally in close agreement with experiments and computer simulations.

As a result, Eq. (9.40) may be replaced with the more conservative bound

$$\boxed{0 < \mu < \frac{2}{(p+1)E\{|x(n)|^2\}}} \tag{9.41}$$

Although we have simply replaced one unknown with another, $E\{|x(n)|^2\}$ is more easily estimated since it represents the power in $x(n)$. For example, $E\{|x(n)|^2\}$ could be estimated using an average such as

$$\hat{E}\{|x(n)|^2\} = \frac{1}{N}\sum_{k=0}^{N-1}|x(n-k)|^2$$

In the following example we consider the use of an LMS adaptive filter for linear prediction.

Example 9.2.1 *Adaptive Linear Prediction Using the LMS Algorithm*

Let $x(n)$ be a second-order autoregressive process that is generated according to the difference equation

$$x(n) = 1.2728x(n-1) - 0.81x(n-2) + v(n) \tag{9.42}$$

where $v(n)$ is unit variance white noise. As we saw in Section 7.3.4, the optimum causal linear predictor for $x(n)$ is

$$\hat{x}(n) = 1.2728x(n-1) - 0.81x(n-2)$$

However, in order to design this predictor (i.e., to know that the optimum predictor coefficients are 1.2728 and -0.81) it is necessary to know the autocorrelation sequence of $x(n)$. Therefore, suppose we consider an adaptive linear predictor of the form:

$$\hat{x}(n) = w_n(1)x(n-1) + w_n(2)x(n-2)$$

as shown in Fig. 9.9. With the LMS algorithm, the predictor coefficients $w_n(k)$ are updated as follows:

$$w_{n+1}(k) = w_n(k) + \mu e(n)x^*(n-k)$$

If the step size μ is sufficiently small, then the coefficients $w_n(1)$ and $w_n(2)$ will converge in the mean to their optimum values, $w(1) = 1.2728$ and $w(2) = -0.81$, respectively. Note that the prediction error is

$$e(n) = x(n) - \hat{x}(n) = \big[1.2728 - w_n(1)\big]x(n-1) + \big[-0.81 - w_n(1)\big]x(n-2) + v(n)$$

Therefore, when $w_n(1) = 1.2728$ and $w_n(2) = -0.81$, the error becomes $e(n) = v(n)$, and the minimum mean-square error is[3]

$$\xi_{\min} = \sigma_v^2 = 1$$

Although we might expect the mean-square error $E\{|e(n)|^2\}$ to converge to $\xi_{\min}$ as $\mathbf{w}_n$ converges to $\mathbf{w}$, as we will soon discover, this is not the case.

To see how this adaptive linear predictor behaves in practice, suppose that the weight vector is initialized to zero, $\mathbf{w}_0 = \mathbf{0}$, and that the step size is $\mu = 0.02$. Shown in Fig. 9.10*a*

[3]This may also be shown by evaluating the expression for the minimum mean-square error given in Eq. (9.26).

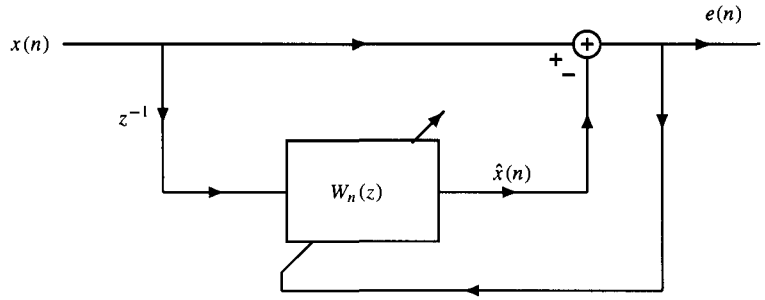


Figure 9.9 *An adaptive filter for linear prediction.*

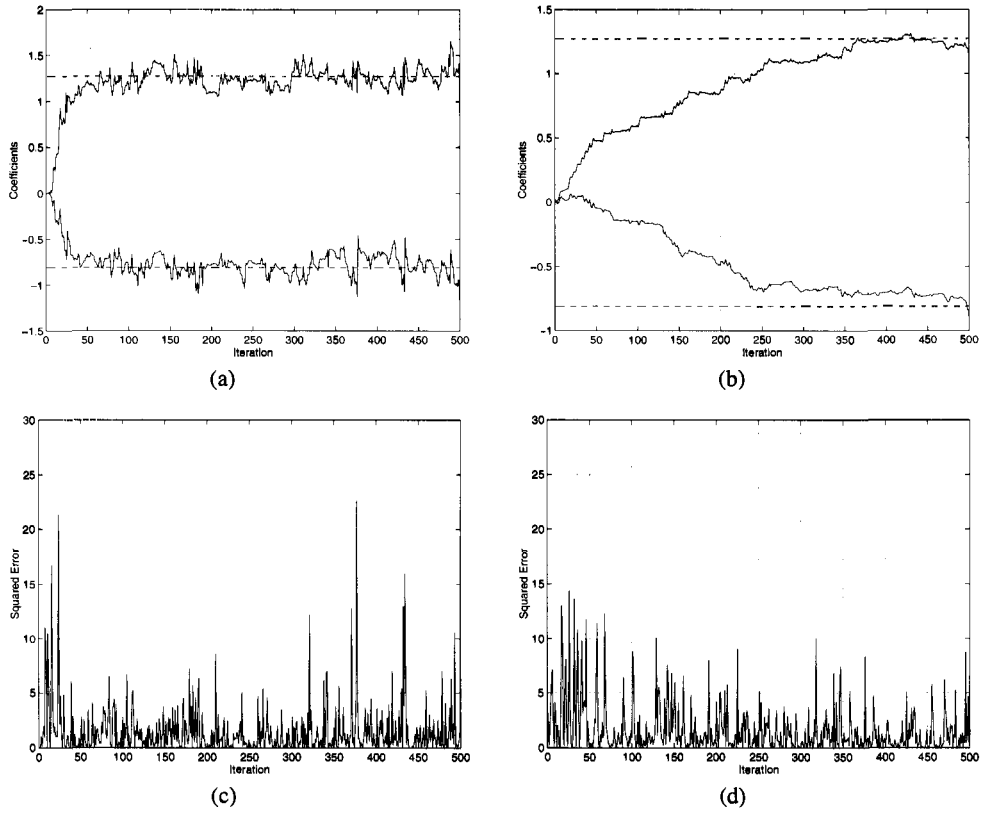


Figure 9.10 *Performance of a two-coefficient LMS adaptive linear predictor. The trajectories of the predictor coefficients are shown for step sizes of (a)* $\mu = 0.02$ *and (b)* $\mu = 0.004$ *with the correct values indicated by the dashed lines. A plot of the squared error,* $e^2(n)$*, is shown in (c) and (d) for step sizes of* $\mu = 0.02$ *and* $\mu = 0.004$*, respectively.*

are the trajectories of $w_n(1)$ and $w_n(2)$ versus n. As we see, there is a great deal of fluctuation in the weights, even after they have converged to within a neighborhood of their steady-state values. By contrast, shown in Fig. 9.10*b* are the trajectories of the predictor coefficients for a step size $\mu = 0.004$. Compared to a step size of $\mu = 0.02$, we see that, although the weights take longer to converge, the trajectories are much smoother, illustrating the basic trade-off

between rate of convergence and stability (variance) of the final solution. Shown in parts *c* and *d* are plots of the squared error $e^2(n)$ for $\mu = 0.02$ and $\mu = 0.004$, respectively. What is so striking in these plots is the large amount of variation in $e^2(n)$ versus n compared to the learning curves for the steepest descent algorithm.

Before leaving this example, let us compute the maximum step size that is allowed for the adaptive linear predictor to converge in the mean. Given the coefficients of the filter that are used to generate $x(n)$, we may use the inverse Levinson-Durbin recursion to find the autocorrelation sequence $r_x(k)$. Using the m-file `ator.m` we find

$$r_x(0) = 5.7523 \quad ; \quad r_x(1) = 4.0450$$

Therefore, $\lambda_{\max} = r_x(0) + r_x(1) = 9.7973$ and the bound on the step size is

$$0 < \mu < 0.2041$$

Note that the step sizes used in Fig. 9.10 are at least an order of magnitude smaller than the largest value allowed for convergence in the mean. This is typically the case and, in fact, with a step size much larger than $\mu = 0.02$, the coefficients $w_n(k)$ begin to fluctuate wildly and eventually become unstable.

As we observed in the previous example, as the weight vector begins to converge in the mean, the coefficients begin to fluctuate about their optimum values. These fluctuations are due to the noisy gradient vectors that are used to form the corrections to $\mathbf{w}_n$. As a result, the variance of the weight error vector does not go to zero and the mean-square error is larger than the minimum mean-square error by an amount referred to as the *excess mean-square error*. This behavior is illustrated in Fig. 9.8*b* which shows that, as $\mathbf{w}_n$ oscillates about $\mathbf{w} = \mathbf{R}_x^{-1}\mathbf{r}_{dx}$, the corresponding mean-square error $\xi(n)$ has a value that, on the average, exceeds the minimum mean-square error. In order to quantify this excess mean-square error, we will write the error at time n as follows:

$$e(n) = d(n) - \mathbf{w}_n^T\mathbf{x}(n) = d(n) - \left(\mathbf{w} + \mathbf{c}_n\right)^T\mathbf{x}(n) = e_{\min}(n) - \mathbf{c}_n^T\mathbf{x}(n)$$

where $e_{\min}(n)$ is the error that would occur if the optimum filter coefficients were used, i.e.,

$$e_{\min}(n) = d(n) - \mathbf{w}^T\mathbf{x}(n)$$

Assuming that the filter is in the steady-state with $E\{\mathbf{c}_n\} = 0$, the mean-square error may be expressed as

$$\xi(n) = E\{|e(n)|^2\} = \xi_{\min} + \xi_{\text{ex}}(n)$$

where

$$\xi_{\min} = E\{|e_{\min}(n)|^2\}$$

is the minimum mean-square error and $\xi_{ex}(n)$ is the *excess mean-square error*, which depends on the statistics of $\mathbf{x}(n)$, $\mathbf{c}_n$, and $d(n)$. Although $\xi_{ex}(n)$ is not easy to evaluate, by invoking the independence assumption, the following property may be established with a fair amount of effort [20].

Property 3. The mean-square error $\xi(n)$ converges to a steady-state value of

$$\xi(\infty) = \xi_{\min} + \xi_{\text{ex}}(\infty) = \xi_{\min} \frac{1}{1 - \mu \sum_{k=0}^{p} \frac{\lambda_k}{2 - \mu\lambda_k}} \tag{9.43}$$

and the LMS algorithm is said to *converge in the mean-square* if and only if the step-size μ satisfies the following two conditions:

$$0 < \mu < \frac{2}{\lambda_{\max}} \tag{9.44}$$

$$\mu \sum_{k=0}^{p} \frac{\lambda_k}{2 - \mu\lambda_k} < 1 \tag{9.45}$$

Note that Eq. (9.44) is the condition that is required for the LMS algorithm to converge in the mean, and Eq. (9.45) guarantees that $\xi(\infty)$ is positive. Solving Eq. (9.43) for $\xi_{ex}(\infty)$ we find

$$\xi_{\text{ex}}(\infty) = \mu\, \xi_{\min} \frac{\sum_{k=0}^{p} \frac{\lambda_k}{2 - \mu\lambda_k}}{1 - \mu \sum_{k=0}^{p} \frac{\lambda_k}{2 - \mu\lambda_k}} \tag{9.46}$$

If $\mu \ll 2/\lambda_{\max}$, as is typically the case, then $\mu\lambda_k \ll 2$ and Eq. (9.45) may be simplified to

$$\frac{1}{2}\mu \sum_{k=0}^{p} \lambda_k < 1$$

or,

$$\mu < \frac{2}{\text{tr}(\mathbf{R}_x)}$$

When $\mu \ll 2/\lambda_{\max}$ it also follows that

$$\xi(\infty) \approx \xi_{\min} \frac{1}{1 - \frac{1}{2}\mu\, \text{tr}(\mathbf{R}_x)}$$

and the excess mean-square error given in Eq. (9.46) is approximately

$$\xi_{\text{ex}}(\infty) \approx \mu\, \xi_{\min} \frac{\frac{1}{2}\text{tr}(\mathbf{R}_x)}{1 - \frac{1}{2}\mu\, \text{tr}(\mathbf{R}_x)} \approx \frac{1}{2}\mu\, \xi_{\min}\, \text{tr}(\mathbf{R}_x) \tag{9.47}$$

Thus, for small μ, the excess mean-square error is proportional to the step size μ. Adaptive filters may be described in terms of their *misadjustment*, which is a normalized mean-square error that is defined as follows.

Definition. The misadjustment $\mathcal{M}$ is the ratio of the steady-state excess mean-square error to the minimum mean-square error,

$$\mathcal{M} = \frac{\xi_{ex}(\infty)}{\xi_{min}}$$

From Eq. (9.47) we see that if the step size is small, $\mu \ll 2/\lambda_{max}$, then the misadjustment is approximately

$$\mathcal{M} \approx \mu \frac{\frac{1}{2}\text{tr}(\mathbf{R}_x)}{1 - \frac{1}{2}\mu\ \text{tr}(\mathbf{R}_x)} \approx \frac{1}{2}\mu\ \text{tr}(\mathbf{R}_x)$$

Example 9.2.2 *LMS Misadjustment*

In this example, we look at the learning curves for the adaptive linear predictor considered in Example 9.2.1, and evaluate the excess mean-square error and the misadjustment for different step sizes. Since the learning curve is a plot of $\xi(n) = E\{|e(n)|^2\}$ versus n, we may approximate the learning curve by averaging plots of $|e(n)|^2$ that are obtained by repeatedly implementing the adaptive predictor. For example, implementing the adaptive predictor K times, and denoting the squared error at time n on the kth trial by $|e_k(n)|^2$, we have

$$\hat{\xi}(n) = \hat{E}\{|e(n)|^2\} = \frac{1}{K}\sum_{k=1}^{K}|e_k(n)|^2$$

With $K = 200$, an initial weight vector of zero, and step sizes of $\mu = 0.02$ and $\mu = 0.004$, these estimates of the learning curves are shown in Fig. 9.11. One property of the LMS algorithm that we are able to observe from these plots is that, when the step size is decreased, the convergence of the adaptive filter to its steady-state value is slower, but the average steady-state squared error is smaller.

We may estimate the steady-state mean-square error from these plots by averaging $\hat{\xi}(n)$ over n after the LMS algorithm has reached steady-state. For example, with

$$\hat{\xi}(\infty) = \frac{1}{100}\sum_{n=901}^{1000}\hat{E}\{|e(n)|^2\}$$

we find

$$\hat{\xi}(\infty) = \begin{cases} 1.1942 & ; \quad \text{for } \mu = 0.02 \\ 1.0155 & ; \quad \text{for } \mu = 0.004 \end{cases}$$

We may compare these results to the theoretical steady-state mean-square error using Eq. (9.43). With $\xi_{min} = 1$, and eigenvalues $\lambda_1 = 9.7924$ and $\lambda_2 = 1.7073$ (see Example 9.2.1), it follows that

$$\xi(\infty) = \begin{cases} 1.1441 & ; \quad \text{for } \mu = 0.02 \\ 1.0240 & ; \quad \text{for } \mu = 0.004 \end{cases}$$

which is in fairly close agreement with the estimated values given above.

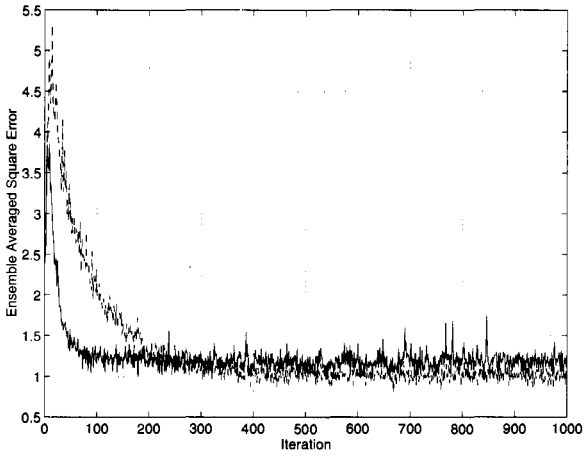


Figure 9.11 *Approximations to the learning curves for a second-order LMS adaptive linear predictor using step sizes of* $\mu = 0.02$ *(solid line) and* $\mu = 0.004$ *(dotted line).*

9.2.4 Normalized LMS

As we have seen, one of the difficulties in the design and implementation of the LMS adaptive filter is the selection of the step size μ. For stationary processes, the LMS algorithm converges in the mean if $0 < \mu < 2/\lambda_{\max}$, and converges in the mean-square if $0 < \mu < 2/\text{tr}(\mathbf{R}_x)$. However, since $\mathbf{R}_x$ is generally unknown, then either $\lambda_{\max}$ or $\mathbf{R}_x$ must be estimated in order to use these bounds. One way around this difficulty is to use the fact that, for stationary processes, $\text{tr}(\mathbf{R}_x) = (p+1)E\{|x(n)|^2\}$. Therefore, the condition for mean-square convergence may be replaced with

$$0 < \mu < \frac{2}{(p+1)E\{|x(n)|^2\}}$$

where $E\{|x(n)|^2\}$ is the power in the process $x(n)$. This power may be estimated using a time average such as[4]

$$\hat{E}\{|x(n)|^2\} = \frac{1}{p+1}\sum_{k=0}^{p}|x(n-k)|^2$$

which leads to the following bound on the step size for mean-square convergence:

$$0 < \mu < \frac{2}{\mathbf{x}^H(n)\mathbf{x}(n)}$$

A convenient way to incorporate this bound into the LMS adaptive filter is to use a (time-varying) step size of the form

$$\mu(n) = \frac{\beta}{\mathbf{x}^H(n)\mathbf{x}(n)} = \frac{\beta}{\|\mathbf{x}(n)\|^2} \tag{9.48}$$

where β is a *normalized step size* with $0 < \beta < 2$. Replacing μ in the LMS weight vector update equation with $\mu(n)$ leads to the Normalized LMS algorithm (NLMS), which is given

[4]Since this estimate uses only those values of $x(n)$ that are within the tapped delay line at time n, no extra memory is required to evaluate this sum.

The Normalized LMS Algorithm

```
function [A,E] = nlms(x,d,beta,nord,a0)
%
X=convm(x,nord);
[M,N] = size(X);
if nargin < 5,   a0 = zeros(1,N);   end
a0 = a0(:).';
E(1) = d(1) - a0*X(1,:).';
DEN=X(1,:)*X(1,:)' + 0.0001;
A(1,:) = a0 + beta/DEN*E(1)*conj(X(1,:));
if M>1
for k=2:M-nord+1;
    E(k) = d(k) - A(k-1,:)*X(k,:).';
    DEN=X(k,:)*X(k,:)' + 0.0001;
    A(k,:) = A(k-1,:) + beta/DEN*E(k)*conj(X(k,:));
    end;
end;
```

Figure 9.12 *A* MATLAB *program for the Normalized LMS algorithm.*

by

$$\mathbf{w}_{n+1} = \mathbf{w}_n + \beta \frac{\mathbf{x}^*(n)}{\|\mathbf{x}(n)\|^2} e(n) \tag{9.49}$$

Note that the effect of the normalization by $\|\mathbf{x}(n)\|^2$ is to alter the magnitude, but not the *direction*, of the estimated gradient vector. Therefore, with the appropriate set of statistical assumptions it may be shown that the normalized LMS algorithm converges in the mean-square if $0 < \beta < 2$ [4,34].

In the LMS algorithm, the correction that is applied to $\mathbf{w}_n$ is proportional to the input vector $\mathbf{x}(n)$. Therefore, when $\mathbf{x}(n)$ is large, the LMS algorithm experiences a problem with *gradient noise amplification*. With the normalization of the LMS step size by $\|\mathbf{x}(n)\|^2$ in the NLMS algorithm, however, this noise amplification problem is diminished. Although the NLMS algorithm bypasses the problem of noise amplification, we are now faced with a similar problem that occurs when $\|\mathbf{x}(n)\|$ becomes too small. An alternative, therefore, is to use the following modification to the NLMS algorithm:

$$\mathbf{w}_{n+1} = \mathbf{w}_n + \beta \frac{\mathbf{x}^*(n)}{\epsilon + \|\mathbf{x}(n)\|^2} e(n) \tag{9.50}$$

where ϵ is some small positive number. A MATLAB program for the normalized LMS algorithm is given in Fig. 9.12.

Compared with the LMS algorithm, the normalized LMS algorithm requires additional computation to evaluate the normalization term $\|\mathbf{x}(n)\|^2$. However, if this term is evaluated recursively as follows

$$\|\mathbf{x}(n+1)\|^2 = \|\mathbf{x}(n)\|^2 + |x(n+1)|^2 - |x(n-p)|^2 \tag{9.51}$$

then the extra computation involves only two squaring operations, one addition, and one substraction.

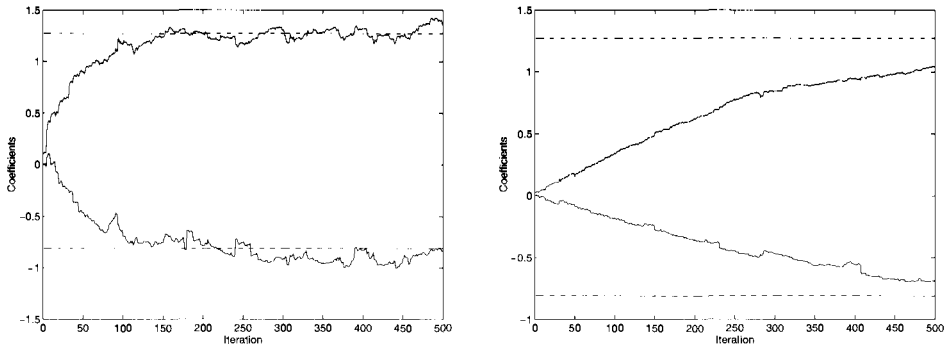


Figure 9.13 *A two-coefficient normalized LMS adaptive linear predictor. The trajectories of the predictor coefficients are shown for step sizes of (a)* $\beta = 0.05$ *and (b)* $\beta = 0.01$ *with the correct values for the coefficients indicated by the dashed lines.*

Example 9.2.3 ***Adaptive Linear Prediction Using the NLMS Algorithm***

In this example, we consider once again the problem of adaptive linear prediction. This time, however, we will use the normalized LMS algorithm. As in Example 9.2.1, the process that is to be predicted is the AR(2) process that is generated by the difference equation

$$x(n) = 1.2728x(n-1) - 0.81x(n-2) + v(n)$$

where $v(n)$ is unit variance white noise. With a two-coefficient LMS adaptive predictor we have

$$\hat{x}(n) = w_n(1)x(n-1) + w_n(2)x(n-2)$$

where the predictor coefficients are updated according to

$$w_{n+1}(k) = w_n(k) + \beta \frac{x(n-k)}{\epsilon + x^2(n-1) + x^2(n-2)} e(n) \quad ; \quad k = 1, 2$$

Using normalized step sizes of $\beta = 0.05$ and $\beta = 0.01$, with $\epsilon = 0.0001$ and $w_0(1) = w_0(2) = 0$ we obtain the sequence of predictor coefficients shown in Fig. 9.13*a* and *b*, respectively. Comparing these results to those shown in Fig. 9.10 we see that the trajectories of the coefficients are similar. The difference, however, is that for the NLMS algorithm it is not necessary to estimate $\lambda_{\max}$ in order to select a step size.

9.2.5 Application: Noise Cancellation

In Section 7.2.3, we looked at the problem of *noise cancellation* in which a process $d(n)$ is to be estimated from a noise corrupted observation

$$x(n) = d(n) + v_1(n)$$

Clearly, without any information about $d(n)$ or $v_1(n)$ it is not possible to separate the signal from the noise. However, given a *reference signal*, $v_2(n)$, that is correlated with $v_1(n)$, then this reference signal may be used to estimate the noise $v_1(n)$, and this estimate may then be subtracted from $x(n)$ to form an estimate of $d(n)$,

$$\hat{d}(n) = x(n) - \hat{v}_1(n)$$

For example, if $d(n)$, $v_1(n)$, and $v_2(n)$ are jointly wide-sense stationary processes, and if the autocorrelation $r_{v_2}(k)$ and the cross-correlation $r_{v_1v_2}(k)$ are known, then a Wiener filter may be designed to find the minimum mean-square estimate of $v_1(n)$ as illustrated in Fig. 9.14*a*. In practice, however, a stationarity assumption is not generally appropriate and, even if it were, the required statistics of $v_2(n)$ and $v_1(n)$ are generally unknown. Therefore, as an alternative to the Wiener filter, let us consider the adaptive noise canceller shown in Fig. 9.14*b*. If the reference signal $v_2(n)$ is uncorrelated with $d(n)$, then it follows from the

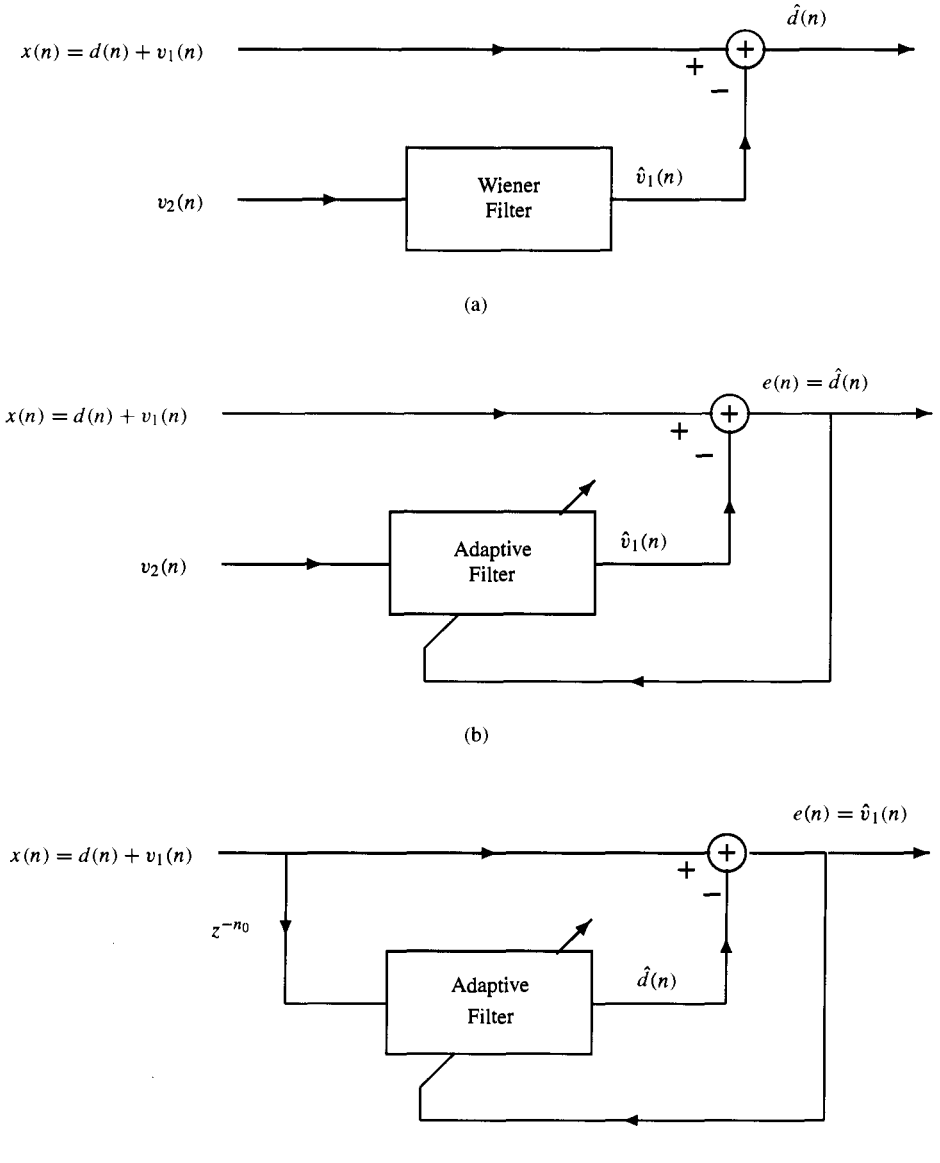


Figure 9.14 *Noise cancellation using (a) a Wiener filter, (b) an adaptive noise canceller with a reference signal v(n), and (c) adaptive noise cancellation without a reference.*

discussion in Section 9.1 that minimizing the mean-square error $E\{|e(n)|^2\}$ is equivalent to minimizing $E\{|v_1(n) - \hat{v}_1(n)|^2\}$. In other words, the output of the adaptive filter is the minimum mean-square estimate of $v_1(n)$. Basically, if there is no information about $d(n)$ in the reference signal $v_2(n)$, then the best that the adaptive filter can do to minimize $E\{|e(n)|^2\}$ is to remove the part of $e(n)$ that may be estimated from $v_2(n)$, which is $v_1(n)$. Since the output of the adaptive filter is the minimum mean-square estimate of $v_1(n)$, then it follows that $e(n)$ is the minimum mean-square estimate of $d(n)$.

As a specific example, let us reconsider the problem presented in Example 7.2.6 in which the signal to be estimated is a sinusoid,

$$d(n) = \sin(n\omega_0 + \phi)$$

with $\omega_0 = 0.05\pi$, and the noise sequences $v_1(n)$ and $v_2(n)$ are generated by the first-order difference equations

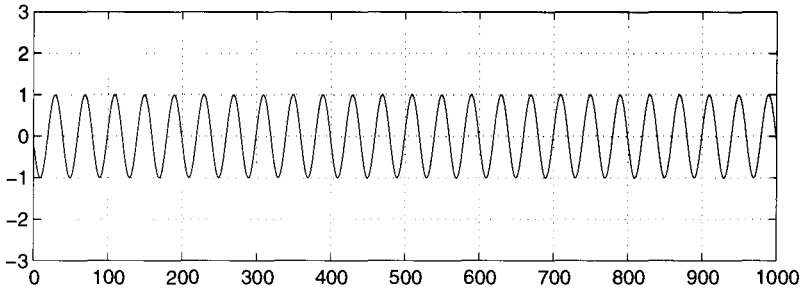
$$\begin{aligned} v_1(n) &= 0.8v_1(n-1) + g(n) \\ v_2(n) &= -0.6v_2(n-1) + g(n) \end{aligned} \tag{9.52}$$

where $g(n)$ is a zero-mean, unit variance white noise process that is uncorrelated with $d(n)$. Shown in Fig. 9.15*a* is a plot of 1000 samples of the sinusoid and in Fig. 9.15*b* is the noisy signal $x(n) = d(n) + v_1(n)$. The reference signal $v_2(n)$ that is used to estimate $v_1(n)$ is shown in Fig. 9.15*c*. Using a 12th-order adaptive noise canceller with coefficients that are updated using the normalized LMS algorithm, the estimate of $d(n)$ that is produced with a step size $\beta = 0.25$ is shown in Fig. 9.15*d*. As we see from this figure, after about 100 iterations the adaptive filter is producing a fairly accurate estimate of $d(n)$ and, after about 200 iterations the adaptive filter appears to have settled down into its steady-state behavior. Although not quite as good as the estimate that is produced with a 6th-order Wiener filter as shown in Fig. 7.9*d* on p. 351, the adaptive noise canceller, unlike the Wiener filter, does not require any statistical information.

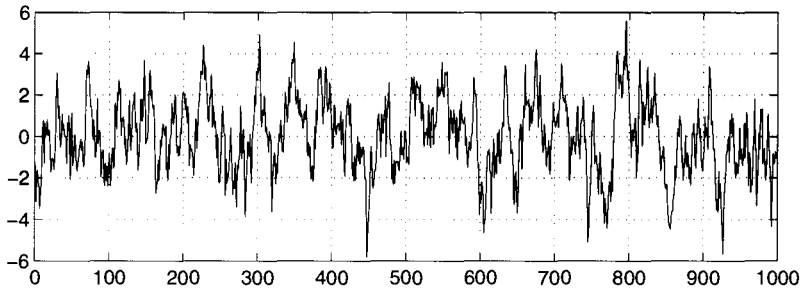
One of the advantages of this adaptive noise canceller over a Wiener filter is that it may be used when the processes are nonstationary. For example, let $v_1(n)$ and $v_2(n)$ be nonstationary processes that are generated by the first-order difference equations given in Eq. (9.52), where $g(n)$ is nonstationary white noise with a variance that increases linearly from $\sigma_g^2(0) = 0.25$ to $\sigma_g^2(1000) = 6.25$. As before, $d(n)$ is estimated using a 12th-order adaptive noise canceller with coefficients that are updated according to the normalized LMS algorithm. Shown in Fig. 9.16*a* is the desired signal $d(n)$ and in Fig. 9.16*b* is the noisy signal $x(n)$. The increasing variance in the additive noise is clearly evident. The nonstationary reference signal $v_2(n)$ that is used to estimate $v_1(n)$ is shown in Fig. 9.16*c* and in Fig. 9.16*d* is the estimate of $d(n)$ that is produced by the adaptive filter. As we see in this figure, the performance of the adaptive noise canceller is not significantly affected by the nonstationarity of the noise (note that for $n > 250$ the variance of the nonstationary noise is larger than the variance of the noise in Fig. 9.15).

The key to the successful operation of the adaptive noise canceller in Fig. 9.14*b* is the availability of a reference signal, $v_2(n)$, that may be used to estimate the additive noise $v_1(n)$. Unfortunately, in many applications a reference signal is not available and another approach must be considered. In some cases, however, it is possible to derive a reference signal by simply delaying the process $x(n) = d(n) + v_1(n)$. For example, suppose that $d(n)$ is a narrowband process and that $v_1(n)$ is a broadband process with

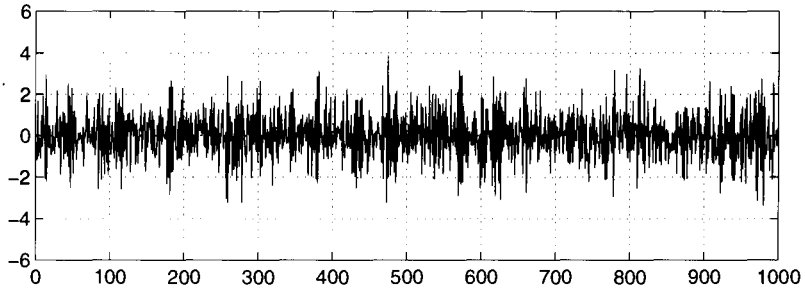
$$E\{v_1(n)v_1(n-k)\} = 0 \quad ; \quad |k| > k_0$$



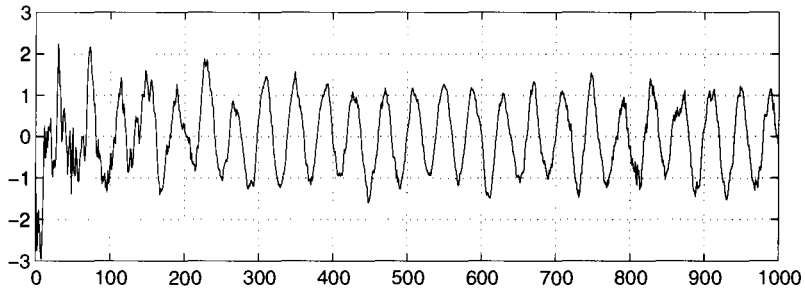
(a)



(b)

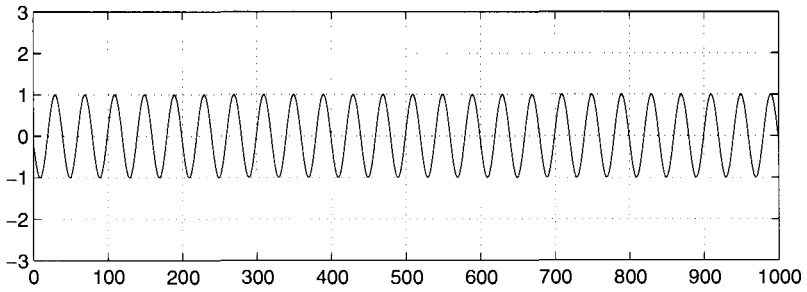


(c)

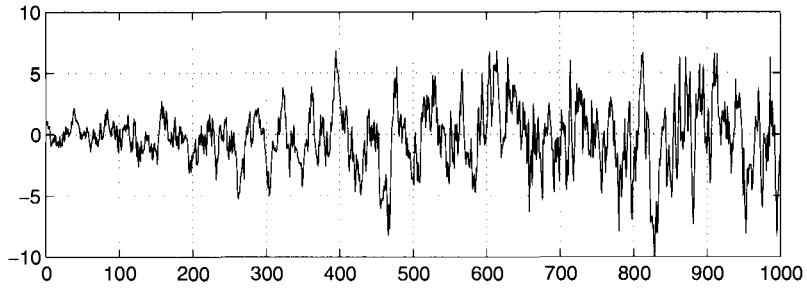


(d)

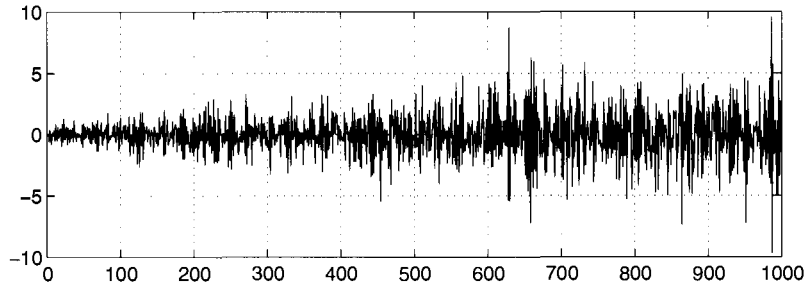
Figure 9.15 *Noise cancellation example. (a) A sinusoid that is to be estimated. (b) Noise corrupted sinusoid. (c) The reference signal used by the secondary sensor. (d) The output of a sixth-order NLMS adaptive noise canceller with a normalized step size* $\beta = 0.25$.



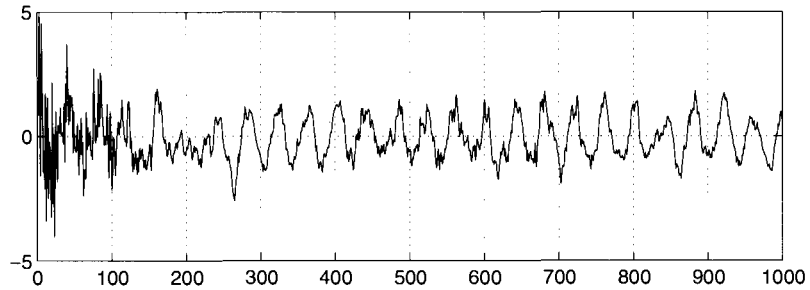
(a)



(b)

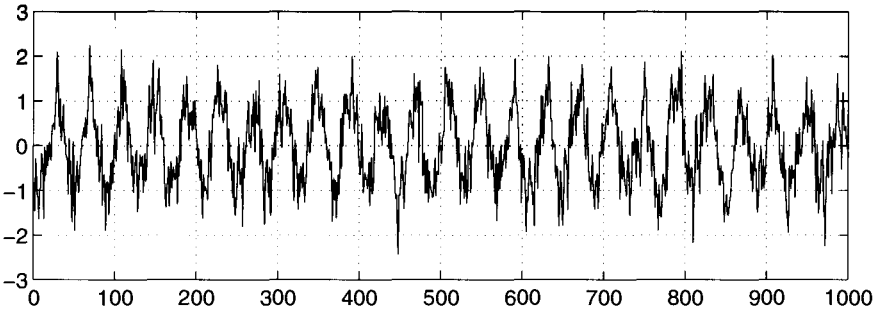


(c)

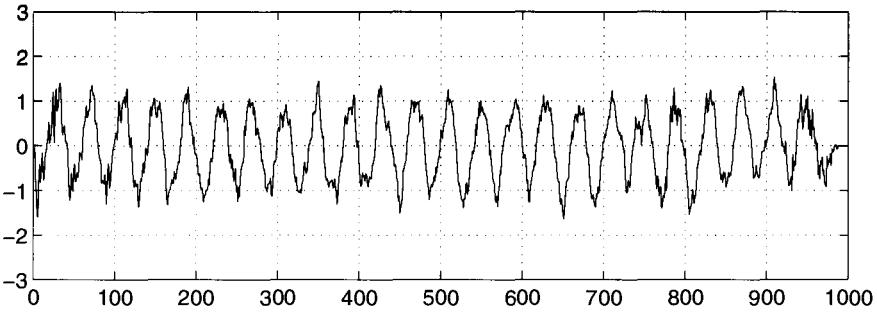


(d)

Figure 9.16 *Noise cancellation of nonstationary noise. (a) Desired signal that is to be estimated. (b) Noise corrupted sinusoid. (c) The reference signal used by secondary sensor. (d) The output of a 12th-order NLMS adaptive noise canceller with a normalized step size* $\beta = 0.25$.



(a)



(b)

Figure 9.17 *Noise cancellation without a reference signal. (a) The noisy process $x(n)$ and (b) the output of a 12th-order NLMS adaptive noise canceller with $\beta = 0.25$ and $n_0 = 25$.*

If $d(n)$ and $v_1(n)$ are uncorrelated, then

$$E\{v_1(n)x(n-k)\} = E\{v_1(n)d(n-k)\} + E\{v_1(n)v_1(n-k)\} = 0 \quad ; \quad |k| > k_0$$

Therefore, if $n_0 > k_0$ then the delayed process $x(n-n_0)$ will be uncorrelated with the noise $v_1(n)$, and correlated with $d(n)$ (with the assumption that $d(n)$ is a broadband process). Thus, $x(n-n_0)$ may be used as a reference signal to estimate $d(n)$ as illustrated in Fig. 9.14*c*. In contrast to the adaptive noise canceller in Fig. 9.14*b*, note that the adaptive filter in Fig. 9.14*c* produces an estimate of the broadband process, $d(n)$, and the error $e(n)$ corresponds to an estimate of the noise $v_1(n)$. As an example of using an adaptive noise canceller without a reference, let $x(n) = d(n)+v_1(n)$ where $d(n)$ is a sinusoid of frequency $\omega_0 = 0.05\pi$, and let $v_1(n)$ be the AR(1) process defined in Eq. (9.52) where $g(n)$ white noise with a variance of $\sigma_g^2 = 0.25$. Shown in Fig. 9.17*a* is the noisy process $x(n) = d(n) + v_1(n)$ and in Fig. 9.17*b* is the output of the adaptive filter using the NLMS algorithm with a normalized step size of $\beta = 0.25$, and a reference signal that is obtained by delaying $x(n)$ by $n_0 = 25$ samples.

9.2.6 Other LMS-Based Adaptive Filters

In addition to the NLMS algorithm, a number of other modifications to the LMS algorithm have been proposed and studied. Each of these modifications attempts to improve one or

more of the properties of the LMS algorithm. In this section, we provide a brief overview of some of these modifications, including the leaky LMS algorithm for dealing with the problems that occur when the autocorrelation matrix is singular, the block LMS algorithm and the sign algorithms for increasing the computational efficiency of the LMS algorithm, and the variable step-size (VS) algorithms for improving the speed of convergence.

The Leaky LMS Algorithm. When the input process to an adaptive filter has an autocorrelation matrix with zero eigenvalues, the LMS adaptive filter has one or more modes that are undriven and undamped. For example, it follows from Eq. (9.39) that if $\lambda_k = 0$, then

$$E\{u_n(k)\} = u_0(k)$$

which does not decay to zero with n. Since it is possible for these undamped modes to become unstable, it is important to stabilize the LMS adaptive filter by forcing these modes to zero [33]. One way to accomplish this is to introduce a leakage coefficient γ into the LMS algorithm as follows:

$$\boxed{\mathbf{w}_{n+1} = (1 - \mu\gamma)\mathbf{w}_n + \mu e(n)\mathbf{x}^*(n)} \tag{9.53}$$

where $0 < \gamma \ll 1$. The effect of this *leakage coefficient* is to force the filter coefficients to zero if either the error $e(n)$ or the input $x(n)$ become zero, and to force any undamped modes of the system to zero.

The properties of the leaky LMS algorithm may be derived by examining the behavior of $E\{\mathbf{w}_n\}$. If we substitute Eq. (9.7) for $e(n)$ into Eq. (9.53), then we have

$$\mathbf{w}_{n+1} = \left[\mathbf{I} - \mu\left\{\mathbf{x}^*(n)\mathbf{x}^T(n) + \gamma\mathbf{I}\right\}\right]\mathbf{w}_n + \mu d(n)\mathbf{x}^*(n)$$

Taking the expected value and using the independence assumption yields

$$E\{\mathbf{w}_{n+1}\} = \left[\mathbf{I} - \mu\left\{\mathbf{R}_x + \gamma\mathbf{I}\right\}\right]E\{\mathbf{w}_n\} + \mu\mathbf{r}_{dx} \tag{9.54}$$

Comparing Eq. (9.54) to Eq. (9.38) we see that the autocorrelation matrix $\mathbf{R}_x$ in the LMS algorithm has been replaced with $\mathbf{R}_x + \gamma\mathbf{I}$. Therefore, the coefficient leakage term effectively adds white noise to $x(n)$ by adding γ to the main diagonal of the autocorrelation matrix. Since the eigenvalues of $\mathbf{R}_x + \gamma\mathbf{I}$ are $\lambda_k + \gamma$ and since $\lambda_k \geq 0$, then none of the modes of the leaky LMS algorithm will be undamped. In addition, the constraint on the step size μ for convergence in the mean becomes

$$0 < \mu < \frac{2}{\lambda_{\max} + \gamma}$$

The drawback to the leaky LMS algorithm is that, for stationary processes, the steady-state solution will be biased. Specifically, note that if $\mathbf{w}_n$ converges in the mean then

$$\lim_{n\to\infty} E\{\mathbf{w}_n\} = (\mathbf{R}_x + \gamma\mathbf{I})^{-1}\mathbf{r}_{dx}$$

Thus, the leakage coefficient introduces a bias into the steady-state solution.

It is interesting to note that the leaky LMS algorithm may also be derived by using the LMS gradient descent algorithm to minimize the error

$$\xi(n) = |e(n)|^2 + \gamma\|\mathbf{w}_n\|^2$$

Specifically, since the gradient of $\xi(n)$ is

$$\nabla\xi(n) = -e(n)\mathbf{x}^*(n) + \gamma\mathbf{w}_n$$

then the gradient descent algorithm becomes

$$\mathbf{w}_{n+1} = \mathbf{w}_n - \mu\nabla\xi(n) = \mathbf{w}_n - \mu\gamma\mathbf{w}_n + \mu e(n)\mathbf{x}^*(n)$$

which is the same as Eq. (9.53).

LMS Algorithms With Reduced Complexity. In spite of the computational efficiency of the LMS algorithm, additional simplifications may be necessary in some applications, such as high speed digital communications. There are several approaches that may be used to reduce the computational requirements of the LMS algorithm. One of these is the block LMS algorithm, which is identical to the LMS algorithm except that the filter coefficients are updated only once for each block of L samples [9]. In other words, the filter coefficients are held constant over each block of L samples, and the filter output $y(n)$ and the error $e(n)$ for each value of n within the block are calculated using the filter coefficients for that block. Then, at the end of each block, the coefficients are updated using an average of the L gradient estimates over the block. Thus, the update equation for the coefficients in the kth block is

$$\mathbf{w}_{(k+1)L} = \mathbf{w}_{kL} + \mu\frac{1}{L}\sum_{l=0}^{L-1} e(kL+l)\mathbf{x}^*(kL+l)$$

where the output for the kth block is

$$y(kL+l) = \mathbf{w}_{kL}^T\mathbf{x}(kL+l) \quad ; \quad l = 0, 1, \ldots, L-1$$

and the error is

$$e(kL+l) = d(kL+l) - \mathbf{w}_{kL}^T\mathbf{x}(kL+l) \quad ; \quad l = 0, 1, \ldots, L-1$$

Since $y(n)$ over each block of L values is the convolution of the weight vector $\mathbf{w}_{kL}$ with a block of input samples, the efficiency of the block LMS algorithm comes from using an FFT to perform this block convolution [12].

Another set of simplifications to the LMS algorithm are found in the *sign algorithms*. In these algorithms, the LMS coefficient update equation is modified by applying the sign operator to either the error $e(n)$, the data $x(n)$, or both the error and the data. For example, assuming that $x(n)$ and $d(n)$ are real-valued processes, the *sign-error* algorithm is

$$\boxed{\mathbf{w}_{n+1} = \mathbf{w}_n + \mu\,\mathrm{sgn}\{e(n)\}\mathbf{x}(n)} \tag{9.55}$$

where

$$\mathrm{sgn}\{e(n)\} = \begin{cases} 1 & ; \quad e(n) > 0 \\ 0 & ; \quad e(n) = 0 \\ -1 & ; \quad e(n) < 0 \end{cases}$$

Note that the sign-error algorithm may be viewed as the result of applying a two-level quantizer to the error. The simplification in the sign-error algorithm comes when the step size is chosen to be a power of two, $\mu = 2^{-l}$. In this case, the coefficient update equation may be implemented using $p + 1$ data shifts instead of $p + 1$ multiplies. Compared to

the LMS algorithm, the sign-error algorithm uses a noisy estimate of the gradient. Since replacing $e(n)$ with the sign of the error changes only the magnitude of the correction that is used to update $\mathbf{w}_n$ and does not alter the direction, the sign-error algorithm is equivalent to the LMS algorithm with a step size that is inversely proportional to the magnitude of the error.

It is interesting to note that the sign error algorithm may also be derived by using the LMS gradient descent algorithm to minimize the error

$$\xi(n) = |e(n)|$$

To show this, note that

$$\frac{\partial |e(n)|}{\partial w_n(k)} = \text{sgn}\{e(n)\}\frac{\partial e(n)}{\partial w_n(k)} = \text{sgn}\{e(n)\}x(n-k)$$

Therefore, the gradient vector is

$$\nabla\xi(n) = \text{sgn}\{e(n)\}\mathbf{x}(n)$$

and the result follows. Thus, the sign-error algorithm is sometimes referred to as the Least Mean Absolute Value (LMAV) algorithm [3].

Along the same lines, instead of using the sign of the error, the computational requirements of the LMS algorithm may be simplified by using the sign of the data as follows:

$$\boxed{\mathbf{w}_{n+1} = \mathbf{w}_n + \mu\, e(n)\text{sgn}\{\mathbf{x}(n)\}} \tag{9.56}$$

which is the *sign-data algorithm*. Note that, unlike the sign-error algorithm, the sign-data algorithm alters the direction of the update vector. As a result, the sign-data algorithm is generally less robust than the sign-error algorithm. In fact, examples have been found in which the coefficients diverge using the sign-data algorithm while converging using the LMS algorithm [28]. Note that the kth coefficient in the sign of the data vector may be written as follows:

$$\text{sgn}\{x(n-k)\} = \frac{x(n-k)}{|x(n-k)|}$$

Therefore, as in the normalized LMS algorithm where the weight vector is normalized by $\|\mathbf{x}(n)\|^2$, the sign-data algorithm individually normalizes each coefficient of the weight vector. Thus, the sign-data algorithm may be written as

$$w_{n+1}(k) = w_n(k) + \frac{\mu}{|x(n-k)|}e(n)x(n-k)$$

which is an LMS algorithm that has a different (time-varying) step size for each coefficient in the weight vector.

Finally, quantizing both the error and the data leads to the *sign-sign* algorithm, which has a coefficient update equation given by

$$\boxed{\mathbf{w}_{n+1} = \mathbf{w}_n + \mu\,\text{sgn}\{e(n)\}\text{sgn}\{\mathbf{x}(n)\}} \tag{9.57}$$

In this algorithm, the coefficients $w_n(k)$ are updated by either adding or subtracting a constant μ. For stability, a leakage term is often introduced into the sign-sign algorithm [3] giving an update equation of the form

$$\mathbf{w}_{n+1} = (1 - \mu\gamma)\mathbf{w}_n + \mu \operatorname{sgn}\{e(n)\}\operatorname{sgn}\{\mathbf{x}(n)\}$$

Generally, the sign-sign algorithm is slower to converge than the LMS adaptive filter and has a larger excess mean-square error [5,11]. Nevertheless, the extreme simplicity of the update algorithm has made it a popular algorithm and has, in fact, been adopted in the international CCITT standard for 32kbps ADPCM [10].[5]

Variable Step-Size Algorithms. In selecting the step size μ in the LMS algorithm, there is a tradeoff between the rate of convergence, the amount of excess mean-square error, and the ability of the filter to track signals as their statistics change. Ideally, when the adaptation begins and $\mathbf{w}_n$ is far from the optimum solution, the step size should be large in order to move the weight vector rapidly toward the desired solution. Then, as the filter begins to converge in the mean to the steady-state solution, the step size should be decreased in order to reduce the excess mean-square error. This suggests the possibility of using a variable step size in the LMS adaptive filter. The difficulty, however, is in specifying a set of rules for changing the step size in such a way that the adaptive filter has a small excess mean-square error while, at the same time, maintaining the ability of the filter to respond quickly to changes in the signal statistics.

Assuming that $x(n)$ and $d(n)$ are real-valued processes, one approach that has been proposed for changing the step size, referred to as the *Variable Step* (VS) algorithm [19], is to use a coefficient update equation of the form

$$w_{n+1}(k) = w_n(k) + \mu_n(k)e(n)x(n - k)$$

where $\mu_n(k)$ are step sizes that are adjusted independently for each coefficient. The rules for adjusting $\mu_n(k)$ are tied to the rate at which the gradient estimate changes sign. With an estimated gradient given by

$$\nabla e^2(n) = -2e(n)x(n - k)$$

these rules are based on the premise that if the sign of $e(n)x(n - k)$ is changing frequently, then the coefficient $w_n(k)$ should be close to its optimum value where the gradient is equal to zero. On the other hand, if the sign is not changing very often, then the coefficient $w_n(k)$ is probably not close to its optimum value. Therefore, $\mu_n(k)$ is decreased by a constant, c_1, if m_1 successive sign changes are observed in $e(n)x(n - k)$, whereas $\mu_n(k)$ is increased by a constant, c_2, if $e(n)x(n - k)$ has the same sign for m_2 successive updates. In addition, hard limits are placed on the step size,

$$\mu_{\min} < \mu_n(k) < \mu_{\max}$$

in order to ensure that the VS algorithm converges in the mean with only a modest increase in computation, the VS algorithm may result in a considerable improvement in the convergence rate [19].

[5]See "32 kbit/s Adaptive differential pulse code modulation (ADPCM)," in the CCITT standard recommendation G.721, Melbourne 1988.

9.2.7 Gradient Adaptive Lattice Filter

Up to this point we have only considered FIR adaptive filters implemented in direct form. However, as discussed in Chapter 6, the lattice filter has some important advantages over a direct form structure. Therefore, in this section we consider the design of adaptive lattice filters. Our development begins with the derivation of the gradient adaptive lattice (GAL) for adaptive linear prediction. Then, in Section 9.2.8, we will consider the design of an adaptive lattice filter for estimating an arbitrary process $d(n)$.

As we saw in Chapter 6, an FIR lattice filter is parameterized in terms of its reflection coefficients, Γ_j. The outputs of the jth stage of the lattice filter are the jth-order forward and backward prediction errors,

$$\begin{aligned} e_j^+(n) &= x(n) + \sum_{k=1}^{j} a_j(k)x(n-k) \\ e_j^-(n) &= x(n-j) + \sum_{k=1}^{j} a_j^*(k)x(n-j+k) \end{aligned} \tag{9.58}$$

where the coefficients $a_j(k)$ are related to the reflection coefficients through the step-up and step-down recursions (see Section 5.2.4). In Section 6.6, we considered several different ways to design a lattice filter for linear prediction. These included the sequential minimization of either the mean-square forward prediction error, the mean-square backward prediction error, or the Burg error. For example, we found that sequentially minimizing the Burg error,

$$\xi_j^B(n) = E\left\{|e_j^+(n)|^2 + |e_j^-(n)|^2\right\}$$

results in a set of reflection coefficients given by

$$\Gamma_j^B = -2\frac{E\left\{e_{j-1}^+(n)\left[e_{j-1}^-(n-1)\right]^*\right\}}{E\left\{|e_{j-1}^+(n)|^2 + |e_{j-1}^-(n-1)|^2\right\}} \tag{9.59}$$

As an alternative to this solution, let us consider a gradient descent approach to minimize the Burg error using an update equation of the form[6]

$$\Gamma_j(n+1) = \Gamma_j(n) - \mu_j \frac{\partial \xi_j^B(n)}{\partial \Gamma_j^*} \tag{9.60}$$

From the lattice filter update equations

$$\begin{aligned} e_j^+(n) &= e_{j-1}^+(n) + \Gamma_j e_{j-1}^-(n-1) \\ e_j^-(n) &= e_{j-1}^-(n-1) + \Gamma_j^* e_{j-1}^+(n) \end{aligned} \tag{9.61}$$

it follows that the derivative in Eq. (9.60) is

$$\frac{\partial \xi_j^B(n)}{\partial \Gamma_j^*} = E\left\{e_j^+(n)\left[e_{j-1}^-(n-1)\right]^* + \left[e_j^-(n)\right]^* e_{j-1}^+(n)\right\}$$

[6]Note that we are allowing the step size to be different for each reflection coefficient.

Therefore, the update equation becomes

$$\boxed{\Gamma_j(n+1) = \Gamma_j(n) - \mu_j E\left\{e_j^+(n)\left[e_{j-1}^-(n-1)\right]^* + \left[e_j^-(n)\right]^* e_{j-1}^+(n)\right\}} \tag{9.62}$$

which is the *steepest descent adaptive lattice filter*. Since $e_j^+(n)$ and $\left[e_j^-(n)\right]^*$ depend on $\Gamma_j(n)$, substituting Eq. (9.61) into Eq. (9.62) it follows that $\Gamma_j(n)$ satisfies the first-order difference equation

$$\begin{aligned}\Gamma_j(n+1) = &\left[1 - \mu_j E\left\{|e_{j-1}^+(n)|^2 + |e_{j-1}^-(n-1)|^2\right\}\right]\Gamma_j(n) \\ &+ 2\mu_j E\left\{e_{j-1}^+(n)\left[e_{j-1}^-(n-1)\right]^*\right\}\end{aligned} \tag{9.63}$$

For stationary processes, $\Gamma_j(n)$ will converge to the Burg solution Γ_j^B in Eq. (9.59) provided the term multiplying $\Gamma_j(n)$ in Eq. (9.63) is less than one in magnitude,

$$\left|1 - \mu_j E\left\{|e_{j-1}^+(n)|^2 + |e_{j-1}^-(n-1)|^2\right\}\right| < 1$$

This, in turn, places the following constraint on the step size μ_j,

$$0 < \mu_j < \frac{2}{E\left\{|e_{j-1}^+(n)|^2 + |e_{j-1}^-(n-1)|^2\right\}} \tag{9.64}$$

Due to the expectation in Eq. (9.62), the steepest descent adaptive lattice filter is only of limited use since it requires that the second-order statistics of the forward and backward prediction errors be known. However, if we adopt the LMS approach of replacing these expected values with instantaneous values, then the update equation (9.62) becomes

$$\boxed{\Gamma_j(n+1) = \Gamma_j(n) - \mu_j\left\{e_j^+(n)\left[e_{j-1}^-(n-1)\right]^* + \left[e_j^-(n)\right]^* e_{j-1}^+(n)\right\}} \tag{9.65}$$

which is the *Gradient Adaptive Lattice* (GAL) filter [17,18]. With an analysis similar to that presented in Section 9.2.3 for the LMS adaptive filter, it follows that if $x(n)$ is wide-sense stationary, then the reflection coefficients $\Gamma_j(n)$ will converge in the mean to the values given in Eq. (9.59), provided that each step size μ_j satisfies the condition given in Eq. (9.64). Due to the fact that the lattice filter orthogonalizes the input process, thereby decoupling the estimation of the reflection coefficients, the convergence of the gradient adaptive lattice filter is typically faster than the direct form adaptive filter [21,27]. This orthogonalization also makes the convergence time essentially independent of the eigenvalue spread [17]. A drawback of the gradient adaptive lattice, however, is that it requires approximately twice as many operations per update than required for the direct form filter. The gradient adaptive lattice is summarized in Table 9.2.

Due to the difficulties involved in determining the step sizes necessary to satisfy the convergence constraint given in Eq. (9.64), the GAL algorithm is typically *normalized* using a time-varying step-size of the form

$$\mu_j(n) = \frac{\beta}{D_j(n)}$$

Table 9.2 *The Gradient Adaptive Lattice Algorithm*

Parameters:	p = Filter order
	μ_j = step sizes, $j = 1, 2, \ldots, p$
Initialization:	$e_0^+(n) = e_0^-(n) = 0$
	$\Gamma_j(0) = 0$ for $j = 1, 2, \ldots, p$
Computation:	For $n = 0, 1, 2, \ldots$ and $j = 1, \ldots, p$ compute
	(a) $e_j^+(n) = e_{j-1}^+(n) + \Gamma_j(n) e_{j-1}^-(n-1)$
	(b) $e_j^-(n) = e_{j-1}^-(n-1) + \Gamma_j^*(n) e_{j-1}^+(n)$
	(c) $\Gamma_j(n+1) = \Gamma_j(n) - \mu_j \left\{ e_j^+(n) \left[e_j^-(n-1) \right]^* + \left[e_j^-(n) \right]^* e_{j-1}^+(n) \right\}$

where β is a constant with $0 < \beta < 2$, and $D_j(n)$ is an estimate of the denominator in Eq. (9.64). An estimate that is frequently used is

$$D_j(n) = (1 - \lambda) \sum_{k=0}^{n} \lambda^{n-k} \left[|e_{j-1}^+(k)|^2 + |e_{j-1}^-(k-1)|^2 \right]$$

where λ is an exponential weighting factor with $0 < \lambda < 1$. An advantage of this estimate is that $D_j(n)$ may be evaluated efficiently with the recursion

$$D_j(n+1) = \lambda D_j(n) + (1 - \lambda) \left[|e_{j-1}^+(n+1)|^2 + |e_{j-1}^-(n)|^2 \right] \tag{9.66}$$

9.2.8 Joint Process Estimator

The gradient adaptive lattice developed in the previous section may be used for adaptive linear prediction or, equivalently, adaptive all-pole signal modeling. However, since it minimizes the mean-square forward and backward prediction errors, the gradient adaptive lattice is not directly applicable to other adaptive filtering applications such as system identification, channel equalization, or echo cancellation. On the other hand, recall that the *joint process estimator* developed in Section 7.2.4 may be used for the estimation of an arbitrary process $d(n)$. This joint process estimator may be made adaptive as shown in Fig. 9.18 and consists of two stages. The first stage is an adaptive lattice predictor as described in the previous section. The second stage produces an estimate of $d(n)$ by taking a linear combination of the backward prediction errors $e_j^-(n)$. Thus, we have an adaptive *lattice preprocessor* working in conjunction with an adaptive tapped delay line.

To analyze this adaptive joint process estimator, let us begin by assuming that the reflection coefficients $\Gamma_k(n)$ and the tapped delay line coefficients $b_n(k)$ are constant. With

$$e(n) = d(n) - \hat{d}(n) = d(n) - \mathbf{b}^T \mathbf{e}^-(n)$$

where $\mathbf{b} = \left[b(0), b(1), \ldots, b(p-1) \right]^T$ and $\mathbf{e}^-(n) = \left[e_0^-(n), e_1^-(n), \ldots, e_{p-1}^-(n) \right]^T$, we may find the coefficients $b(k)$ that minimize the mean-square error $\xi(n) = E\left\{ |e(n)|^2 \right\}$

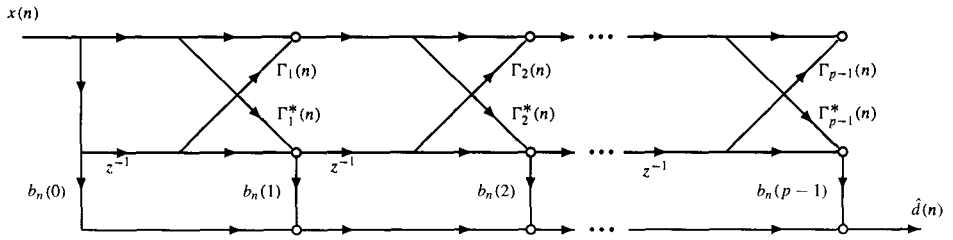


Figure 9.18 *An adaptive formulation of the lattice joint process estimator for estimating a process* $d(n)$ *from* $x(n)$.

by differentiating $\xi(n)$ with respect to $b^*(k)$ and setting the derivatives equal to zero. The result is a set of normal equations of the form

$$\mathbf{R}_e\mathbf{b} = \mathbf{r}_{de}$$

where

$$\mathbf{R}_e = E\{\mathbf{e}^-(n)[\mathbf{e}^-(n)]^H\}$$

is the autocorrelation matrix of the backward prediction error, and

$$\mathbf{r}_{de} = E\{d(n)[\mathbf{e}^-(n)]^H\}$$

is the cross-correlation between the desired process and the backward prediction error. These equations may be simplified considerably if we use the fact that the backward prediction errors are orthogonal (see Section 6.6),

$$E\{\mathbf{e}^-(n)[\mathbf{e}^-(n)]^H\} = \mathbf{D}_p = \text{diag}\{\epsilon_0, \epsilon_1, \ldots, \epsilon_p\}$$

This diagonal structure for $\mathbf{R}_e$ decouples the normal equations and allows us to solve for each coefficient $b(k)$ independently as follows:

$$b(k) = \frac{1}{\epsilon_k}E\{d(n)[e_k^-(n)]^*\} \tag{9.67}$$

Since this solution assumes that the processes are stationary and requires knowledge of the cross-correlation between $d(n)$ and $e_k^-(n)$, we may consider using a gradient descent approach to adaptively adjust the coefficients $b(k)$ as follows:

$$\mathbf{b}_{n+1} = \mathbf{b}_n - \mu\nabla\xi(n)$$

However, since the gradient involves an expectation,

$$\nabla\xi(n) = -E\{e(n)[\mathbf{e}^-(n)]^*\}$$

we may use the LMS approach of replacing the expected value with an instantaneous value. This results in the update equation

$$\boxed{\mathbf{b}_{n+1} = \mathbf{b}_n + \mu e(n)[\mathbf{e}^-(n)]^*} \tag{9.68}$$

which, in terms of the individual filter coefficients, is

$$b_{n+1}(k) = b_n(k) + \mu e(n)[e_k^-(n)]^*$$

In the overall system, there are two sets of parameters that are being updated in parallel: the reflection coefficients in the lattice predictor, $\Gamma_j(n)$, which are attempting to orthogonalize the input process $x(n)$, and the coefficients $b_n(k)$ that are used to form the estimate of $d(n)$. Due to the orthogonalization of $x(n)$, this filter tends to be less sensitive to the eigenvalue spread than the LMS adaptive filter. However, the backward prediction errors will not be orthogonal until the reflection coefficients have converged. Therefore, the backward prediction errors will be correlated and the behavior of the adaptive filter will depend on the eigenvalue spread until each $\Gamma_j(n)$ has converged. One interesting and important feature of this adaptive filter is that the orthogonalization of $x(n)$ does not depend on the desired signal $d(n)$. Therefore, if $x(n)$ is stationary, once the lattice predictor has converged, the backwards prediction error will remain orthogonal even if $d(n)$ is nonstationary.

9.2.9 Application: Channel Equalization

In this section, we will look briefly at the problem of adaptive channel equalization [20,26,27]. Adaptive equalizers are necessary for reliable communication of digital data across nonideal channels. The basic operation of a digital communication system may be described as follows. Let $d(n)$ be a digital sequence that is to be transmitted across a channel, with $d(n)$ having values of plus or minus one. This sequence is input to a pulse generator, which produces a pulse of amplitude A at time n if $d(n) = 1$ and a pulse of amplitude $-A$ if $d(n) = -1$. This sequences of pulses is then modulated and transmitted across a channel to a remote receiver. The receiver demodulates and samples the received waveform, which produces a discrete-time sequence $x(n)$. Although ideally $x(n)$ will be equal to $d(n)$, in practice this will rarely be the case. There are two reasons for this. First, since the channel is never ideal, it will introduce some distortion. One common type of distortion is *channel dispersion* that is the result of the nonlinear phase characteristics of the channel. This dispersion causes a distortion of the pulse shape, thereby causing neighboring pulses to interfere with each other, resulting in an effect known as *intersymbol interference* [24,26]. The second reason is that the received waveform will invariably contain noise. This noise may be introduced by the channel or it may be the result of nonideal elements in the transmitter and receiver. Assuming a linear dispersive channel, a model for the received sequence $x(n)$ is

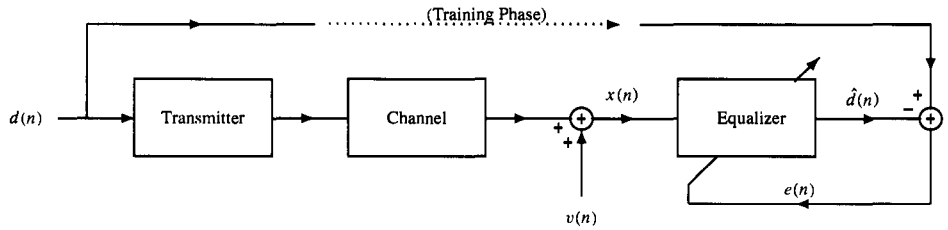
$$x(n) = \sum_{k=-\infty}^{n} d(k)h(n-k) + v(n)$$

where $h(n)$ is the unit sample response of the channel and $v(n)$ is additive noise. Given the received sequence $x(n)$, the receiver then makes a decision as to whether a plus one or a minus one was transmitted at time n. This decision is typically made with a simple threshold device, i.e.,

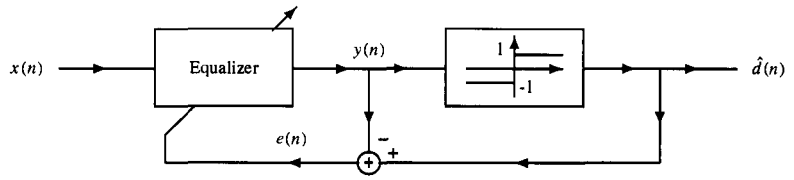
$$\hat{d}(n) = \begin{cases} 1 & ; \quad x(n) \geq 0 \\ -1 & ; \quad x(n) < 0 \end{cases}$$

However, in order to reduce the chance of making an error, the receiver will often employ an equalizer in order to reduce the effects of channel distortion. Since the precise characteristics of the channel are unknown, and possibly time-varying, the equalizer is typically an adaptive filter. A simplified block diagram of the digital communication system is shown in Fig. 9.19.

As discussed in Section 9.1, one of the challenges in the design of an adaptive filter is generating the desired signal $d(n)$, which is required in order to compute the error $e(n)$.



(a) Channel equalization using a training sequence.



(b) The operation of the equalizer in decision-directed mode.

Figure 9.19 *A basic digital communication system with channel equalization.*

Without the error, a gradient descent algorithm will not work. For the channel equalizer, there are two methods that may be used to generate the error signal.

1. **Training method**. This method takes place during an initial training phase, which occurs when the transmitter and receiver first establish a connection. During this phase, the transmitter sends a sequence of pseudorandom digits that is known to the receiver. With knowledge of $d(n)$ the error sequence is easily determined and the tap weights of the equalizer may be *initialized*.
2. **Decision-directed method**. Once the training period has ended and data is being exchanged between the transmitter and receiver, the receiver has no a priori knowledge of what is being sent. However, there is a clever scheme that may be used to extract the error sequence from the output of the threshold device as illustrated in Fig. 9.19*b*. Specifically, note that if no errors are made at the output of the threshold device and $\hat{d}(n) = d(n)$, then the error sequence may be formed by taking the difference between the equalizer output, $y(n)$, and the output of the threshold device,

$$e(n) = y(n) - \hat{d}(n)$$

This approach is said to be *decision directed* since it is based on the decisions made by the receiver. Although based on the threshold device making a correct decision, this approach will work even in the presence of errors, provided that they are infrequent enough. Of course, once the error rate exceeds a certain level, the inaccuracies in the error signal will cause the equalizer to diverge away from the correct solution thereby causing an increase in the error rate and eventually a loss of reliable communication. Before this happens, however, the receiver may request that the training sequence be retransmitted in order to re-initialize the equalizer.

To get a feeling for how well such an equalizer may work, we will look at the use of the LMS algorithm for the adaptive equalization of a linear dispersive channel that has a unit

sample response given by [20,27]

$$h(n) = \begin{cases} \frac{1}{2} + \frac{1}{2}\cos\left[\frac{2\pi(n-2)}{W}\right] & ; \quad n = 1, 2, 3 \\ 0 & ; \quad \text{otherwise} \end{cases} \tag{9.69}$$

In addition to controlling the amount of distortion introduced by the channel, the parameter W affects the eigenvalues of the autocorrelation matrix $\mathbf{R}_x$ of the received signal $x(n)$. We will assume that the data is real-valued and that the input to the channel, $d(n)$, is a sequence of uncorrelated random variables with $d(n) = \pm 1$ with equal probability. Finally, we will assume that the additive noise $v(n)$ is zero mean white Gaussian noise with a variance $\sigma_v^2 = 0.005$ and that $v(n)$ is uncorrelated with $d(n)$.

We begin by finding the autocorrelation matrix for $x(n)$ and the corresponding eigenvalues. Recall that our model for $x(n)$ is

$$x(n) = d(n) * h(n) + v(n)$$

Since $d(n)$ is a sequence of uncorrelated random variables then $r_d(k) = \delta(k)$. Therefore, with $v(n)$ white noise with a variance σ_v^2, and with $v(n)$ being uncorrelated with $d(n)$, the autocorrelation sequence of $x(n)$ is

$$r_x(k) = h(k) * h(-k) + \sigma_v^2 \delta(k)$$

Since $h(n)$ is nonzero only for $n = 1, 2, 3$ we have

$$\begin{aligned} r_x(0) &= h^2(1) + h^2(2) + h^2(3) + \sigma_v^2 \\ r_x(1) &= h(1)h(2) + h(2)h(3) \\ r_x(2) &= h(1)h(3) \end{aligned}$$

and $r_x(k) = 0$ for all $k > 2$. Therefore, the autocorrelation matrix of $x(n)$ is a Toeplitz matrix of the form

$$\mathbf{R}_x = \text{Toep}\{r_x(0),\ r_x(1),\ r_x(2),\ 0, \ldots,\ 0\}$$

which is a *quintdiagonal* matrix, i.e., the only nonzero elements are along the main diagonal and the two diagonals above and below the main diagonal. The eigenvalues of $\mathbf{R}_x$, which affect the convergence of the adaptive filter, depend on the value of W as well as on the size, p, of the autocorrelation matrix $\mathbf{R}_x$, which is equal to the number of coefficients in the adaptive filter. Shown in Table 9.3 are the maximum and minimum eigenvalues of $\mathbf{R}_x$ along with the condition number χ for several different values of W and $p = 15$. From our discussions in Chapter 2 (see Property 7 on p. 48) we may upper bound the largest eigenvalue of $\mathbf{R}_x$ as the largest row sum which, for $p \geq 5$ is

$$\lambda_{\max} \leq r_x(0) + 2r_x(1) + 2r_x(2) \tag{9.70}$$

For example, for the given values of W listed in Table 9.3 we have

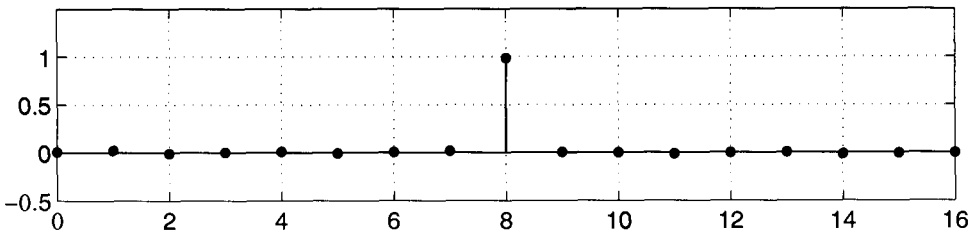
$$\lambda_{\max} \leq 1.8998\,(W = 2.8) \quad ; \quad \lambda_{\max} \leq 2.2550\,(W = 3.0) \quad ; \quad \lambda_{\max} \leq 2.6207\,(W = 3.2)$$

Note that these bounds are not tied to a specific value of p.

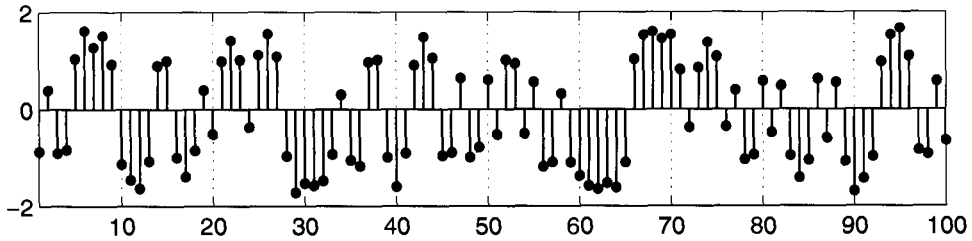
Let us now look at the performance of the adaptive equalizer using the normalized LMS algorithm to adjust the filter coefficients. With $\beta = 0.3$, $W = 3.0$, and $p = 15$, the equalizer is first initialized using a training sequence of length 500. Shown in Fig. 9.20*a*

Table 9.3 ***The Maximum and Minimum Eigenvalues for Three Different Channel Characteristics***

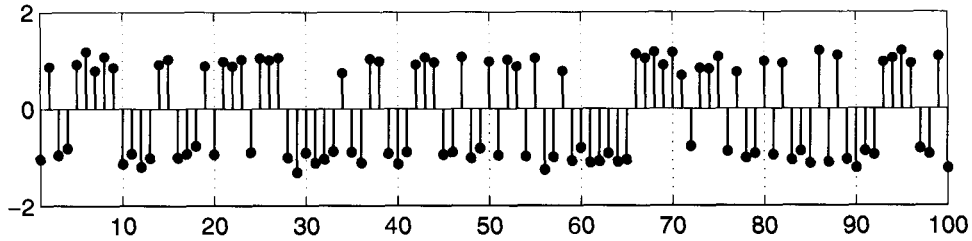
W	2.8	3.0	3.2
$r_x(0)$	1.0759	1.1300	1.1955
$r_x(1)$	0.3765	0.5000	0.6173
$r_x(2)$	0.0354	0.0625	0.0953
$\lambda_{\max}$	1.8803	2.2269	2.5835
$\lambda_{\min}$	0.4031	0.2652	0.1614
χ	4.6643	8.3966	16.006



(a)



(b)



(c)

Figure 9.20 *Performance of the channel equalizer. (a) The convolution of the unit sample response of the channel with the equalizer at the end of the training sequence. (b) The received sequence after the training period has been completed. (c) The output of the equalizer.*

is the convolution of the unit sample response of the channel, $h(n)$, with the unit sample response of the equalizer $w(n)$ at the end of the training sequence. Since this convolution is approximately a unit sample at $n = 8$, the equalizer is a good approximation to the

inverse of $h(n)$. However, after the training sequence, the equalizer no longer has knowledge of $d(n)$, and must rely on decision directed adaptation to generate the error sequence to drive the coefficient update equation. Shown in Fig. 9.20*b* is the received sequence, $x(n)$, for the first 100 values after the training period and in Fig. 9.20*c* is the output of adaptive equalizer. As we see, the equalizer is effective in producing a sequence $y(n)$ with $|y(n)| \approx 1$. Due to noise, however, and the fact that the equalizer is only an approximation to the inverse of $h(n)$, the equalized process is not exactly equal to $d(n)$. However, the equalizer output is accurate enough so that the threshold device produces an output $\hat{d}(n) = d(n)$.

9.3 ADAPTIVE RECURSIVE FILTERS

In the previous section, a variety of different methods were presented for designing an FIR (non-recursive) adaptive filter for producing the minimum mean-square estimation of a process $d(n)$ from measurements of a related process, $x(n)$. In this section, we turn our attention briefly to the problem of designing IIR (recursive) adaptive filters of the form

$$y(n) = \sum_{k=1}^{p} a_n(k)y(n-k) + \sum_{k=0}^{q} b_n(k)x(n-k) \tag{9.71}$$

where $a_n(k)$ and $b_n(k)$ are the coefficients of the adaptive filter at time n. Just as with linear shift-invariant (non-adaptive) filters, recursive adaptive filters potentially have an advantage over non-recursive filters in providing better performance for a given filter order (number of filter coefficients). However, with an adaptive recursive filter there are potential instability problems that may affect the convergence time as well as the general numerical sensitivity of the filter. High resonance poles, for example, may have time constants that are longer than the modes of the adaptive algorithm. In spite of these difficulties, there are many applications in which a recursive filter may be required. For example, suppose that a channel used to transmit a signal $d(n)$ introduces an echo so that the received signal is

$$x(n) = d(n) + \alpha d(n-N)$$

where $|\alpha| < 1$ and N is the delay associated with the echo. If both α and N are known, then the ideal *echo canceller* for recovering $d(n)$ from $x(n)$ is an IIR filter that has a system function given by

$$H(z) = \frac{1}{1-\alpha z^{-N}}$$

However, since α and N are generally unknown and possibly time-varying, then it is more appropriate for the echo canceller to be an adaptive recursive filter. Although we could consider a nonrecursive adaptive filter, the order of the filter required for a sufficiently accurate estimate of $d(n)$ may be too large. For example, suppose that the inverse filter, $H(z)$, is expanded in a geometric series as follows:

$$H(z) = \frac{1}{1-\alpha z^{-N}} = \sum_{k=0}^{\infty} \alpha^k z^{-Nk}$$

If p is large enough so that $|\alpha|^p \ll 1$, then we could form a finite-order approximation to

$H(z)$ as follows:

$$\hat{H}(z) = \sum_{k=0}^{p} \alpha^k z^{-Nk}$$

and consider an adaptive nonrecursive echo canceller of the form

$$\hat{d}(n) = \sum_{k=0}^{Np} b_n(k) x(n-k)$$

However, if $\alpha \approx 1$, which forces p to be large, or if $N \gg 1$, then the order of the adaptive filter, Np, required to produce a sufficiently accurate approximation to the inverse filter may be too large for this to be a viable solution.

We begin our discussion of adaptive recursive filters with a derivation of the conditions that are necessary in order for the coefficients of the recursive filter to minimize the mean-square error, $\xi(n) = E\{|e(n)|^2\}$, where $e(n) = d(n) - y(n)$ is the difference between the desired process $d(n)$ and the output of the adaptive filter, $y(n)$. We will initially assume that the filter is shift-invariant (non-adaptive) with

$$y(n) = \sum_{l=1}^{p} a(l) y(n-l) + \sum_{l=0}^{q} b(l) x(n-l) \tag{9.72}$$

If we let Θ be the vector of filter coefficients,

$$\Theta = \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix} = [a(1),\ a(2),\ \ldots,\ a(p),\ b(0),\ b(1),\ \ldots,\ b(q)]^T$$

and let $\mathbf{z}(n)$ denote the *aggregate data vector*

$$\begin{aligned} \mathbf{z}(n) &= \begin{bmatrix} \mathbf{y}(n-1) \\ \mathbf{x}(n) \end{bmatrix} \\ &= [y(n-1),\ y(n-2),\ \ldots,\ y(n-p),\ x(n),\ \ldots,\ x(n-q)]^T \end{aligned} \tag{9.73}$$

then the output of the filter at time n may be expressed in terms of Θ and $\mathbf{z}(n)$ as follows:

$$y(n) = \mathbf{a}^T \mathbf{y}(n-1) + \mathbf{b}^T \mathbf{x}(n) = \Theta^T \mathbf{z}(n) \tag{9.74}$$

In order for Θ to minimize the mean-square error

$$\xi(n) = E\{|e(n)|^2\} = E\{|d(n) - y(n)|^2\}$$

it is necessary that the derivatives of $\xi(n)$ with respect to each of the coefficients, $a^*(k)$ and $b^*(k)$, be equal to zero, i.e., the gradient vector must be zero.[7] Since

$$\nabla\xi(n) = \nabla E\{|e(n)|^2\} = E\{\nabla|e(n)|^2\} = E\{e(n)\nabla e^*(n)\} \tag{9.75}$$

then the gradient will be zero when

$$E\{e(n)\nabla e^*(n)\} = 0 \tag{9.76}$$

[7]For an FIR filter, this is a necessary and sufficient condition since the mean-square error is a quadratic function of the coefficients $b(k)$. However, with the feedback coefficients $a(k)$, the mean-square error is no longer quadratic and $\xi(n)$ may have multiple local minima and maxima [31]. Therefore, this condition is only necessary and, in general, not sufficient.

Since $e(n) = d(n) - y(n)$ then

$$\nabla e^*(n) = -\nabla y^*(n)$$

and Eq. (9.76) becomes

$$E\left\{e(n)\nabla y^*(n)\right\} = 0 \tag{9.77}$$

The next step is to evaluate $\nabla y^*(n)$ by differentiating with respect to $a^*(k)$ and $b^*(k)$. In forming these derivatives, it must be kept in mind that $y(n)$ depends on $y(n-l)$ for $l = 1, 2, \ldots, p$, and each of these, in turn, depends on $a(k)$ and $b(k)$. Therefore, the partial derivatives of $y^*(n)$ with respect to $a^*(k)$ and $b^*(k)$ are

$$\begin{aligned}
\frac{\partial y^*(n)}{\partial a^*(k)} &= y^*(n-k) + \sum_{l=1}^{p} a^*(l)\frac{\partial y^*(n-l)}{\partial a^*(k)} \quad ; \quad k = 1, 2, \ldots, p \\
\frac{\partial y^*(n)}{\partial b^*(k)} &= x^*(n-k) + \sum_{l=1}^{p} a^*(l)\frac{\partial y^*(n-l)}{\partial b^*(k)} \quad ; \quad k = 0, 1, \ldots, q
\end{aligned} \tag{9.78}$$

Writing these equations in vector form leads to the following expression for the gradient vector:

$$\boxed{\nabla y^*(n) = \mathbf{z}^*(n) + \sum_{l=1}^{p} a^*(l)\nabla y^*(n-l)} \tag{9.79}$$

where $\mathbf{z}(n)$ is the aggregate data vector defined in Eq. (9.73). Finally, combining Eqs. (9.77) and (9.79) we obtain the following conditions that are necessary for $a(k)$ and $b(k)$ to minimize the mean-square error:

$$\begin{aligned}
E\left\{e(n)\left[y(n-k) + \sum_{l=1}^{p} a(l)\frac{\partial y(n-l)}{\partial a(k)}\right]^*\right\} &= 0 \quad ; \quad k = 1, 2, \ldots, p \\
E\left\{e(n)\left[x(n-k) + \sum_{l=1}^{p} a(l)\frac{\partial y(n-l)}{\partial b(k)}\right]^*\right\} &= 0 \quad ; \quad k = 0, 1, \ldots, q
\end{aligned} \tag{9.80}$$

Since these equations are nonlinear, they are difficult to solve for the optimum filter coefficients. This nonlinearity also implies that the solution to these equations may not be unique. Therefore, a solution may correspond to a local rather than a global minimum. As a result, these equations are generally not solved directly.

An alternative to solving Eq. (9.81) directly is to use a steepest descent approach to search for the solution iteratively. With $\boldsymbol{\Theta}_n$ the estimate of the solution at time n,

$$\boldsymbol{\Theta}_n = \begin{bmatrix} \mathbf{a}_n \\ \mathbf{b}_n \end{bmatrix} = \left[a_n(1),\ a_n(2),\ \ldots,\ a_n(p),\ b_n(0),\ \ldots,\ b_n(q)\right]^T$$

a steepest descent update for $\boldsymbol{\Theta}_n$ is given by

$$\boldsymbol{\Theta}_{n+1} = \boldsymbol{\Theta}_n - \mu\ \nabla\xi(n) \tag{9.81}$$

where μ is the step size and $\nabla\xi(n)$ is the gradient vector. The difficulty with this update equation is that the gradient vector involves an expectation

$$\nabla\xi(n) = -E\{e(n)\nabla y^*(n)\}$$

Therefore, unless the required ensemble averages are known, the steepest descent algorithm cannot be implemented. However, if we adopt the approach used in the LMS adaptive filter of replacing expected values with instantaneous values,

$$\hat{\nabla}\xi(n) = -e(n)\nabla y^*(n)$$

then the coefficient update equation (9.81) becomes

$$\boxed{\Theta_{n+1} = \Theta_n + \mu\, e(n)\nabla y^*(n)} \tag{9.82}$$

Expressed in terms of the filter coefficients $a_n(k)$ and $b_n(k)$, Eq. (9.82) becomes

$$\begin{aligned} a_{n+1}(k) &= a_n(k) + \mu\, e(n)\frac{\partial y^*(n)}{\partial a_n^*(k)} \\ b_{n+1}(k) &= b_n(k) + \mu\, e(n)\frac{\partial y^*(n)}{\partial b_n^*(k)} \end{aligned} \tag{9.83}$$

where the partial derivatives are evaluated using Eq. (9.79) with the filter coefficients $a(k)$ and $b(k)$ replaced by the time-varying coefficients $a_n(k)$ and $b_n(k)$,

$$\begin{aligned} \frac{\partial y^*(n)}{\partial a_n^*(k)} &= y^*(n-k) + \sum_{l=1}^{p} a_n^*(l)\frac{\partial y^*(n-l)}{\partial a_n^*(k)} \quad ; \quad k = 1, 2, \ldots, p \\ \frac{\partial y^*(n)}{\partial b_n^*(k)} &= x^*(n-k) + \sum_{l=1}^{p} a_n^*(l)\frac{\partial y^*(n-l)}{\partial b_n^*(k)} \quad ; \quad k = 0, 1, \ldots, q \end{aligned} \tag{9.84}$$

Equations (9.84) and (9.83) along with the filter output equation

$$y(n) = \mathbf{a}_n^T\mathbf{y}(n-1) + \mathbf{b}_n^T\mathbf{x}(n) = \Theta_n^T\mathbf{z}(n)$$

form what is known as the *IIR LMS* algorithm [35]. Based on numerous simulations, it has been reported that, for sufficiently small step sizes, the IIR LMS algorithm generally converges to the filter coefficients that minimize the mean-square error [10,32]. However, compared to their FIR counterparts, convergence is generally much slower. In addition, due to the possibility of local minima in the mean-square error surface, convergence may not be to the global minimum.

We will now look at a couple of ways to simplify the IIR LMS adaptive filter. First, note that the derivatives of $y^*(n-l)$ in Eq. (9.84) are taken with respect to the *current* values of $a_n^*(k)$ and $b_n^*(k)$. Therefore, these expressions for the partial derivatives are not recursive and cannot be expressed in terms of a filtering operation. However, if the step size is small enough so that the coefficients are adapting slowly, then

$$\frac{\partial y^*(n-l)}{\partial a_n^*(k)} \approx \frac{\partial y^*(n-l)}{\partial a_{n-l}^*(k)} \quad ; \quad \frac{\partial y^*(n-l)}{\partial b_n^*(k)} \approx \frac{\partial y^*(n-l)}{\partial b_{n-l}^*(k)} \tag{9.85}$$

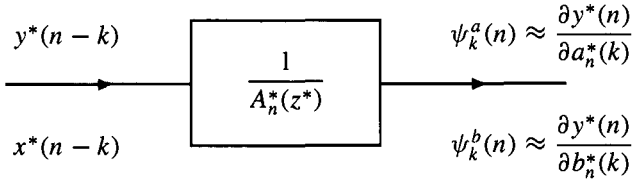


Figure 9.21 *The recursive evaluation of the gradient vector, formed by filtering $x^*(n-k)$ and $y^*(n-k)$ with the recursive filter $1/A_n^*(z^*)$.*

and the derivatives in Eq. (9.84) may be approximated as follows:

$$\begin{aligned}\frac{\partial y^*(n)}{\partial a_n^*(k)} &\approx y^*(n-k) + \sum_{l=1}^{p} a_n^*(l)\frac{\partial y^*(n-l)}{\partial a_{n-l}^*(k)} \\ \frac{\partial y^*(n)}{\partial b_n^*(k)} &\approx x^*(n-k) + \sum_{l=1}^{p} a_n^*(l)\frac{\partial y^*(n-l)}{\partial b_{n-l}^*(k)}\end{aligned} \tag{9.86}$$

Note that these expressions for the partial derivatives are now recursive since the partial derivatives in the sum correspond to delayed versions of the derivatives on the left. Therefore, let $\psi_k^a(n)$ and $\psi_k^b(n)$ be the approximations to the partial derivatives that are generated by these recursions, i.e.,

$$\begin{aligned}\psi_k^a(n) &= y^*(n-k) + \sum_{l=1}^{p} a_n^*(l)\psi_k^a(n-l) \\ \psi_k^b(n) &= x^*(n-k) + \sum_{l=1}^{p} a_n^*(l)\psi_k^b(n-l)\end{aligned} \tag{9.87}$$

As illustrated in Fig. 9.21, these approximations are generated by filtering $x^*(n-k)$ and $y^*(n-k)$ with the shift-varying recursive filter $1/A_n^*(z^*)$. A block diagram of this simplified IIR LMS adaptive filter is shown in Fig. 9.22. Note that there are $p+q+1$ recursive filters operating in parallel that produce the approximations $\psi_k^a(n)$ and $\psi_k^b(n)$ to the gradient vector. The outputs of these filters are then used to update the coefficients of the adaptive filter. The simplified IIR LMS algorithm is summarized in Table 9.4.

In spite of the simplification that results from the approximation made in Eq. (9.85), the implementation of $p+q+1$ shift-varying filters in parallel represents a significant computational load and requires a significant amount of storage. However, if we again assume that the step size μ is small, then the IIR LMS adaptive filter may be simplified further [22]. Specifically, note that the gradient estimate $\psi_k^a(n)$ is formed by delaying $y^*(n)$ and filtering this delayed process with the shift-varying filter $1/A_n^*(z^*)$ as illustrated in Fig. 9.23*a*. The estimate $\psi_k^b(n)$ is found in a similar fashion by filtering delayed versions of $x^*(n)$. If μ is small enough so that the coefficients $a_n(k)$ do not vary significantly over intervals of length p, then the filter $1/A_n^*(z^*)$ may be assumed to be shift-invariant and the cascade of the delay and the shift-varying filter may be interchanged as shown in Fig. 9.23*b*. As a result, $\psi_k^a(n)$ and $\psi_k^b(n)$ may be estimated by simply generating the filtered signals $y^f(n) = \psi_0^a(n)$ and $x^f(n) = \psi_0^b(n)$, and then delaying these signals. This method, referred to as the *filtered signal approach*, requires only two recursive filters to estimate the gradient vector as illustrated in Fig. 9.23*c*. Compared to the $p+q+1$ filters necessary for the simplified IIR LMS algorithm, this method is much more efficient and, for a sufficiently

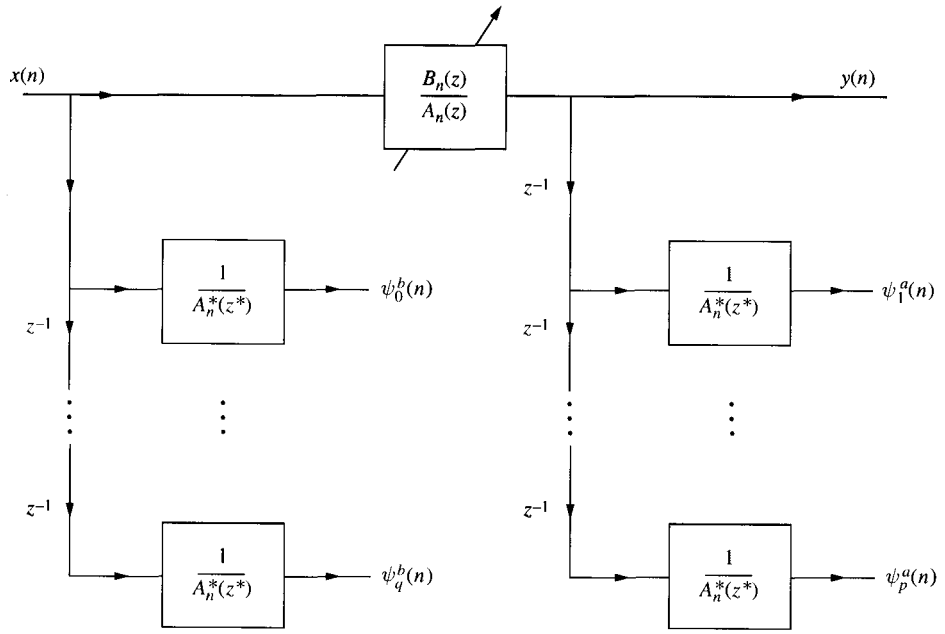


Figure 9.22 *A simplified IIR LMS adaptive filter that assumes that the step size is small enough so that an estimate of the gradient vector may be computed recursively. These estimates, $\psi_k^a(n)$ and $\psi_k^b(n)$, are used by the adaptive filter to update the filter coefficients. (Note that the inputs to the filters $1/A_n^*(z^*)$ must be conjugated prior to filtering.)*

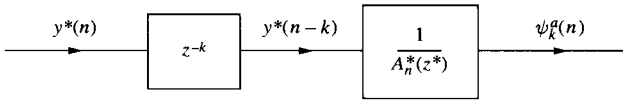
Table 9.4 *A Simplified IIR LMS Adaptive Filter*

Parameters:	p, q = Filter order
	μ = step size
Initialization:	$\mathbf{a}_0 = \mathbf{b}_0 = \mathbf{0}$
Computation:	For $n = 0, 1, \ldots$ compute
	(a) $y(n) = \mathbf{a}_n^T \mathbf{y}(n-1) + \mathbf{b}_n^T \mathbf{x}(n)$
	(b) $e(n) = d(n) - y(n)$
	(c) For $k = 1, 2, \ldots, p$
	$\psi_k^a(n) = y^*(n-k) + \sum_{l=1}^{p} a_n^*(l)\psi_k^a(n-l)$
	$a_{n+1}(k) = a_n(k) + \mu e(n)\psi_k^a(n)$
	(d) For $k = 0, 1, \ldots, q$
	$\psi_k^b(n) = x^*(n-k) + \sum_{l=1}^{p} a_n^*(l)\psi_k^b(n-l)$
	$b_{n+1}(k) = b_n(k) + \mu e(n)\psi_k^b(n)$

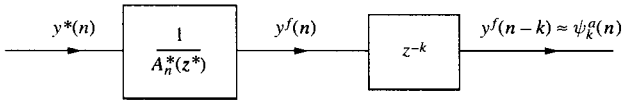
small step size, has a performance that is similar to the IIR LMS algorithm [22]. The filtered signal adaptive recursive filter is summarized in Table 9.5. For a more complete treatment of adaptive recursive filters, see Reference [29].

Table 9.5 ***The Equations for the Filtered Signal Approach to Adaptive Recursive Filtering***

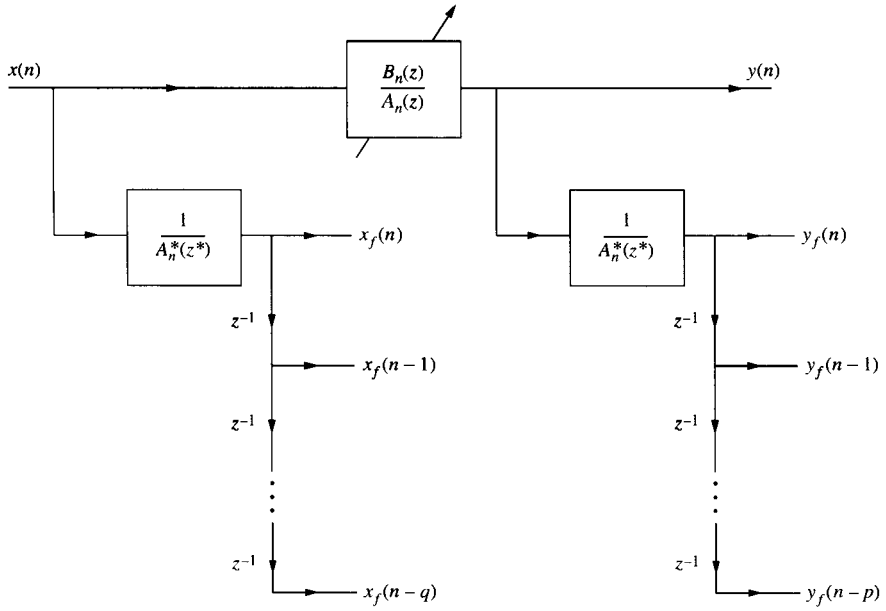
Parameters:	p, q = Filter order μ = step size	
Initialization:	$\mathbf{a}_0 = \mathbf{b}_0 = \mathbf{0}$	
Computation:	For $n = 0, 1, \ldots$ compute	
	(a)	$y(n) = \mathbf{a}_n^T \mathbf{y}(n-1) + \mathbf{b}_n^T \mathbf{x}(n)$
	(b)	$e(n) = d(n) - y(n)$
	(c)	$y^f(n) = y(n) + \sum_{k=1}^{p} a_n^*(k) y^f(n-k)$ $x^f(n) = x(n) + \sum_{k=1}^{p} a_n^*(k) x^f(n-k)$
	(d)	$a_{n+1}(k) = a_n(k) + \mu e(n) y^f(n-k)$ $b_{n+1}(k) = b_n(k) + \mu e(n) x^f(n-k)$



(*a*) The gradient approximation that is formed by filtering $y^*(n-k)$ with the shift-varying recursive filter $1/A_n^*(z^*)$.



(*b*) Assuming that the filter coefficients are slowly varying, the delay and the recursive filter may be interchanged.



(*c*) The resulting simplification to the IIR LMS adaptive filter. (Note that $x(n)$ and $y(n)$ must be conjugated prior to filtering.)

Figure 9.23 *The filtered signal approach for IIR adaptive filtering.*

9.4 RECURSIVE LEAST SQUARES

In each of the adaptive filtering methods discussed so far, we have considered gradient descent algorithms for the minimization of the *mean-square error*

$$\xi(n) = E\{|e(n)|^2\}$$

The difficulty with these methods is that they all require knowledge of the autocorrelation of the input process, $E\{x(n)x^*(n-k)\}$, and the cross-correlation between the input and the desired output, $E\{d(n)x^*(n-k)\}$. When this statistical information is unknown, we have been forced to estimate these statistics from the data. In the LMS adaptive filter, for example, these ensemble averages are estimated using instantaneous values,

$$\hat{E}\{e(n)x^*(n-k)\} = e(n)x^*(n-k) \tag{9.88}$$

Although this approach may be adequate in some applications, in others this gradient estimate may not provide a sufficiently rapid rate of convergence or a sufficiently small excess mean-square error. An alternative, therefore, is to consider error measures that do not include expectations and that may be computed directly from the data. For example, a least squares error

$$\mathcal{E}(n) = \sum_{i=0}^{n} |e(i)|^2$$

requires no statistical information about $x(n)$ or $d(n)$, and may be evaluated directly from $x(n)$ and $d(n)$. There is an important philosophical difference, however, between minimizing the least squares error and the mean-square error. Minimizing the mean-square error produces the same set of filter coefficients for all sequences that have the same statistics. Therefore, the coefficients do not depend on the incoming data, only on their ensemble average. With the least squares error, on the other hand, we are minimizing a squared error that depends explicitly on the specific values of $x(n)$ and $d(n)$. Consequently, for different signals we get different filters. As a result, the filter coefficients that minimize the least squares error will be optimal for the given data rather than statistically optimal over a particular class of process. In other words, different realizations of $x(n)$ and $d(n)$ will lead to different solutions, even if the statistics of these sequences are the same. In this section, we will look at the filters that are derived by minimizing a weighted least squares error, and derive an efficient algorithm for performing this minimization known as *recursive least squares*.

9.4.1 Exponentially Weighted RLS

Let us reconsider the design of an FIR adaptive Wiener filter and find the filter coefficients

$$\mathbf{w}_n = [w_n(0),\ w_n(1),\ \ldots,\ w_n(p)]^T$$

that minimize, at time n, the weighted least squares error

$$\mathcal{E}(n) = \sum_{i=0}^{n} \lambda^{n-i} |e(i)|^2 \tag{9.89}$$

where $0 < \lambda \leq 1$ is an exponential weighting (forgetting) factor and

$$e(i) = d(i) - y(i) = d(i) - \mathbf{w}_n^T \mathbf{x}(i) \tag{9.90}$$

Note that $e(i)$ is the difference between the desired signal $d(i)$ and the filtered output at time i, using the *latest* set of filter coefficients, $w_n(k)$. Thus, in minimizing $\mathcal{E}(n)$ it is assumed that

the weights $\mathbf{w}_n$ are held constant over the entire observation interval $[0, n]$. To find the coefficients that minimize $\mathcal{E}(n)$ we proceed exactly as we have done many times before by setting the derivative of $\mathcal{E}(n)$ with respect to $w_n^*(k)$ equal to zero for $k = 0, 1, \ldots, p$. Thus, we have

$$\frac{\partial \mathcal{E}(n)}{\partial w_n^*(k)} = \sum_{i=0}^{n} \lambda^{n-i} e(i) \frac{\partial e^*(i)}{\partial w_n^*(k)} = -\sum_{i=0}^{n} \lambda^{n-i} e(i) x^*(i-k) = 0 \tag{9.91}$$

for $k = 0, 1, \ldots, p$. Incorporating Eq. (9.90) into Eq. (9.91) yields

$$\sum_{i=0}^{n} \lambda^{n-i} \Big\{ d(i) - \sum_{l=0}^{p} w_n(l) x(i-l) \Big\} x^*(i-k) = 0$$

Interchanging the order of summation and rearranging terms we have

$$\sum_{l=0}^{p} w_n(l) \left[\sum_{i=0}^{n} \lambda^{n-i} x(i-l) x^*(i-k) \right] = \sum_{i=0}^{n} \lambda^{n-i} d(i) x^*(i-k) \tag{9.92}$$

We may express these equations concisely in matrix form as follows:

$$\boxed{\mathbf{R}_x(n)\mathbf{w}_n = \mathbf{r}_{dx}(n)} \tag{9.93}$$

where $\mathbf{R}_x(n)$ is a $(p+1) \times (p+1)$ exponentially weighted deterministic autocorrelation matrix for $x(n)$

$$\mathbf{R}_x(n) = \sum_{i=0}^{n} \lambda^{n-i} \mathbf{x}^*(i) \mathbf{x}^T(i)$$

with $\mathbf{x}(i)$ the data vector

$$\mathbf{x}(i) = [x(i),\ x(i-1),\ \ldots,\ x(i-p)]^T$$

and where $\mathbf{r}_{dx}(n)$ is the deterministic cross-correlation between $d(n)$ and $x(n)$,

$$\mathbf{r}_{dx}(n) = \sum_{i=0}^{n} \lambda^{n-i} d(i) \mathbf{x}^*(i) \tag{9.94}$$

Eq. (9.93) is referred to as the *deterministic normal equations.*

Having derived the equations that define the optimum filter coefficients, we may now evaluate the minimum squared error. With

$$\begin{aligned} \mathcal{E}(n) &= \sum_{i=0}^{n} \lambda^{n-i} e(i) e^*(i) = \sum_{i=0}^{n} \lambda^{n-i} e(i) \Big\{ d(i) - \sum_{l=0}^{p} w_n(l) x(i-l) \Big\}^* \\ &= \sum_{i=0}^{n} \lambda^{n-i} e(i) d^*(i) - \sum_{l=0}^{p} w_n^*(l) \sum_{i=0}^{n} \lambda^{n-i} e(i) x^*(i-l) \end{aligned}$$

note that if $w_n(l)$ are the coefficients that minimize the squared error, then it follows from Eq. (9.91) that the second term is zero and the minimum error is

$$\begin{aligned} \{\mathcal{E}(n)\}_{\min} &= \sum_{i=0}^{n} \lambda^{n-i} e(i) d^*(i) \\ &= \sum_{i=0}^{n} \lambda^{n-i} \Big\{ d(i) - \sum_{l=0}^{p} w_n(l) x(i-l) \Big\} d^*(i) \\ &= \sum_{i=0}^{n} \lambda^{n-i} |d(i)|^2 - \sum_{l=0}^{p} w_n(l) \sum_{i=0}^{n} \lambda^{n-i} x(i-l) d^*(i) \end{aligned}$$

Alternatively, we may express the minimum error in vector form as follows:

$$\boxed{\{\mathcal{E}(n)\}_{\min} = \|\mathbf{d}(n)\|_{\lambda}^2 - \mathbf{r}_{dx}^H(n)\mathbf{w}_n} \tag{9.95}$$

where $\|\mathbf{d}(n)\|_{\lambda}^2$ is the weighted norm of the vector $\mathbf{d}(n) = \big[d(n),\ d(n-1),\ \ldots,\ d(0)\big]^T$.

Since $\mathbf{R}_x(n)$ and $\mathbf{r}_{dx}(n)$ both depend on n, instead of solving the deterministic normal equations directly, for each value of n, we will derive a recursive solution of the form

$$\mathbf{w}_n = \mathbf{w}_{n-1} + \Delta\mathbf{w}_{n-1}$$

where $\Delta\mathbf{w}_{n-1}$ is a correction that is applied to the solution at time $n-1$. Since

$$\mathbf{w}_n = \mathbf{R}_x^{-1}(n)\mathbf{r}_{dx}(n)$$

this recursion will be derived by first expressing $\mathbf{r}_{dx}(n)$ in terms of $\mathbf{r}_{dx}(n-1)$, and then deriving a recursion that allows us to evaluate $\mathbf{R}_x^{-1}(n)$ in terms of $\mathbf{R}_x^{-1}(n-1)$ and the new data vector $\mathbf{x}(n)$. From Eq. (9.94) it follows that the cross-correlation may be updated recursively as follows:

$$\mathbf{r}_{dx}(n) = \lambda\mathbf{r}_{dx}(n-1) + d(n)\mathbf{x}^*(n) \tag{9.96}$$

Similarly, the autocorrelation matrix may be updated from $\mathbf{R}_x(n-1)$ and the new data vector $\mathbf{x}(n)$ using the recursion

$$\mathbf{R}_x(n) = \lambda\mathbf{R}_x(n-1) + \mathbf{x}^*(n)\mathbf{x}^T(n) \tag{9.97}$$

However, since it is the inverse of $\mathbf{R}_x(n)$ that we are interested in, we may apply *Woodbury's Identity*, Eq. (2.29), to Eq. (9.97) to get the desired recursion. Woodbury's identity, repeated below for convenience, is

$$(\mathbf{A} + \mathbf{u}\mathbf{v}^H)^{-1} = \mathbf{A}^{-1} - \frac{\mathbf{A}^{-1}\mathbf{u}\mathbf{v}^H\mathbf{A}^{-1}}{1 + \mathbf{v}^H\mathbf{A}^{-1}\mathbf{u}}$$

Note that if we set $\mathbf{A} = \lambda\mathbf{R}_x(n-1)$ and $\mathbf{u} = \mathbf{v} = \mathbf{x}^*(n)$ then we obtain the following recursion for the inverse of $\mathbf{R}_x(n)$:

$$\mathbf{R}_x^{-1}(n) = \lambda^{-1}\mathbf{R}_x^{-1}(n-1) - \frac{\lambda^{-2}\mathbf{R}_x^{-1}(n-1)\mathbf{x}^*(n)\mathbf{x}^T(n)\mathbf{R}_x^{-1}(n-1)}{1 + \lambda^{-1}\mathbf{x}^T(n)\mathbf{R}_x^{-1}(n-1)\mathbf{x}^*(n)} \tag{9.98}$$

To simplify notation, we will let $\mathbf{P}(n)$ denote the inverse of the autocorrelation matrix at time n,

$$\mathbf{P}(n) = \mathbf{R}_x^{-1}(n)$$

and define what is referred to as the *gain vector*, $\mathbf{g}(n)$, as follows:

$$\boxed{\mathbf{g}(n) = \frac{\lambda^{-1}\mathbf{P}(n-1)\mathbf{x}^*(n)}{1 + \lambda^{-1}\mathbf{x}^T(n)\mathbf{P}(n-1)\mathbf{x}^*(n)}} \tag{9.99}$$

Incorporating these definitions into Eq. (9.98) we have

$$\boxed{\mathbf{P}(n) = \lambda^{-1}\Big[\mathbf{P}(n-1) - \mathbf{g}(n)\mathbf{x}^T(n)\mathbf{P}(n-1)\Big]} \tag{9.100}$$

An interesting and useful expression for the gain vector may be derived from Eq. (9.99) as follows. Cross-multiplying to eliminate the denominator on the right side of Eq. (9.99), we have

$$\mathbf{g}(n) + \lambda^{-1}\mathbf{g}(n)\mathbf{x}^T(n)\mathbf{P}(n-1)\mathbf{x}^*(n) = \lambda^{-1}\mathbf{P}(n-1)\mathbf{x}^*(n)$$

Next, bringing the second term on the left to the right side of the equation yields

$$\mathbf{g}(n) = \lambda^{-1}\Big[\mathbf{P}(n-1) - \mathbf{g}(n)\mathbf{x}^T(n)\mathbf{P}(n-1)\Big]\mathbf{x}^*(n)$$

Finally, from Eq. (9.100) we see that the term multiplying $\mathbf{x}^*(n)$ is $\mathbf{P}(n)$ and we have

$$\mathbf{g}(n) = \mathbf{P}(n)\mathbf{x}^*(n) \tag{9.101}$$

Thus, the gain vector is the solution to the linear equations

$$\boxed{\mathbf{R}_x(n)\mathbf{g}(n) = \mathbf{x}^*(n)} \tag{9.102}$$

which is the same as the deterministic normal equations for $\mathbf{w}_n$ in Eq. (9.93) except that the cross-correlation vector $\mathbf{r}_{dx}(n)$ has been replaced with the data vector $\mathbf{x}^*(n)$.

To complete the recursion, we must derive the time-update equation for the coefficient vector $\mathbf{w}_n$. With

$$\mathbf{w}_n = \mathbf{P}(n)\mathbf{r}_{dx}(n)$$

it follows from the update for $\mathbf{r}_{dx}(n)$ given in Eq. (9.94) that

$$\mathbf{w}_n = \lambda\mathbf{P}(n)\mathbf{r}_{dx}(n-1) + d(n)\mathbf{P}(n)\mathbf{x}^*(n) \tag{9.103}$$

Next, incorporating the update for $\mathbf{P}(n)$ given in Eq. (9.100) into the first term on the right side of Eq. (9.103), and setting $\mathbf{P}(n)\mathbf{x}^*(n) = \mathbf{g}(n)$ we have

$$\mathbf{w}_n = \Big[\mathbf{P}(n-1) - \mathbf{g}(n)\mathbf{x}^T(n)\mathbf{P}(n-1)\Big]\mathbf{r}_{dx}(n-1) + d(n)\mathbf{g}(n)$$

Finally, recognizing that $\mathbf{P}(n-1)\mathbf{r}_{dx}(n-1) = \mathbf{w}_{n-1}$ it follows that

$$\mathbf{w}_n = \mathbf{w}_{n-1} + \mathbf{g}(n)\Big[d(n) - \mathbf{w}_{n-1}^T\mathbf{x}(n)\Big]$$

which may be written as

$$\boxed{\mathbf{w}_n = \mathbf{w}_{n-1} + \alpha(n)\mathbf{g}(n)} \tag{9.104}$$

where

$$\alpha(n) = d(n) - \mathbf{w}_{n-1}^T\mathbf{x}(n) \tag{9.105}$$

is the difference between $d(n)$ and the estimate of $d(n)$ that is formed by applying the *previous* set of filter coefficients, $\mathbf{w}_{n-1}$, to the new data vector, $\mathbf{x}(n)$. This sequence, called the *a priori error*, is the error that would occur if the filter coefficients were not updated. The *a posteriori error*, on the other hand, is the error that occurs after the weight vector is updated,

$$e(n) = d(n) - \mathbf{w}_n^T\mathbf{x}(n)$$

Table 9.6 *Recursive Least Squares*

Parameters:	p = Filter order
	λ = Exponential weighting factor
	δ = Value used to initialize $\mathbf{P}(0)$
Initialization:	$\mathbf{w}_0 = \mathbf{0}$
	$\mathbf{P}(0) = \delta^{-1}\mathbf{I}$
Computation:	For $n = 1, 2, \ldots$ compute
	$\mathbf{z}(n) = \mathbf{P}(n-1)\mathbf{x}^*(n)$
	$\mathbf{g}(n) = \dfrac{1}{\lambda + \mathbf{x}^T(n)\mathbf{z}(n)}\mathbf{z}(n)$
	$\alpha(n) = d(n) - \mathbf{w}_{n-1}^T\mathbf{x}(n)$
	$\mathbf{w}_n = \mathbf{w}_{n-1} + \alpha(n)\mathbf{g}(n)$
	$\mathbf{P}(n) = \dfrac{1}{\lambda}\left[\mathbf{P}(n-1) - \mathbf{g}(n)\mathbf{z}^H(n)\right]$

Note that when $\alpha(n)$ is small, the current set of filter coefficients are close to their optimal values (in a least squares sense), and only a small correction needs to be applied to the coefficients. On the other hand, when $\alpha(n)$ is large, the current set of filter coefficients are not performing well in estimating $d(n)$ and a large correction must be applied to update the coefficients.

One final simplification may be realized if we note that, in the evaluation of the gain vector $\mathbf{g}(n)$, and the inverse autocorrelation matrix $\mathbf{P}(n)$, it is necessary to compute the product

$$\boxed{\mathbf{z}(n) = \mathbf{P}(n-1)\mathbf{x}^*(n)} \tag{9.106}$$

Therefore, we may explicitly evaluate this *filtered information vector* and then use it in the calculation of both $\mathbf{g}(n)$ and $\mathbf{P}(n)$. Equations (9.99), (9.100), (9.104), (9.105), and (9.106) form what is known as the exponentially weighted *Recursive Least Squares* (RLS) algorithm, which is summarized in Table 9.6. The special case of $\lambda = 1$ is referred to as the *growing window RLS algorithm.*

The last issue that needs to be addressed concerns the initialization of the RLS algorithm. Since the RLS algorithm involves the recursive updating of the vector $\mathbf{w}_n$ and the inverse autocorrelation matrix $\mathbf{P}(n)$, initial conditions for both of these terms are required. There are two ways that this initialization is typically performed. The first is to build up the autocorrelation matrix recursively according to Eq. (9.97) until it is of full rank (typically $p + 1$ input vectors), and then compute the inverse directly

$$\mathbf{P}(0) = \left[\sum_{i=-p}^{0} \lambda^{-i}\mathbf{x}^*(i)\mathbf{x}^T(i)\right]^{-1}$$

Evaluating the cross-correlation vector $\mathbf{r}_{dx}(0)$ in the same manner,

$$\mathbf{r}_{dx}(0) = \sum_{i=-p}^{0} \lambda^{-i} d(i)\mathbf{x}^*(i)$$

we may then initialize $\mathbf{w}_0$ by setting $\mathbf{w}_0 = \mathbf{P}(0)\mathbf{r}_{dx}(0)$. The advantage of this approach is that optimality is preserved at each step since the RLS algorithm is initialized at $n = 0$ with the vector $\mathbf{w}_0$ that minimizes the weighted least squares error $\mathcal{E}(0)$. A disadvantage, however, is that it requires the direct inversion of $\mathbf{R}_x(0)$, which requires on the order of $(p+1)^3$ operations. In addition, with this approach there will be a delay of $p+1$ samples before any updates are performed. Another approach that may be used is to initialize the autocorrelation matrix as follows:

$$\mathbf{R}_x(0) = \delta\mathbf{I}$$

where δ is a small positive constant, i.e., $\mathbf{P}(0) = \delta^{-1}\mathbf{I}$, and to initialize the weight vector to zero,

$$\mathbf{w}_0 = \mathbf{0}$$

The disadvantage of this approach is that it introduces a bias in the least squares solution [20]. However, with an exponential weighting factor, $\lambda < 1$, this bias goes to zero as n increases. A MATLAB program for the RLS algorithmis given in Fig. 9.24.

Unlike the LMS algorithm, which requires on the order of p multiplications and additions, the RLS algorithm requires on the order of p^2 operations. Specifically, the evaluation of $\mathbf{z}(n)$ requires $(p+1)^2$ multiplications, computing the gain vector $\mathbf{g}(n)$ requires $2(p+1)$ multiplications, finding the a priori error $\alpha(n)$ requires another $p+1$ multiplications, and the update of the inverse autocorrelation matrix $\mathbf{P}(n)$ requires $2(p+1)^2$ multiplications for a total of $3(p+1)^2 + 3(p+1)$. There is a similar number of additions. What is gained with this increase in computational complexity over the LMS algorithm, however, is an increase in performance. Generally, the RLS algorithm converges faster than the LMS algorithm and is less sensitive to eigenvalue disparities in the autocorrelation matrix of $x(n)$ for stationary

The RLS Algorithm

```
function [A,E] = rls(x,d,nord,lambda)
%
   delta=0.001;
   X=convm(x,nord);
   [M,N] = size(X);
   if nargin < 4,   lambda = 1.0;   end
   P=eye(N)/delta;
   W(1,:)=zeros(1,N);
   for k=2:M-nord+1;
       z=P*X(k,:)';
       g=z/(lambda+X(k,:)*z);
       alpha=d(k)-X(k,:)*W(k-1,:).';
       W(k,:)=W(k-1,:)+alpha*g.';
       P=(P-g*z.')/lambda;
   end;
```

Figure 9.24 *A* MATLAB *program for the RLS algorithm.*

processes. On the other hand, without exponential weighting, RLS does not perform very well in tracking nonstationary processes. This is due to the fact that, with $\lambda = 1$, all of the data is equally weighted in estimating the correlations. Although exponential weighting improves the tracking characteristics of RLS, it is not clear how to choose λ and, in some cases, the LMS algorithm may have better tracking properties [6,13] For a more complete discussion of the RLS algorithm, the reader is referred to Reference [20].

Example 9.4.1 *Linear Prediction Using RLS*

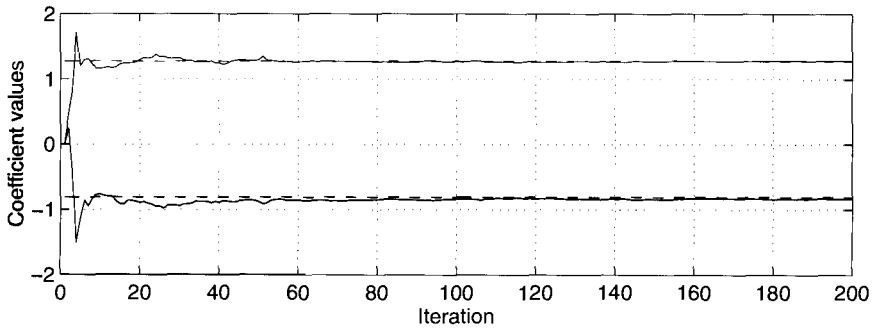
In this example, we compare LMS and RLS for the problem of linear prediction introduced in Example 9.2.1 (p. 509). With $x(n)$ generated according to the difference equation

$$x(n) = 1.2728x(n-1) - 0.81x(n-2) + v(n)$$

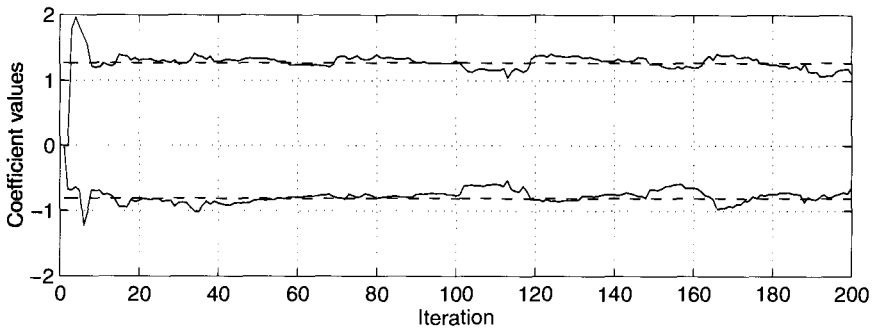
where $v(n)$ is unit variance white noise, we will consider a second-order RLS predictor of the form

$$\hat{x}(n) = w_n(1)x(n-1) + w_n(2)x(n-2)$$

Shown in Fig. 9.25*a* is a plot of the predictor coefficients $w_n(1)$ and $w_n(2)$ versus n using the growing window RLS algorithm ($\lambda = 1$). Compared to the LMS algorithm in Fig. 9.10,



(a)



(b)

Figure 9.25 *Linear prediction of a second-order autoregressive process using recursive least squares. The dashed lines indicate the correct values for the coefficients. (a) Growing window RLS. (b) Exponentially weighted RLS with* $\lambda = 0.95$.

what is clearly evident is that the RLS algorithm has a much higher rate of convergence. Fig. 9.25*b*, on the other hand, shows the predictor coefficients using the exponential window RLS algorithm with $\lambda = 0.95$. What we observe is an increase in the fluctuations of the weights with the exponential window. This is due to the decrease in the effective length of the data window that is used to evaluate $\mathbf{R}_x(n)$. With a growing window RLS algorithm, if the signal properties are not changing in time then $\mathbf{R}_x(n)$ and the weights $\mathbf{w}_n$ become more stable as n increases.

As a final comparison, let us evaluate what happens if $x(n)$ is generated according to the time-varying difference equation

$$x(n) = a_n(1)x(n-1) - 0.81x(n-2) + v(n)$$

where $v(n)$ is unit variance white noise and $a_n(1)$ is a time-varying coefficient of the following form

$$a_n(1) = \begin{cases} 1.2728 & ; \quad 0 \leq n < 100 \\ 0 & ; \quad 100 \leq n \leq 200 \end{cases}$$

This corresponds to a filter having a pair of poles at radius $r = 0.9$ and angle $\omega_0 = \pi/4$ when $a_n(1) = 1.2728$ and radius $r = 0.9$ and angle $\omega_0 = \pi/2$ when $a_n(1) = 0$. Thus, at time $n = 100$ the poles move from $\omega_0 = \pi/4$ to $\omega_0 = \pi/2$. Shown in Fig. 9.26*a* is a plot of $w_n(1)$ versus n for $\lambda = 1$. What we see is that the growing window RLS algorithm is not able to track the time-variations effectively due to the infinite length window (infinite memory). Shown in Fig. 9.26*b*, on the other hand, is a plot of $w_n(1)$ versus n for $\lambda = 0.9$. Note that with this decrease in the weighting factor, the RLS algorithm is able to more accurately track the true values. Finally, shown in Fig. 9.26*c* is a plot of the coefficients obtained with the LMS algorithm using a step size $\mu = 0.02$. As we see, the performance of the LMS algorithm in tracking nonstationary processes is similar to the exponentially weighted RLS algorithm.

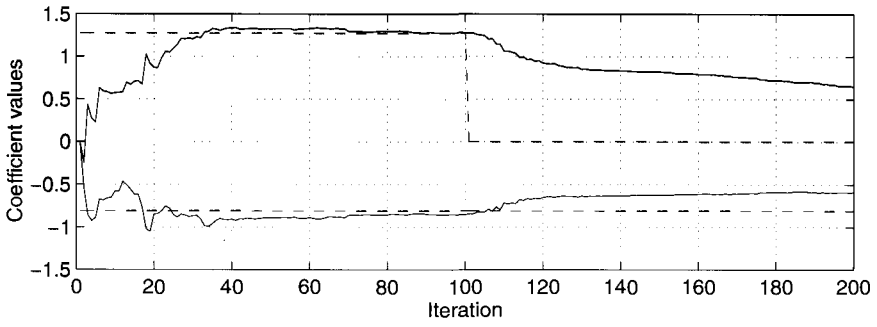
9.4.2 Sliding Window RLS

The RLS algorithm minimizes the exponentially weighted least squares error $\mathcal{E}(n)$ given in Eq. (9.89). With a growing window RLS algorithm ($\lambda = 1$), each of the squared errors $|e(i)|^2$ from $i = 0$ to $i = n$ are equally weighted, whereas with an exponentially weighted RLS algorithm ($\lambda < 1$), the squared errors $|e(i)|^2$ become less important for values of i that are small compared to n. In both cases, however, the RLS algorithm has infinite memory in the sense that all of the data beginning from time $n = 0$ will affect the values of the coefficients $\mathbf{w}_n$. In some applications this may not be desirable. For example, for a nonstationary process whose statistics are changing rapidly in time, a small value for λ will be necessary in order to track the process. An alternative, however, is to minimize the sum of the squares of $e(i)$ over a finite window,[8]

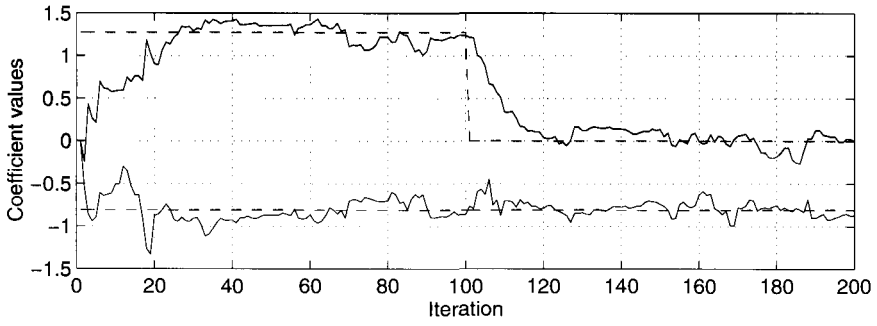
$$\mathcal{E}_L(n) = \sum_{i=n-L}^{n} |e(i)|^2 \tag{9.107}$$

In addition to being able to more easily track nonstationary processes, the finite window allows for any data *outliers* to be *forgotten* after a finite number of iterations.

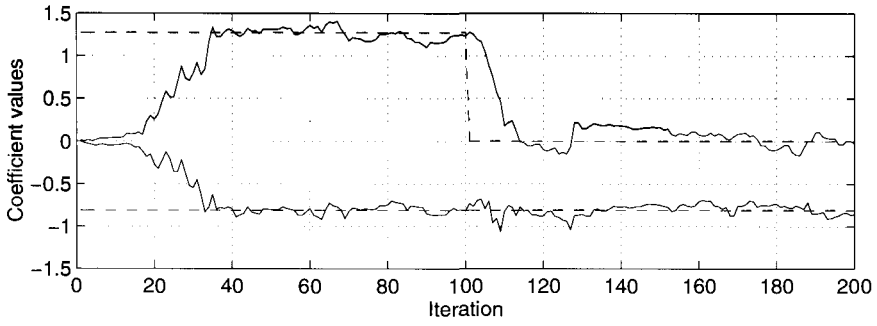
[8]Note that in the special case of $L = 0$, a gradient descent approach to minimize $\mathcal{E}_0(n)$ results in the LMS algorithm.



(a)



(b)



(c)

Figure 9.26 *Linear prediction of a nonstationary second-order autoregressive random process. The dashed lines indicate the correct values for the coefficients. (a) Growing window RLS. (b) Exponentially weighted RLS with* $\lambda = 0.9$. *(c) LMS algorithm with a step size* $\mu = 0.02$.

Following the same steps used to derive the deterministic normal equations for minimizing the exponentially weighted squared error, the filter coefficients $\mathbf{w}_n$ that minimize $\mathcal{E}_L(n)$ are found to be solutions to the equations

$$\mathbf{R}_x(n)\mathbf{w}_n = \mathbf{r}_{dx}(n) \tag{9.108}$$

where

$$\mathbf{R}_x(n) = \sum_{i=n-L}^{n} \mathbf{x}^*(i)\mathbf{x}^T(i) \tag{9.109}$$

and

$$\mathbf{r}_{dx}(n) = \sum_{i=n-L}^{n} d(i)\mathbf{x}^*(i)$$

Using a procedure similar to that used to derive the growing window RLS algorithm, a *sliding window* RLS algorithm may be developed that solves Eq. (9.108) recursively with a computational complexity on the order of p^2 operations. The sliding window RLS algorithm consists of the following two steps.

1. Given the solution $\mathbf{w}_{n-1}$ to Eq. (9.108) at time $n-1$, with the new data value $x(n)$, the weight vector $\widetilde{\mathbf{w}}_n$ is found that minimizes the error

$$\mathcal{E}_{L+1}(n) = \sum_{i=n-L-1}^{n} |e(i)|^2 \tag{9.110}$$

2. The weight vector $\mathbf{w}_n$ that minimizes $\mathcal{E}_L(n)$ is then determined by discarding the last data point, $x(n-L-1)$.

In the first step, note that we find the vector $\widetilde{\mathbf{w}}_n$ that minimizes the error $\mathcal{E}_{L+1}(n)$ that results from the addition of one additional data value. Therefore, we may use the growing window RLS algorithm to find $\widetilde{\mathbf{w}}_n$ from $\mathbf{w}_{n-1}$ as follows:

$$\begin{aligned}
\mathbf{g}(n+1) &= \frac{1}{1+\mathbf{x}^T(n)\mathbf{P}(n-1)\mathbf{x}(n)}\mathbf{P}(n-1)\mathbf{x}^*(n) \\
\widetilde{\mathbf{w}}_n &= \mathbf{w}_{n-1} + \mathbf{g}(n)[d(n) - \mathbf{w}_{n-1}\mathbf{x}^*(n)] \\
\widetilde{\mathbf{P}}(n) &= \mathbf{P}(n-1) - \mathbf{g}(n)\mathbf{x}^T(n)\mathbf{P}(n-1)
\end{aligned}$$

Note that $\widetilde{\mathbf{P}}(n)$ is the inverse of the matrix

$$\widetilde{\mathbf{R}}_x(n) = \sum_{k=n-L-1}^{n} \mathbf{x}^*(k)\mathbf{x}^T(k)$$

which is based on $L+2$ data values, and that $\widetilde{\mathbf{w}}_n$ is the solution to

$$\widetilde{\mathbf{R}}_x(n)\widetilde{\mathbf{w}}_n = \widetilde{\mathbf{r}}_{dx}(n)$$

where

$$\widetilde{\mathbf{r}}_{dx}(n) = \sum_{k=n-L-1}^{n+1} d(k)\mathbf{x}^*(k)$$

In the second step of the recursion, the last data point $x(n-L-1)$ is discarded to restore the $(L+1)$ point window. This is accomplished with a slight modification to the growing window RLS algorithm. Specifically, we begin with the matrix update

$$\mathbf{R}(n) = \widetilde{\mathbf{R}}(n) - \mathbf{x}^*(n-L)\mathbf{x}^T(n-L-1)$$

and the update of $\mathbf{r}_{dx}(n)$

$$\mathbf{r}_{dx}(n) = \widetilde{\mathbf{r}}_{dx}(n) - d(n-L-1)\mathbf{x}^*(n-L-1)$$

Finally, using the matrix inversion lemma and retracing the steps used to derive the RLS

algorithm we obtain the following update equations

$$\begin{aligned}\tilde{\mathbf{g}}(n+1) &= \frac{1}{1-\mathbf{x}^T(n-L-1)\tilde{\mathbf{P}}(n)\mathbf{x}^*(n-L-1)}\tilde{\mathbf{P}}(n)\mathbf{x}^*(n-L-1)\\ \mathbf{w}_n &= \tilde{\mathbf{w}}_n - \tilde{\mathbf{g}}(n)[d(n-L-1) - \tilde{\mathbf{w}}_n\mathbf{x}^*(n-L-1)]\\ \mathbf{P}(n) &= \tilde{\mathbf{P}}(n) + \tilde{\mathbf{g}}(n)\mathbf{x}^T(n-L-1)\tilde{\mathbf{P}}(n)\end{aligned}$$

which yields the desired solution and completes the derivation of the sliding window RLS algorithm. Compared to exponentially weighted RLS, the sliding window RLS requires about twice the number of multiplications and additions and, in addition, requires that $p + L$ values of $x(n)$ be stored. This storage requirement may potentially be a problem for long windows.

9.4.3 Summary

In this chapter, we have looked at a number of techniques for processing nonstationary signals using adaptive filters. These techniques have been extensively used in a variety of applications including system identification, signal modeling, spectrum estimation, noise cancellation, and adaptive equalization. Our treatment of adaptive filtering began with the design of an FIR filter to minimize the mean-square error $\xi(n) = E\{|e(n)|^2\}$ between a desired process $d(n)$ and an estimate of this process that is formed by filtering another process $x(n)$. First, we considered a steepest descent approach to minimize $\xi(n)$. However, since the gradient of $\xi(n)$ involves an expectation of $e(n)\mathbf{x}^*(n)$, this approach requires knowledge of the statistics of $x(n)$ and $d(n)$ and, therefore, is of limited use in practice. Next, we replaced the ensemble average $E\{|e(n)|^2\}$ with the instantaneous squared error $|e(n)|^2$. This led to the LMS algorithm, a simple and often effective algorithm that does not require any ensemble averages to be known. For wide-sense stationary processes, the LMS algorithm converges in the mean if the step size is positive and no larger than $2/\lambda_{\max}$, and it converges in the mean-square if the step size is positive and no larger than $2/\mathrm{tr}(\mathbf{R}_x)$. We then looked at several modifications of the LMS algorithm. The first was the normalized LMS algorithm, which simplifies the selection of the step size to ensure that the coefficients converge. Next was the leaky LMS algorithm, which is useful in overcoming the problems that occur when the autocorrelation matrix of the input process is singular. We then looked at the block LMS algorithm and the sign algorithms, which are designed to increase the efficiency of the LMS algorithm. In the block LMS algorithm, the filter coefficients are held constant over blocks of length L, which allows for the use of fast convolution algorithms to compute the filter output. The sign algorithms, on the other hand, achieve their simplicity by replacing $e(n)$ with $\mathrm{sgn}\{e(n)\}$, or $\mathbf{x}(n)$ with $\mathrm{sgn}\{\mathbf{x}(n)\}$, or both. Finally, we discussed how it would be beneficial to consider using a variable step size in the LMS algorithm. Ideally, when the filter coefficients are far from their optimum values, the step size should be large. Then, as the cofficients begin to converge to the optimum solution, the step size should be small in order to reduce the excess mean-square error. It is difficult, however, to define a set of rules for adapting the step size so that the adaptive filter will have a small excess mean-square error while, at the same time, maintaining the ability of the filter to respond quickly to changes in the signal statistics. Finally, we looked at the lattice filter as an alternative structure to use as an adaptive filter. Due to the orthogonalization of the input process, the gradient adaptive lattice filter converges more rapidly than the LMS adaptive

filter, and tends to be less sensitive to the eigenvalue spread in the autocorrelation matrix of $x(n)$.

Following the FIR adaptive filters, we then turned our attention briefly to the design of adaptive recursive filters. These filters are much more complex than the FIR filters. In addition, since the mean-square error $\xi(n)$ is a nonlinear function of the filter coefficients, the adaptive filter may not converge or it may converge to a local rather than the global minimum. Also, since they are recursive, it is possible for these filters to become unstable. As a result, adaptive recursive filters are not as widely used in applications as their FIR counterparts.

This chapter concluded with a derivation of the recursive least squares algorithm to minimize a deterministic least squares error. Three different forms of RLS algorithm were derived: the growing window RLS algorithm, the exponentially weighted RLS algorithm, and the sliding window RLS algorithm. Although computationally more complex than the LMS adaptive filter, for wide-sense stationary processes the growing window RLS algorithm converges much more rapidly. However, in order to effectively track a nonstationary process, it is necessary to use either the exponentially weighted RLS algorithm or the sliding window RLS algorithm.

The treatment of adaptive filtering algorithms in this chapter is, by no means, complete. Many other approaches to FIR and IIR adaptive filtering have been developed and many papers have been published that analyze the performance of adaptive filtering algorithms and evaluate their effectiveness in applications [30]. A few of the more notable omissions include the fast RLS recursive algorithms [2,8,7], recursive least squares lattice filters [16,23], and nonlinear adaptive filters [25]. For a general and more comprehensive treatment of adaptive filters, one may consult any one of a number of excellent textbooks [1,3,10,20,37].

References

1. S. T. Alexander, *Adaptive Signal Processing: Theory and Applications*, Springer-Verlag, New York, 1986.
2. S. T. Alexander, "Fast adaptive filters: A geometric approach," *IEEE ASSP Mag.*, vol. 3, no. 4, pp. 18–28, October 1986.
3. M. G. Bellanger, *Adaptive Digital Filters and Signal Analysis*, Marcel Dekker, Inc., New York, 1987.
4. N. J. Bershad, "Analysis of the normalized LMS algorithm with Gaussian inputs," *IEEE Trans. Acoust., Speech, Sig. Proc.*, vol. ASSP-34, pp. 793–806, 1986.
5. N. J. Bershad, "On the optimum data non-linearity in LMS adaptation," *IEEE Trans. Acoust., Speech, Sig. Proc.*, vol. ASSP-34, pp. 69–76, February 1986.
6. N. J. Bershad and O. Macchi, "Comparison of RLS and LMS algorithms for tracking a chirped signal," *Proc. 1989 Int. Conf. Acoust., Speech, Sig. Proc.*, pp. 896–899, 1989.
7. G. Carayannis, D. G. Manolakis, and N. Kalouptsidis, "A fast sequential algorithm for least-squares filtering and prediction," *IEEE Trans. Acoust., Speech, Sig. Proc.*, vol. ASSP-31, no. 6, pp. 1394–1402, December 1983.
8. J. M. Cioffi and T. Kailath, "Fast recursive-least-squares transversal filters for adaptive filtering," *IEEE Trans. Acoust., Speech, Sig. Proc.*, vol. ASSP-32, no. 2, pp. 304–337, April 1984.
9. G. A. Clark, S. K. Mitra, and S. R. Parker, "Block implementation of adaptive digital filters," *IEEE Trans. Circuits and Systems*, vol. CAS-28, pp. 584–592, 1981.
10. P. M. Clarkson, *Optimal and Adaptive Signal Processing*, CRC Press, Boca Raton, FL, 1993.
11. T. Claasen and W. Mecklenbrauker, "Comparison of the convergence of two algorithms for adaptive FIR digital filters," *IEEE Trans. Acoust., Speech, Sig. Proc.*, vol. ASSP-29, pp. 670–678, June 1981.

12. C. F. N. Cowan and P. M. Grant, *Adaptive Filters*, Prentice-Hall, Englewood Cliffs, NJ, 1985.
13. E. Eleftheriou and D. Falconer, "Tracking properties and steady state performance of RLS adaptive filtering algorithms," *IEEE Trans. Acoust., Speech, Sig. Proc.*, vol. ASSP-34, no. 5, pp. 1097–1110, October 1986.
14. S. J. Elliott and P. A. Nelson, "Active noise control," *Sig. Proc. Magazine*, vol. 10, no. 4, October 1993, pp. 12–35.
15. P. L. Feintuch, "An adaptive recursive LMS filter," *Proc. IEEE*, pp. 1622–1624, November 1976.
16. B. Freidlander, "Lattice filters for adaptive processing," *Proc. IEEE*, vol. 70, no. 8, pp. 829–867, August 1982.
17. L. J. Griffiths, "A continuously-adaptive filter implemented as a lattice structure," *Proc. 1977 Int. Conf. Acoust., Speech, Sig. Proc.*, pp. 683–686, 1977.
18. L. J. Griffiths, "An adaptive lattice structure for noise cancellation," *Proc. 1978 Int. Conf. Acoust., Speech, Sig. Proc.*, pp. 87–90, 1978.
19. R. W. Harris, D. M. Chabries, and F. A. Bishop, "A variable step (VS) adaptive filter algorithm," *IEEE Trans. on Acoust., Speech, Sig. Proc.*, vol. ASSP-34, no. 2, pp. 309–316, April 1986.
20. S. Haykin, *Adaptive Filter Theory*, Prentice-Hall, Englewood-Cliffs, NJ, 1986.
21. M. L. Honig and D. G. Messerschmitt, "Convergence properties of an adaptive digital lattice filter," *IEEE Trans. Acoust., Speech, Sig. Proc.*, vol. ASSP-29, pp. 642–653, 1981.
22. S. Horvath Jr., "A new adaptive recursive LMS filter," in *Digital Signal Processing*, V. Cappellini and A.G. Constantinides, Eds., Academic Press, New York, pp. 21–26.
23. D. T. L. Lee, M. Morf, and B. Friedlander, "Recursive least squares ladder estimation algorithms," *IEEE Trans. Acoust., Speech, Sig. Proc.*, vol. ASSP-29, no. 3, pp. 467–481, June 1981.
24. R. W. Lucky, J. Salz, and E. J. Weldon Jr., *Principles of Data Communications*, McGraw-Hill, New York, 1968.
25. I. Pitas and A. N. Venetsanopoulos, *Nonlinear digital filters*, Kluwer Academic Publishers, Boston, 1990.
26. J. G. Proakis, *Digital Communication*, McGraw-Hill, New York, 1983.
27. E. H. Satorius and S. T. Alexander, "Channel equalization using adaptive lattice algorithms," *IEEE Trans. Comm.*, vol. COM-27, pp. 899–905, 1979.
28. W. A. Sethares, I. M. Y. Mareels, B. D. O. Anderson, C. R. Johnson, and R. R. Bitmead, "Excitation conditions for signed regressor least mean-squares adaptation," *IEEE Trans. Circuits Syst.*, vol. CAS-25, pp. 613–624, 1988.
29. J. Shynk, "Adaptive IIR filtering," *IEEE ASSP Magazine*, pp. 4–21, April 1989.
30. L. H. Sibul, Ed., *Adaptive Signal Processing*, IEEE Press, New York, 1987.
31. S. Stearns, "Error surfaces of recursive adaptive filters," *IEEE Trans. Acoust., Speech, Sig. Proc.*, vol. ASSP-29, pp. 763–766, 1981.
32. J. R. Treichler, "Adaptive algorithms for infinite impulse response filters," in *Adaptive Filters*, C. F. Cowan and P. M. Grant (Eds.), Prentice-Hall, Englewood Cliffs, NJ, 1985.
33. J. R. Treichler, C. R. Johnson Jr., and M. G. Larimore, *Theory and Design of Adaptive Filters*, John Wiley & Sons, 1987.
34. A. Weiss and D. Mitra, "Digital adaptive filters: Conditions for convergence, rates of convergence, effects of noise and errors arising from the implementation," *IEEE Trans. on Inf. Theory*, vol. IT-25, pp. 637–652.
35. S. White, "An adaptive recursive digital filter" *Proc. 9th Asilomar Conf. Circuits, Syst.*, pp. 21–25, 1975.
36. B. Widrow et. al., "Stationary and nonstationary learning characteristics of the LMS adaptive filter," *Proc. IEEE*, vol. 64, pp. 1151–1162, 1976.
37. B. Widrow and S. Stearns, *Adaptive Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1985.

9.5 PROBLEMS

9.1. In order for the steepest descent algorithm to converge, the step size must be in the range

$$0 < \mu < 2/\lambda_{\max}$$

However, in some cases it may be of interest to find the value of μ that gives the fastest rate of convergence. For a given step size μ, the rate of convergence for the weight vector $\mathbf{w}_n$ is dominated by the slowest converging mode in the expansion

$$\mathbf{w}_n = \mathbf{w} + \sum_{k=0}^{p} (1 - \mu\lambda_k)^n u_0(k)\mathbf{v}_k$$

(a) In terms of the eigenvalues λ_k, find the value for μ that maximizes the rate of convergence. In other words, find the value for μ that maximizes the rate of convergence of the slowest decaying mode.

(b) At what rate does the slowest mode decrease for the step size found in part (a)?

9.2. Suppose that the input to an adaptive linear predictor is white noise with an autocorrelation sequence $r_x(k) = \sigma_x^2\delta(k)$.

(a) Solve the normal equations and find the optimum pth-order one-step linear predictor, $\mathbf{w}$.

(b) Minimize the mean-square prediction error using the method of steepest descent with a step size $\mu = 1/(5\sigma_x^2)$ and an initial weight vector $\mathbf{w}_0 = [1, 1, \ldots, 1]^T$. Does the method of steepest descent converge to the solution found in part (a)?

9.3. Newton's method is an iterative algorithm that may be used to find the minimum of a nonlinear function. Applied to the minimization of the mean-square error

$$\xi(n) = E\{e^2(n)\}$$

where $e(n) = d(n) - \mathbf{w}^T\mathbf{x}(n)$, Newton's method is

$$\mathbf{w}_{n+1} = \mathbf{w}_n - \tfrac{1}{2}\mathbf{R}_x^{-1}\nabla\xi(n)$$

where $\mathbf{R}_x$ is the autocorrelation matrix of $x(n)$. Introducing a step-size parameter μ, Newton's method becomes

$$\mathbf{w}_{n+1} = \mathbf{w}_n - \tfrac{1}{2}\mu\mathbf{R}_x^{-1}\nabla\xi(n)$$

Comparing this to the steepest descent algorithm, we see that the step size μ is replaced with a matrix, $\mu\mathbf{R}_x^{-1}$, which alters the descent direction.

(a) For what values of μ is Newton's method stable, i.e., for what values of μ will $\mathbf{w}_n$ converge?

(b) What is the optimum value of μ, i.e., for what value of μ is the convergence the fastest?

(c) Suppose that we form an LMS version of Newton's method by replacing the gradient with a gradient estimate

$$\hat{\nabla}\xi(n) = \nabla e^2(n)$$

Derive the coefficient update equation that results from using this gradient estimate and describe how it differs from the LMS algorithm.

(d) Derive an expression that describes the time evolution of $E\{\mathbf{w}_n\}$ using the LMS Newton algorithm derived in part (c).

9.4. One way to derive the steepest descent algorithm for solving the normal equations $\mathbf{R}_x\mathbf{w} = \mathbf{r}_{dx}$ is to use a power series expansion for the inverse of $\mathbf{R}_x$. This expansion is

$$\mathbf{R}_x^{-1} = \mu \sum_{k=0}^{\infty} (\mathbf{I} - \mu\mathbf{R}_x)^k$$

where $\mathbf{I}$ is the identity matrix and μ is a positive constant. In order for this expansion to converge, $\mathbf{R}_x$ must be positive definite and the constant μ must lie in the range

$$0 < \mu < 2/\lambda_{\max}$$

where $\lambda_{\max}$ is the largest eigenvalue of $\mathbf{R}_x$.

(a) Let

$$\mathbf{R}_x^{-1}(n) = \mu \sum_{k=0}^{n} (\mathbf{I} - \mu\mathbf{R}_x)^k$$

be the nth-order approximation to $\mathbf{R}_x^{-1}$, and let

$$\mathbf{w}_n = \mathbf{R}_x^{-1}(n)\mathbf{r}_{dx}$$

be the nth-order approximation to the desired solution $\mathbf{w} = \mathbf{R}_x^{-1}\mathbf{r}_{dx}$. Express $\mathbf{R}_x^{-1}(n+1)$ in terms of $\mathbf{R}_x^{-1}(n)$, and show how this may be used to derive the steepest descent algorithm

$$\mathbf{w}_{n+1} = \mathbf{w}_n - \mu[\mathbf{R}_x\mathbf{w}_n - \mathbf{r}_{dx}]$$

(b) If the statistics of $x(n)$ are unknown, then $\mathbf{R}_x$ is unknown and the expansion for $\mathbf{R}_x^{-1}$ in part (a) cannot be evaluated. However, suppose that we approximate $\mathbf{R}_x = E\{\mathbf{x}(n)\mathbf{x}^T(n)\}$ at time n as follows

$$\hat{\mathbf{R}}_x(n) = \mathbf{x}(n)\mathbf{x}^T(n)$$

and use, as the nth-order approximation to $\mathbf{R}_x^{-1}$,

$$\hat{\mathbf{R}}_x^{-1}(n) = \mu \sum_{k=0}^{n} [\mathbf{I} - \mu\mathbf{x}(k)\mathbf{x}^T(k)]^k$$

Express $\hat{\mathbf{R}}_x^{-1}(n+1)$ in terms of $\hat{\mathbf{R}}_x^{-1}(n)$ and use this expression to derive a recursion for $\mathbf{w}_n$.

(c) Compare your recursion derived in part (b) to the LMS algorithm.

9.5. The convergence of a pth-order LMS adaptive filter depends on the eigenvalues of the autocorrelation matrix, $\mathbf{R}_x$, of the input process $x(n)$. These eigenvalues, in turn, depend upon the size, p, of $\mathbf{R}_x$. For example, it follows from the Bordering Theorem (see p. 48) that the maximum eigenvalue is a monotonically nondecreasing function of p, and the minimum eigenvalue is a monotonically nonincreasing function of p. In addition, it follows from the eigenvalue extremal property (see p. 97) that the maximum and minimum eigenvalues approach the maximum and minimum values of the power spectrum, $P_x(e^{j\omega})$, as $p \to \infty$,

$$\lambda_{\max} \to \max_{\omega} P_x(e^{j\omega}) \quad ; \quad \lambda_{\min} \to \min_{\omega} P_x(e^{j\omega})$$

Suppose that the input to an adaptive filter has an autocorrelation

$$r_x(k) = \alpha^{|k|} \quad ; \quad |\alpha| < 1$$

(a) Find the eigenvectors and eigenvalues of the 2×2 autocorrelation matrix $\mathbf{R}_x = \text{Toep}\{1, \alpha\}$. (Your answer will be in terms of α).

(b) Find the asymptotic values for the maximum and minimum eigenvalues of the $p \times p$ autocorrelation matrix $\mathbf{R}_x$ as $p \rightarrow \infty$.

(c) Find, as a function of α, the largest step size μ for convergence in the mean of the LMS algorithm, and find the slowest converging mode (assume that p is large).

9.6. The *condition number*, χ, of an autocorrelation matrix $\mathbf{R}_x$ may be bounded in terms of the power spectrum of the process $P_x(e^{j\omega})$ as follows (see Prob. 9.5),

$$\chi = \frac{\lambda_{\max}}{\lambda_{\min}} \leq \frac{\max\limits_{\omega} P_x(e^{j\omega})}{\min\limits_{\omega} P_x(e^{j\omega})}$$

(a) Use this inequality to bound the condition number of the autocorrelation matrix for the moving average process

$$x(n) = w(n) + \alpha w(n-1)$$

where $w(n)$ is unit variance white noise.

(b) Repeat part (a) for the autoregressive process

$$x(n) = \alpha x(n-1) + w(n)$$

where $|\alpha| < 1$ and $w(n)$ is unit variance white noise.

(c) What does this bound imply about the performance of an adaptive filter when the input to the filter is a lowpass process with a power spectrum of the form

$$P_x(e^{j\omega}) = \begin{cases} 1 & ; \quad |\omega| < \omega_0 \\ 0 & ; \quad \omega_0 \leq |\omega| \leq \pi \end{cases}$$

9.7. Suppose that the input to an FIR LMS adaptive filter is a first-order autoregressive process with an autocorrelation

$$r_x(k) = c\alpha^{|k|}$$

where $c > 0$ and $0 < \alpha < 1$. Suppose that the step size μ is

$$\mu = \frac{1}{5\lambda_{\max}}$$

(a) How does the rate of convergence of the LMS algorithm depend upon the value of α?

(b) What effect does the value of c have on the rate of convergence?

(c) How does the rate of convergence of the LMS algorithm depend upon the desired signal $d(n)$?

9.8. The coefficient update equation for the LMS adaptive filter applies a correction to the weight $\mathbf{w}_n$ at time n as follows

$$\mathbf{w}_{n+1} = \mathbf{w}_n + \mu e(n)\mathbf{x}^*(n)$$

The *Block LMS* algorithm, on the other hand, accumulates these corrections for L samples, beginning at time n, while holding the weight vector $\mathbf{w}_n$ constant. A correction is then applied at the end of the block to form an update at time $n + L$ as follows

$$\mathbf{w}_{n+L} = \mathbf{w}_n + \mu \sum_{l=0}^{L-1} e(n+l)\mathbf{x}^*(n+l)$$

where

$$e(n+l) = d(n+l) - \mathbf{w}_n\mathbf{x}^*(n+l)$$

for $l = 0, 1, \ldots, L-1$.

(a) By evaluating the behavior of $E\{\mathbf{w}_n\}$ as a function of n, determine the conditions on the step size μ that are necessary for the block LMS algorithm to converge in the mean.

(b) Discuss the advantages and/or disadvantages of the block LMS algorithm compared to the standard LMS algorithm.

9.9. The first three autocorrelations of a process $x(n)$ are

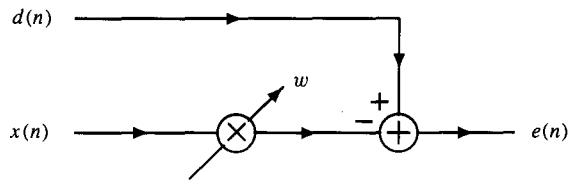
$$r_x(0) = 1, \quad r_x(1) = 0.5, \quad r_x(2) = 0.5$$

Design a two-coefficient LMS adaptive linear predictor for $x(n)$ that has a misadjustment

$$\mathcal{M} = 0.05$$

and find the steady-state mean-square error.

9.10. Consider the single-weight adaptive filter shown in the figure below

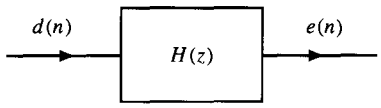


(a) Write down the LMS algorithm for updating the weight w.

(b) Suppose that $x(n)$ is a constant:

$$x(n) = \begin{cases} K & ; \quad n \geq 0 \\ 0 & ; \quad \text{otherwise} \end{cases}$$

Find the system function relating $d(n)$ to $e(n)$ using the LMS algorithm, i.e., find $H(z)$ in the figure below.



(c) Determine the range of values for μ for which $H(z)$ is stable.

9.11. The LMS adaptive filter minimizes the instantaneous squared error

$$\xi(n) = |e(n)|^2$$

Consider the modified functional

$$\xi'(n) = |e(n)|^2 + \beta \mathbf{w}_n^H \mathbf{w}_n$$

where $\beta > 0$.

(a) Derive the LMS coefficient update equation for $\mathbf{w}_n$ that minimizes $\xi'(n)$.

(b) Determine the condition on the step size μ that will ensure that $\mathbf{w}_n$ converges in the mean.

(c) If μ is small enough so that $\mathbf{w}_n$ converges in the mean, what does $\mathbf{w}_n$ converge to?

9.12. Show that the normalized LMS algorithm is equivalent to using the update equation

$$\mathbf{w}_{n+1} = \mathbf{w}_n + \mu e'(n)\mathbf{x}(n)$$

where $e'(n)$ is the error at time n that is based on the new filter coefficients $\mathbf{w}_{n+1}$,

$$e'(n) = d(n) + \mathbf{w}_{n+1}\mathbf{x}(n)$$

Discuss the relationship between μ and the parameter ϵ in the normalized LMS algorithm.

9.13. A process $x(n)$ is formed by passing white noise $w(n)$ through a filter that has a system function

$$H(z) = \frac{1}{1 - 0.08z^{-1} - 0.9z^{-2}}$$

The variance of the white noise is $\sigma_w^2 = (0.19)(0.18)$. The LMS algorithm with two coefficients is used to estimate a process $d(n)$ from $x(n)$.

(a) What is the maximum value for the step size, μ, in order for the LMS algorithm to converge in the mean? Hint: Use the inverse Levinson-Durbin recursion to find the autocorrelation sequence of $x(n)$.

(b) What is the time constant for convergence?

(c) What value for the step size would you use to maximize the rate of convergence of the weights?

(d) If the cross-correlation between $x(n)$ and $d(n)$ is zero,

$$E\{d(n)\mathbf{x}^*(n)\} = 0$$

what are the optimum filter coefficients $\mathbf{w} = [w(0), w(1)]^T$?

9.14. Griffiths developed an algorithm for adaptive beamforming referred to as the p-vector algorithm that eliminates the need for a reference signal $d(n)$ [17]. This algorithm may be derived as follows. Recall that the filter coefficient update equation for the LMS algorithm is

$$\mathbf{w}_{n+1} = \mathbf{w}_n + \mu e(n)\mathbf{x}^*(n) = \mathbf{w}_n + \mu d(n)\mathbf{x}^*(n) - \mu\left[\mathbf{w}_n^T\mathbf{x}(n)\right]\mathbf{x}^*(n)$$

Note that $d(n)$ is not explicitly needed in this update equation. Instead, what is required is the product $d(n)\mathbf{x}^*(n)$. Therefore, if $d(n)\mathbf{x}^*(n)$ is replaced with its expected value, $\mathbf{r}_{dx} = E\{d(n)\mathbf{x}^*(n)\}$, or by an estimate of this ensemble average, then we have an update equation that does not require knowledge of $d(n)$,

$$\mathbf{w}_{n+1} = \mathbf{w}_n + \mu\mathbf{r}_{dx} - \mu\left[\mathbf{w}_n^T\mathbf{x}(n)\right]\mathbf{x}^*(n)$$

This is the p-vector algorithm proposed by Griffiths.[9]

[9]The name p-vector algorithm comes from the fact that the vector $\mathbf{p}$ was used to denote the cross-correlation between $d(n)$ and $x(n)$.

(a) What constraints must be placed on the step size μ in order for $\mathbf{w}_n$ to converge in the mean?

(b) Develop a "leaky" p-vector algorithm and determine the range of values for μ for convergence in the mean. Assuming that μ is selected so that $\mathbf{w}_n$ converges in the mean, find $\lim_{n\to\infty} E\{\mathbf{w}_n\}$.

9.15. An FIR filter with system function $H(z)$ may be implemented using a frequency sampling structure as follows

$$H(z) = \frac{1 - z^{-M}}{M} \sum_{k=0}^{M-1} \frac{H(k)}{1 - e^{j2\pi k/M} z^{-1}} = H_1(z) H_2(z)$$

where $H_1(z)$ is a comb filter that has M zeroes equally spaced around the unit circle and $H_2(z)$ is a filterbank of resonators where the coefficients $H(k)$ are the DFT coefficients of $h(n)$, i.e.,

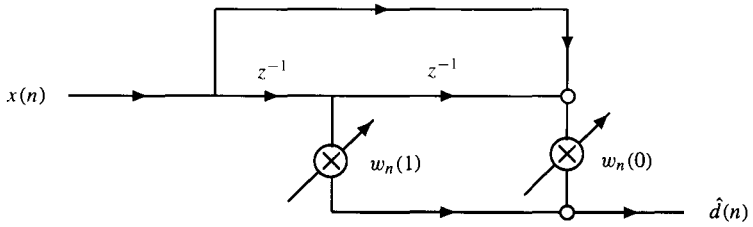
$$H(k) = \sum_{n=0}^{M-1} h(n) e^{-j2\pi kn/M}$$

Suppose that this structure is implemented as an adaptive filter using the LMS algorithm to adjust the filter (DFT) coefficients, $H(k)$. Derive the time-update equation for these coefficients and sketch the adaptive filter structure.

9.16. In many signal processing applications, it is important for a filter to have *linear phase*. This is particularly true in speech and image processing applications, where phase distortion produced by a filter may severely degrade the signal. Therefore, suppose that we would like to design an adaptive linear phase filter whose weights at time n satisfy the following symmetry constraint

$$w_n(k) = w_n(p - k) \quad ; \quad k = 0, 1, \ldots, p$$

For example, consider the two-coefficient linear phase adaptive filter shown in the figure below.



Note that this filter may be viewed as an FIR adaptive filter that is constrained to have the first coefficient equal to the third. As before, define the weight vector, $\mathbf{w}_n$, by

$$\mathbf{w}_n = [w_n(0), w_n(1)]^T$$

and the error sequence, $e(n)$, by

$$e(n) = d(n) - \hat{d}(n)$$

and assume that

$$r_x(0) = 1, \quad r_x(1) = 0.5, \quad r_x(2) = 0.1$$

(a) Derive the normal equations for the filter that minimizes the mean-square error

$$\xi = E\{[d(n) - \hat{d}(n)]^2\}$$

(b) Derive the LMS filter coefficient update equations for this constrained transversal filter.

(c) What values for the step size μ may be used if the weights are to converge in the *mean-square* sense?

(d) If we drop the linear phase constraint and use a three-coefficient LMS adaptive filter with $\mathbf{w} = [w(0), w(1), w(2)]^T$, what values of the step size μ may be used if the adaptive filter is to converge in the mean-square sense?

9.17. In some applications it is known that a given complex-valued process has a constant envelope, e.g. phase modulated signals. The *Constant Modulus Algorithm* (CMA) is an adaptive filtering technique that adjusts the filter coefficients in order to minimize the envelop variation. Given a complex signal $x(n)$ and a set of complex weights $w_n(k)$ at time n, the output of the adaptive filter is

$$y(n) = \mathbf{w}_n^H \mathbf{x}(n)$$

With the constant modulus algorithm, the weights are to be found that minimize the error

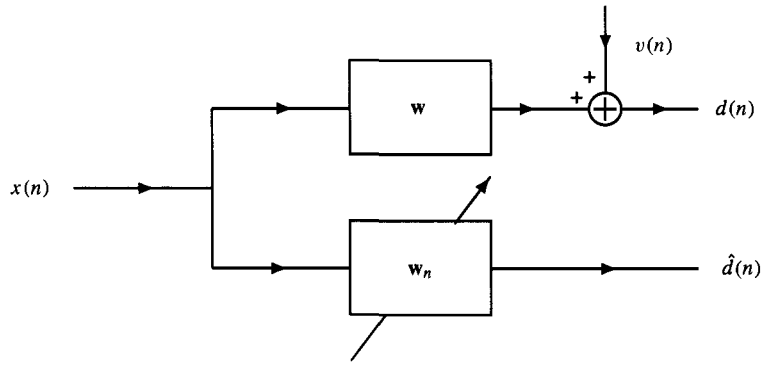
$$\xi(n) = \tfrac{1}{4}E\{(|y(n)|^2 - 1)^2\}$$

which is a non-negative measure of the average amount that the envelope of the filter output $y(n)$ deviates from a nominal level (unity in this case). Using Widrow's approach of estimating ensemble averages with one point sample averages, derive the CMA algorithm, which is an LMS version of a steepest descent algorithm to minimize the error $\xi(n)$ defined above.

9.18. The output $d(n)$ of an unknown system is given by

$$d(n) = \sum_{k=0}^{p} w(k)x(n-k) + v(n) = \mathbf{w}^T\mathbf{x}(n) + v(n)$$

where $w(k)$ are the unknown system parameters, $x(n)$ is the system input, and $v(n)$ is zero mean white noise with a variance of σ_v^2. The block diagram below shows an adaptive filter that is used to model the unknown system.



Assume that $x(n)$ is real-valued and suppose that we would like to find the weight vector $\mathbf{w}_n$ that minimizes the error

$$\xi(k) = E\{e^{2k}(n)\}$$

for some positive integer k where $e(n) = d(n) - \hat{d}(n)$.

(a) As in the LMS algorithm, use the instantaneous gradient vector and derive an LMS update equation for $\mathbf{w}_n$.

(b) Assuming that $v(n)$ is independent of $\mathbf{x}(n)$, and that the weight-error vector

$$\mathbf{c}_n = \hat{\mathbf{w}}_n - \mathbf{w}$$

is close to zero, and that $v(n)$ is independent of $\mathbf{x}(n)$, show that

$$E\{\mathbf{c}_{n+1}\} = \left[\mathbf{I} - \mu k(2k-1)E\{v^{2k-2}(n)\}\mathbf{R}_x\right] E\{\mathbf{c}_n\}$$

where $\mathbf{R}_x$ is the autocorrelation matrix of $x(n)$.

(c) Show that the modified LMS algorithm derived in part (a) will converge in the mean if the step size μ satisfies the condition

$$0 < \mu < \frac{1}{k(2k-1)E\{v^{2(k-1)}(n)\}\lambda_{\max}}$$

where $\lambda_{\max}$ is the largest eigenvalue of the matrix $\mathbf{R}_x$.

(d) For $k = 1$, show that the results derived in parts (a), (b), and (c) reduce to those in the conventional LMS algorithm.

9.19. Adaptive filters are commonly used for linear prediction. Although harmonic signals such as sinusoids are perfectly predictable, measurement noise will degrade the performance of the predictor and add a bias to the coefficients, $\mathbf{w}$. For example, suppose that we want to design an adaptive linear predictor for a real-valued process $x(n)$ using the noisy measurements

$$y(n) = x(n) + v(n)$$

where $v(n)$ is zero mean white noise that is uncorrelated with $x(n)$. Assume that the variance of $v(n)$ is σ_v^2.

(a) Using the LMS algorithm

$$\mathbf{w}_{n+1} = \mathbf{w}_n + \mu e(n)\mathbf{y}(n)$$

find the range of values for μ for which the LMS algorithm converges in the mean and find

$$\lim_{n\to\infty} E\{\mathbf{w}_n\}$$

(b) The γ-LMS algorithm has been proposed as an adaptive filtering algorithm to combat the effect of measurement noise. Using the noisy observations, $y(n)$, this algorithm is

$$\mathbf{w}_{n+1} = \gamma\mathbf{w}_n + \mu e(n)\mathbf{y}(n)$$

where γ is a constant. Explain how the γ-LMS algorithm can be used to remove the bias in the steady-state solution of the LMS algorithm. Specifically, how would you select values for μ and γ?

9.20. In some applications, it may be necessary to delay the update of the filter coefficients for a short period of time. For example, in decision-directed feedback equalization, if a sophisticated algorithm such a Viterbi decoder is used to improve the decisions, then the desired signal and thus the error is not available until a number of samples later. Therefore, assume that $x(n)$ is real-valued, and consider the delayed LMS algorithm that has a filter coefficient update equation given by

$$\mathbf{w}_{n+1} = \mathbf{w}_n + \mu e(n - n_0)\mathbf{x}(n - n_0)$$

where

$$e(n-n_0) = d(n-n_0) - y(n-n_0)$$

Note that if the delay, n_0, is equal to zero then we have the conventional LMS algorithm.

(a) For $n_0 = 1$, determine the values of μ for which the delayed LMS algorithm converges in the mean.

(b) If $\lambda_k = 1$, for $k = 1, \ldots, p$ and if the step size $\mu = 0.1$, find the time constant, τ_0 for the LMS adaptive filter ($n_0 = 0$) and the time constant τ_1 for the delayed LMS adaptive filter with $n_0 = 1$.

9.21. In recent years, there has been an increasing interest in nonlinear digital filters. This interest has included the design of adaptive nonlinear filters. Volterra systems are an important class of nonlinear filters. Assuming that $x(n)$ is real-valued, a second-order Volterra digital filter has the form

$$y(n) = \sum_{k=0}^{K} a(k)x(n-k) + \sum_{k_1=0}^{K_1} \sum_{k_2=k_1}^{K_2} b(k_1,k_2)x(n-k_1)x(n-k_2)$$

Note that the output, $y(n)$, is formed from a linear combination of first-order (linear) terms $x(n-k)$, and a linear combination of second-order (nonlinear) terms $x(n-k_1)x(n-k_2)$. As a specific example, consider the following second-order digital Volterra filter with time-varying coefficients,

$$y(n) = a_n(0)x(n) + a_n(1)x(n-1) + b_n(0)x^2(n) + b_n(1)x(n)x(n-1)$$

Let $\boldsymbol{\Theta}_n$ be the coefficient vector

$$\boldsymbol{\Theta}_n = \left[a_n(0),\ a_n(1),\ b_n(0), b_n(1)\right]^T$$

and let $\mathbf{x}(n)$ be the data vector

$$\mathbf{x}(n) = \left[x(n),\ x(n-1),\ x^2(n),\ x(n)x(n-1)\right]^T$$

(a) Using the LMS update equation

$$\boldsymbol{\Theta}_{n+1} = \boldsymbol{\Theta}_n - \frac{1}{2}\mu\nabla e^2(n)$$

where $e(n) = d(n) - y(n)$, derive the coefficient update equations for $a_n(0)$, $a_n(1)$, $b_n(0)$, and $b_n(1)$.

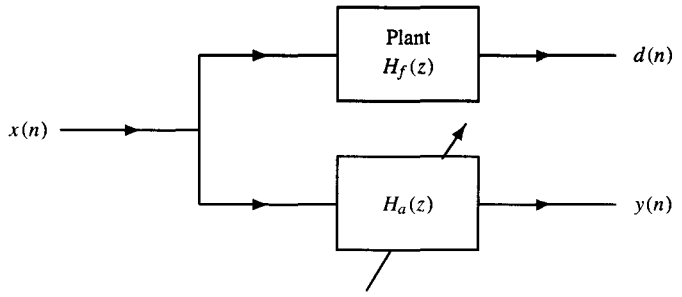
(b) What condition must be placed on μ in order for the coefficient vector $\boldsymbol{\Theta}_n$ to converge in the mean?

(c) Describe what happens if the third-order statistics of $x(n)$ are zero, i.e.,

$$\begin{aligned} E\{x^3(n)\} &= 0 \\ E\{x^2(n)x(n-1)\} &= 0 \\ E\{x(n)x^2(n-1)\} &= 0 \end{aligned}$$

Discuss how you might improve the performance of the adaptive Volterra filter by having two step size parameters, μ_1 and μ_2, one for the linear terms and one for the nonlinear terms, and discuss how these parameters must be restricted in order for the filter to converge in the mean.

9.22. Consider the system identification problem shown in the figure below.



The plant has a rational system function of the form

$$H_f(z) = \frac{0.05 - 0.4z^{-1}}{1 - 1.1314z^{-1} + 0.25z^{-2}}$$

and the adaptive filter that is used to model $H_f(z)$ has two free parameters, a and b, as follows

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

The input, $x(n)$, to both systems is unit variance white noise, and the goal is to find the values of a and b that minimize the mean-square error, $\xi = E\{|e(n)|^2\}$, where $e(n) = d(n) - y(n)$. This mean-square error is a bimodal function of a and b, having a global minimum at $(b, a) = (-0.311, 0.906)$ and a local minimum at $(b, a) = (0.114, -0.519)$.

(a) Write down the equations for the simplified IIR LMS adaptive filter.

(b) Repeat part (a) using the filtered signal approach.

(c) A simplification to the filtered signal approach that has been proposed by Feintuch [15] is to ignore the feedback terms in the equation for the gradient estimates $\psi_k^a(n)$ and $\psi_k^b(n)$. For real-valued signals, this simplification is

$$\psi_k^a(n) = y(n-k) + \sum_{l=1}^{p} a_n(l)\psi_k^a(n-l) \approx y(n-k)$$

$$\psi_k^b(n) = x(n-k) + \sum_{l=1}^{p} a_n(l)\psi_k^b(n-l) \approx x(n-k)$$

Although more efficient than the filtered signal approach, this algorithm may converge to a false minimum, even when the simplified IIR LMS algorithm and the filtered signal approach converge to a minimum. Write down the adaptive filtering equations for $H_a(z)$ using Feintuch's algorithm.

(d) In order for the filter coefficients a_n and b_n in $H_a(z)$ to converge in the mean using Feintuch's algorithm, it is necessary that

$$\lim_{n\to\infty} E\{e(n)x(n)\} = 0$$

and

$$\lim_{n\to\infty} E\{e(n)y(n-1)\} = 0$$

Find the values of $E\{e(n)x(n)\}$ and $E\{e(n)y(n-1)\}$ at the global minimum of ξ. What does this imply about Feintuch's algorithm?

(e) Find the stationary point of the Feintuch adaptive filter, i.e., the value or values of a and b for which $E\{e(n)x(n)\} = 0$ and $E\{e(n)y(n-1)\} = 0$.

9.23. The Hyperstable Adaptive Recursive Filter (HARF) is an IIR adaptive filtering algorithm with known convergence properties. Due to its computational complexity, a simplified version of the HARF algorithm, known as SHARF, is often used. Although the convergence properties of HARF are not preserved in the SHARF algorithm, both algorithms are similar when the filters are adapting slowly (small μ). For real-valued signals, the coefficient update equations for the SHARF algorithm are

$$a_{n+1}(k) = a_n(k) + \mu_a y(n-k)v(n)$$
$$b_{n+1}(k) = b_n(k) + \mu_b x(n-k)v(n)$$

where

$$v(n) = e(n) + \sum_{k=1}^{K} c(k)e(n-k)$$

is the error signal that has been filtered with an FIR filter $C(z)$. Suppose that the coefficients of a second-order adaptive filter

$$H(z) = \frac{b(0) + b(1)z^{-1}}{1 + a(1)z^{-1}}$$

are updated using the SHARF algorithm with

$$C(z) = 1 + c(1)z^{-1} + c(2)z^{-2}$$

(a) Write down the SHARF adaptive filter equations for $H(z)$.

(b) If the SHARF algorithm converges in the mean, then $E\{v(n)\mathbf{x}(n)\}$ converges to zero where

$$\mathbf{x}(n) = [x(n),\ x(n-1),\ y(n-1)]^T$$

What does this imply about the relationship between the filter coefficients $c(1)$ and $c(2)$ and $E\{e(n)\mathbf{x}(n)\}$?

(c) What does the SHARF algorithm correspond to when $C(z) = 1$?

9.24. Modify the RLS algorithm so that the coefficients $w(k)$ satisfy the linear phase constraint, $w(k) = w(p-k)$. For example, with a five-coefficient filter, the coefficient vector $\mathbf{w}_n$ is of the form

$$\mathbf{w}_n = [w_n(0), w_n(1), w_n(2), w_n(1), w_n(0)]^T$$

9.25. There are many different ways that one may compare the performance of adaptive filtering algorithms. Suppose that we are interested in adaptive linear prediction and our measure of performance is the number of arithmetic operations required for the adaptive filter to converge. Let the time constant τ be used as the convergence time of the LMS algorithm. For the RLS algorithm, it is often stated that the rate of convergence is an order of magnitude faster than the LMS algorithm. Therefore, assume that the time constant for the RLS algorithm is one tenth that of the LMS algorithm.

(a) If the eigenvalues of the $p \times p$ autocorrelation matrix for $x(n)$ are

$$\lambda_1 = 1.0 \quad \text{and} \quad \lambda_2 = \cdots = \lambda_p = 0.01$$

and if we use a step size $\mu = 0.1$ for the LMS algorithm, for what order filter, p, are the RLS and LMS adaptive filters equal in terms of their computational requirements to reach convergence?

(b) For high order filters, $p >> 1$, the computational requirements of the RLS filter become large, and the LMS algorithm becomes an attractive alternative. For what reasons might you prefer to use the RLS algorithm in spite of its increased computational cost?

9.26. Let $\alpha(n)$ be the a priori error and $e(n)$ the a posteriori error in the RLS algorithm, and let

$$\mu(n) = \frac{1}{1 + \mathbf{x}^T(n)\mathbf{R}_x^{-1}(n-1)\mathbf{x}^*(n)}$$

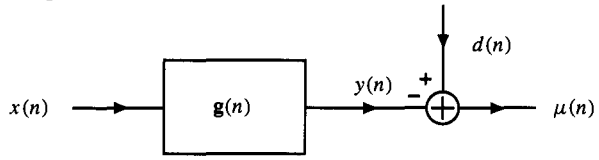
be the scalar that is used in the calculation of the gain vector $\mathbf{g}(n)$.

(a) Show that $e(n)$ may be written in terms of $\alpha(n)$ and $\mu(n)$ by finding an explicit relation between $e(n)$, $\alpha(n)$, and $\mu(n)$. Hint: Begin with the RLS update equation for $\mathbf{w}_n$ and form the product $\mathbf{x}^T(n)\mathbf{w}_n$.

(b) Let $\mathbf{g}(n) = \mu(n)\mathbf{R}_x^{-1}(n-1)\mathbf{x}^*(n)$ be the gain vector in the RLS algorithm. Consider the time-varying filter that has coefficients $\mathbf{g}(n)$ and an input, $x(n)$, that is the same as that used in the RLS algorithm to compute the gain $\mathbf{g}(n)$, i.e.,

$$y(n) = \sum_{k=0}^{p-1} g_n(k)x(n-k)$$

For what signal, $d(n)$, will the difference between $d(n)$ and $y(n)$ be equal $\mu(n)$ as illustrated in the figure below?



9.27. Suppose that the least-squares error used in the RLS algorithm is modified as follows

$$\mathcal{E}_a(n) = \sum_{i=0}^{n} \lambda^{n-i}|e(n)|^2 + \lambda^n \mathbf{w}_n^T \mathbf{w}_n$$

where $\mathbf{w}_n$ is the vector of filter coefficients at time n for a pth-order FIR adaptive filter and

$$e(n) = d(n) - \mathbf{w}_n^T \mathbf{x}(n)$$

Derive the equations for the optimal least squares filter $\mathbf{w}_n$ that minimizes $\mathcal{E}_a(n)$ for each n.

Computer Exercises

C9.1. In this exercise we consider the problem of adaptive linear prediction using the LMS algorithm. Let $x(n)$ be a second-order autoregressive process that is generated by the difference equation

$$x(n) + a(1)x(n-1) + a(2)x(n-2) = v(n)$$

where $v(n)$ is zero mean white noise with a variance σ_v^2. This process is the input to a two-coefficient LMS adaptive predictor.

(a) If $a(1) = -0.1$, $a(2) = -0.8$, and $\sigma_v^2 = 0.25$, for what values of the step size, μ, will the LMS algorithm converge in the mean? For what values of μ will the LMS algorithm converge in the mean-square?

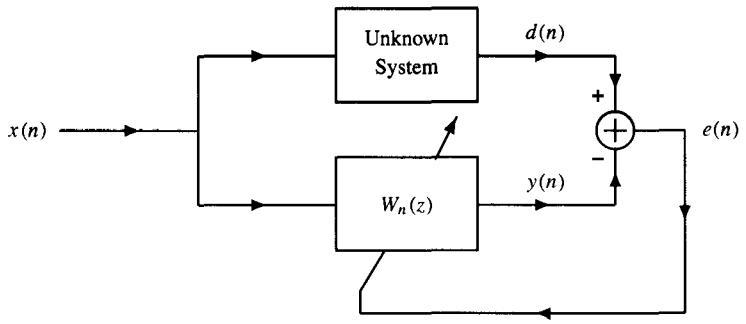
(b) Implement an LMS adaptive predictor using $N = 500$ samples of $x(n)$ and make a plot of the squared prediction error, $e^2(n)$, versus n using step-sizes of $\mu = 0.05$ and $\mu = 0.01$. As described in Example 9.2.2, repeat this experiment for 100 different realizations of $x(n)$, and plot the learning curve by averaging the plots of $e^2(n)$ versus n. Estimate the corresponding values of the misadjustment by time-averaging over the final iterations of the ensemble-averaged learning curves. Compare these estimated values with theory.

(c) Estimate the steady-state values of the adaptive filter coefficients for step-sizes of $\mu = 0.05$ and $\mu = 0.01$. Note that you may do this by averaging the steady-state values of the coefficients (obtained at the final iteration of the LMS algorithm) over 100 independent trials of the experiment. Compare these estimated values with theory.

(d) Repeat the experiments in parts (b) and (c) with $a(1) = -0.1$, $a(2) = 0.8$, and $\sigma_v^2 = 0.25$.

(e) Repeat the above exercises using the sign-error, sign-data, and sign-sign algorithms.

C9.2. One of the many uses of adaptive filters is for system identification as shown in the figure below. In this configuration, the same input is applied to an adaptive filter and to an unknown system, and the coefficients of the adaptive filter are adjusted until the difference between the outputs of the two systems is as small as possible. Adaptive system identification may be used to model a system with slowly-varying parameters, provided both the input and output signals are available.



Let the unknown system that is to be identified be an FIR filter with unit sample response

$$g(n) = \delta(n) + 1.8\delta(n-1) + 0.81\delta(n-2)$$

With an input $x(n)$ consisting of 1000 samples of unit variance white Gaussian noise, create the reference signal $d(n)$.

(a) Determine the range of values for the step size μ in the LMS algorithm for convergence in the mean.

(b) Implement an adaptive filter of order $p = 4$ using the LMS algorithm. Set the initial weight vector equal to zero, and use a step size of $\mu = 0.1\mu_{\max}$, where $\mu_{\max}$ is the largest step size allowed for convergence in the mean. Let the adaptive filter adapt and record the final set of coefficients.

(c) Repeat part (b) using the normalized LMS algorithm with $\beta = 0.1$, and compare your results.

(d) As described in Example 9.2.2, make a plot of the learning curve by repeating the experiment described in part (b) for 100 different realizations of $d(n)$, and plotting the average of the plots of $e^2(n)$ versus n. How many iterations are necessary for the mean-square error to fall to 10% of its peak value? Calculate the theoretical value for the excess mean-square error and compare it to what you observe in your plot of the learning curve.

(e) Repeat parts (b) and (d) for $\mu = 0.01\mu_{\max}$ and $\mu = 0.2\mu_{\max}$.

(f) Repeat parts (b) and (d) for $p = 3$ and $p = 2$.

C9.3. In this exercise, we consider the system identification experiment described in Exercise C9.2, but this time, we look at what happens if the reference signal is corrupted by noise. Instead of $d(n)$, suppose that we observe

$$\tilde{d}(n) = d(n) + \gamma v(n)$$

where $v(n)$ is unit variance white Gaussian noise.

(a) With an adaptive filter of order $p = 4$ and $\gamma = 0.1$, use the normalized LMS algorithm to model the system defined in Exercise C9.2. Use a step size of $\beta = 0.1$ and set the initial weight vector equal to zero. Let the filter adapt and record the final set of coefficients. Repeat your experiment using $p = 5$ and discuss your results.

(b) Repeat part (a) with $\gamma = 1$ and comment on how the accuracy of the model varies with the variance of the noise. Does the accuracy of your model depend on the step size?

C9.4. In the previous two exercises, we considered the problem of using an FIR adaptive filter to model an unknown FIR system. In this exercise, we consider system identification for an unknown IIR system. The arrangement for the filter is the same as that described in Exercise C9.2, except that the unknown system is an IIR filter with a system function

$$G(z) = \frac{1 + 0.5z^{-1}}{1 - 0.9z^{-1}}$$

Generate a 500-point sequence of unit variance white Gaussian noise, and use this process as the input to the unknown system and the adaptive filter.

(a) Use the normalized LMS algorithm with $p = 4$ and $\beta = 0.1$ to model $G(z)$. Record the final values of the filter coefficients and make a plot of the learning curve as described in Exercise C9.2. How many iterations are required for the filter to converge?

(b) Repeat part (a) for different values of p. What seams to be a "reasonable" value to use?

(c) Repeat parts (a) and (b) when the denominator of $G(z)$ is replaced with $A(z) = 1 - 0.2z^{-1}$. Explain your observations.

(d) Add white noise $v(n)$ to the output of the unknown system. Repeat part (a) with noise variances of $\sigma_v^2 = 0.01, 0.1$, and 1.0.

(e) Write a MATLAB m-file to implement the filtered signal approach for IIR adaptive filtering. With an adaptive recursive filter of the form

$$y(n) = a_n(1)y(n-1) + b_n(0)x(n) + b_n(1)x(n-1)$$

determine an appropriate value for the step size μ, and use your m-file to model the system $G(z)$. Discuss your results in terms of the convergence of the coefficients and the accuracy of the model. Describe what happens if the model order is increased, e.g., suppose you were to use two poles and two zeros in the adaptive filter.

(f) Add white noise $v(n)$ to the output of the unknown system. Repeat part (e) with noise variances of $\sigma_v^2 = 0.01, 0.1$, and 1.0.

C9.5. Modify the m-file for the normalized LMS algorithm, `nlms.m`, to take advantage of the computational simplification given in Eq. (9.51).

C9.6. In this problem we will compare LMS and RLS for adaptive linear prediction. As in Example 9.2.1, let $x(n)$ be a process that is generated according to the difference equation

$$x(n) = 1.2728x(n-1) - 0.81x(n-2) + v(n)$$

where $v(n)$ is unit variance white Gaussian noise. The adaptive linear predictor will be of the form

$$\hat{x}(n) = w_n(1)x(n-1) + w_n(2)x(n-2)$$

(a) Implement an RLS adaptive predictor with $\lambda = 1$ (growing window RLS) and plot $w_n(k)$ versus n for $k = 1, 2$. Compare the convergence of the coefficients $w_n(k)$ to those that are obtained using the LMS algorithm for several different values of the step size μ.

(b) Make a plot of the learning curve for RLS and compare it to the LMS learning curve (See Example 9.2.2 on how to plot learning curves). Comment on the excess mean-square error for RLS and discuss how it compares to that for LMS.

(c) Repeat part (b) for exponential weighting factors of $\lambda = 0.99, 0.95, 0.92, 0.90$ and discuss the trade-offs involved in the choice of λ.

(d) Modify the m-file for the RLS algorithm to implement a sliding window RLS algorithm.

(e) Let $x(n)$ be generated by the time-varying difference equation

$$x(n) = a_n(1)x(n-1) - 0.81x(n-2) + v(n)$$

where $v(n)$ is unit variance white noise and $a_n(1)$ is a time-varying coefficient given by

$$a_n(1) = \begin{cases} 1.2728 & ; \quad 0 \leq n < 50 \\ 0 & ; \quad 50 \leq n < 100 \\ 1.2728 & ; \quad 100 \leq n \leq 200 \end{cases}$$

Compare the effectiveness of the LMS, growing window RLS, exponentially weighted RLS, and sliding window RLS algorithms for adaptive linear prediction. What approach would you propose to use for linear prediction when the process has step changes in its parameters?

C9.7. Let $x(n)$ be a process that is generated according to the difference equation

$$x(n) = 0.8x(n-1) - 0.9x(n-2) + v(n)$$

where $v(n)$ is unit variance white Gaussian noise. Suppose that we would like to implement an adaptive linear predictor using the LMS algorithm,

$$\hat{x}(n) = w_n(1)x(n-1) + w_n(2)x(n-2)$$

(a) Generate a sequence $x(n)$ of length $N = 500$ using the difference equation above and determine a value for the LMS step size so that $\mathbf{w}_n$ converges in the mean to $\mathbf{w} = [0.8 \ -0.9]^T$ within about 200 iterations.

(b) Implement the LMS adaptive predictor and plot $w_n(k)$ versus n for $k = 1, 2$.

(c) Estimate the mean-square error $\xi(\infty)$ and compare it to the theoretical value.

(d) Write a MATLAB program or modify the m-file `lms.m` to implement the p-vector algorithm to estimate a process $d(n)$ from $x(n)$ (see Prob. 9.14),

$$\mathbf{w}_{n+1} = \mathbf{w}_n + \mu\mathbf{r}_{dx} - \mu[\mathbf{w}_n^T\mathbf{x}(n)]\mathbf{x}(n)$$

where $\mathbf{r}_{dx} = E\{d(n)\mathbf{x}(n)\}$.

(e) Find the vector, $\mathbf{r}_{dx} = E\{d(n)\mathbf{x}(n)\}$ for a one-step linear predictor, and implement the p-vector algorithm for $x(n)$. Plot $w_n(k)$ versus n and compare your results to the LMS adaptive linear predictor. Does the p-vector algorithm converge? If so, to what? What constraints are necessary on the step size for the p-vector algorithm to be well-behaved?

(f) Investigate the sensitivity of the p-vector algorithm to errors in the vector $\mathbf{r}_{dx}$ by making small changes in the values of $\mathbf{r}_{dx}$ used in part (e).

C9.8. The LMS algorithm is very popular due to its inherent simplicity. However, it suffers from relatively slow convergence properties, and its performance is severely affected when there is a large eigenvalue spread in the autocorrelation matrix of the input signal.

A number of different algorithms have been proposed to improve the convergence properties of the LMS algorithm that incorporate a variable step size parameter, $\mu(n)$. One approach that has been proposed is to use a monotonically decreasing step size of the form

$$\mu(n) = \frac{1}{c+n}$$

where c is some constant. Thus, the variable step size adaptive filter becomes

$$\mathbf{w}_{n+1} = \mathbf{w}_n + \mu(n)e(n)\mathbf{x}(n)$$

(a) Evaluate the effectiveness of this algorithm for linear prediction. What value should you choose for c? What limitations do you see that this algorithm might have?

(b) Another possible form for $\mu(n)$ would be to use

$$\mu(n) = \frac{1}{c_1 + c_2 n}$$

This would allow for better control in how fast the step size decreases to zero. Repeat part (a) for this variable step size adaptive filter.

C9.9. In this problem we look at the problem of adaptive noise cancellation without a reference (see Fig. 9.2*b*). Let

$$x(n) = d(n) + v(n)$$

where $d(n)$ is a unit amplitude sinusoid of frequency $\omega_0 = 0.01\pi$. Suppose that $v(n)$ is a

moving average process that is generated as follows

$$v(n) = g(n) + 0.5g(n-2)$$

where $g(n)$ is unit variance white noise.

(a) What is the minimum value for the delay k_0 that may be used in the adaptive noise canceller?

(b) Write a MATLAB program for adaptive noise cancellation using the normalized LMS algorithm with $x(n-k_0)$ as the reference signal.

(c) Generate 1000 samples of $v(n)$ and $x(n)$ and use your MATLAB program to estimate $d(n)$ for filter orders $p = 5, 10, 15, 20$. Use values for k_0 that range from the minimum value determined in part (a) to $k_0 = 25$. What dependence on k_0, if any, do you observe? Explain. What is the effect of the filter order p? Given that the computational costs go up with p, what order do you think would be the best to use?

(d) Find the coefficients of an FIR Wiener filter to estimate and compare them with the steady-state values of the coefficients of the adaptive noise canceller.

APPENDIX MATLAB PROGRAMS

A.1 INTRODUCTION

One of the most effective ways for a student to gain a deep appreciation and understanding of signal processing is to process signals. There is a great deal of information to be gained by experimenting with algorithms, testing them on real signals and unusual sequences, developing "new approaches," and discovering at what point the theory begins to break down in practice. MATLAB provides an extremely powerful and versatile environment for performing signal processing experiments. As a result, we have included over thirty m-files in this book that cover everything from signal modeling and the Levinson recursion to spectrum estimation and adaptive filtering. In this appendix, we provide a description of each of these m-files, define the format of the inputs and outputs, give pointers to places where the user may encounter problems, and provide simple illustrative examples. Although it is assumed that the reader is familiar with MATLAB, the first-time user should be able to use the m-files in this book after reading this appendix. Anyone without MATLAB experience who wishes to go beyond what is presented here and begin writing their own m-files should consult the MATLAB Users Guide. The reader should keep in mind, however, that MATLAB has an online help facility that is accessible by typing `help` at the MATLAB prompt for general help, or by typing `help filename` for specific help on the m-file `filename`.

None of the m-files described in this appendix are particularly long and may be typed-in by hand without too much difficulty. However, all m-files are available by anonymous ftp from

```
ftp.ece.gatech.edu/pub/users/mhayes/stat_dsp
```

The reader may also want to browse the home page for this book on the Web at

```
http://www.ece.gatech.edu/users/mhayes/stat_dsp
```

for additional information such as new problems and m-files, errata and clarifications, and reader comments and feedback.

A.2 GENERAL INFORMATION

The m-files described in this appendix may be used to process discrete-time signals in a variety of different ways. Each m-file is written to accept either real or complex-valued signals. These signals are input as vectors and may be generated in a number of different ways. For example, a signal may be defined using a mathematical expression such as[1]

```
>>x = 2*cos([0:99]*pi/4);
```

which generates the signal $x(n) = 2\cos(n\pi/4)$ for $n = 0, 1, \ldots, 99$. Alternatively, a signal may be defined by specifying the signal values individually as in the command

```
>>x = [2 1 3 5 1];
```

which sets the first five values of $x(n)$ equal to 2, 1, 3, 5, 1, or with the command

```
>>x = [ones(1,32) zeros(1,32)];
```

which creates a *pulse* consisting of 32 values that are equal to 1 followed by 32 values that are equal to zero.

It is important to keep in mind that indices in MATLAB begin with one. For example, the signal

$$x(n) = \delta(n) + 2\delta(n-1) - 3\delta(n-2)$$

would be input using the command

```
>>x = [1 2 -3];
```

with the value of $x(0)$ being stored in the variable `x(1)`. Therefore, typing `x(1)` at the MATLAB prompt would return the value 1 whereas typing `x(0)` would result in the error message

```
>>x(0)
??? Index exceeds matrix dimensions.
```

Many of the m-files described in this appendix call other m-files. In particular, two m-files that are frequently called are `convm.m` and `covar.m`. The first, `convm.m`, is used to set up a convolution matrix. In other words, given a signal $x(n)$ of length N that is stored in the vector `x`, the command

```
>>X = convm(x,p);
```

produces a $(N + p - 1) \times p$ non-symmetric Toeplitz matrix $\mathbf{X}$ that has the N values of $x(n)$ in the first column (padded with zeros) and $[x(0), 0, \ldots, 0]$ in the first row.[2] For example,

```
>> x=[1 3 2];
>> X=convm(x,4)
```

[1] Note that >> represents the MATLAB command line prompt.

[2] Although common usage of the term Toeplitz matrix assumes that the matrix is square, here we use the term for a rectangular matrix to highlight the structure of the convolution matrix. Note that MATLAB allows the user to generate a rectangular Toeplitz matrix using the command `x=toeplitz(c,r)` where `c` and `r` are the first row and column, respectively, of the Toeplitz matrix, and `c` and `r` need not be the same length.

```
ans =

     1     0     0     0
     3     1     0     0
     2     3     1     0
     0     2     3     1
     0     0     2     3
     0     0     0     2
```

The second m-file, `covar.m`, forms a covariance matrix. Thus, if the sequence $x(n)$ is stored in the vector x, then

```
>> R=covar(x,p)
```

creates a $p \times p$ Hermitian Toeplitz matrix, R, of sample covariances. Listings of both of these m-files are given in Fig. A.1.

Many of the m-files in this book compute the DFT of a signal using the m-file `fft.m`. This m-file is called as follows:

```
>>X = fft(x,n);
```

where x is an input vector that contains the signal values that are to be transformed, and X is the output vector that contains the DFT coefficients. The second argument, n, is an optional integer that may be used to specify the length of the DFT. If `fft.m` is called without a second argument,

```
>>X = fft(x);
```

```
function X = convm(x,p)
%
% This function sets up a convolution matrix
%
N    = length(x)+2*p-2;
x    = x(:);
xpad = [zeros(p-1,1);x;zeros(p-1,1)];
for  i=1:p
     X(:,i)=xpad(p-i+1:N-i+1);
     end;


function R = covar(x,p)
%
% This function sets up a covariance matrix
%
x = x(:);
m = length(x);
x = x - ones(m,1)*(sum(x)/m);
R = convm(x,p+1)'*convm(x,p+1)/(m-1);
end;
```

Figure A.1 MATLAB *programs for generating a convolution matrix,* `convm.m`, *and a covariance matrix,* `covar.m`.

then the length of the DFT is equal to the length of the vector x. All of the m-files in this book that call `fft.m` set the value of n equal to 1024. Therefore, for signals longer than 1024-points, it may be necessary to modify the m-files that make a call to `fft.m`.

A.3 RANDOM PROCESSES

The computer exercises at the end of Chapter 3 require no special m-files and may be completed using only MATLAB functions and a few m-files from the Signal Processing Toolbox. Two of the m-files required in these exercises, `randn` and `rand`, are used to generate Gaussian and uniform white noise processes, respectively.[3] For example,

```
>>noise1=randn(1,100);
```

generates 100 samples of unit variance white Gaussian noise, and

```
>>noise2=2*rand(1,100)-1;
```

generates 100 samples of zero mean white noise that is uniformly distributed over the interval $[-1, 1]$. Once a white noise sequence has been created, it is straightforward to generate an autoregressive, moving average, or autoregressive moving average process. This may be done by filtering the white noise using the m-file `filter`. For example, the sequence of commands

```
>>a=[1, 0.5, 0.8];
>>b=[1, 2, 1];
>>x=filter(b,a,randn(1,100));
```

creates 100 samples of an ARMA(2,2) process that has a power spectrum

$$P_x(e^{j\omega}) = \frac{\left|1 + 2e^{-j\omega} + e^{-2j\omega}\right|^2}{\left|1 + 0.5e^{-j\omega} + 0.8e^{-2j\omega}\right|^2} = \frac{6 + 8\cos\omega + 2\cos 2\omega}{1.89 + 1.8\cos\omega + 1.6\cos 2\omega}$$

A.4 SIGNAL MODELING

In this section, we describe the m-files presented in Chapter 4, which are concerned primarily with signal modeling. These include m-files for the Padé approximation, Prony's method, Shanks' method, iterative prefiltering, the autocorrelation and covariance methods, and Durbin's method. In addition we describe the m-file to design a least squares inverse filter. In all cases, the signals that are used as inputs to these m-files may be either real-valued or complex-valued.

Padé Approximation. Given a signal $x(n)$, the Padé approximation finds the coefficients in the model

$$H(z) = \frac{B_q(z)}{A_p(z)} = \frac{\sum_{k=0}^{q} b_q(k) z^{-k}}{1 + \sum_{k=1}^{p} a_p(k) z^{-k}} \tag{A.1}$$

[3] See the MATLAB Users Manual or type `help randn` or `help rand` to see how to set the seed of the random number generator. This is useful when it is necessary to repeatedly generate the same noise process.

by matching the first $p+q+1$ signal values exactly, i.e., $x(n) = h(n)$ for $n = 0, 1, \ldots, p+q$. The m-file that is used to find the Padé approximation of a signal $x(n)$ is `pade.m`, and this m-file may be called as follows

```
>>[a,b]=pade(x,p,q);
```

There are three required inputs to this m-file. The first is the vector `x` that contains the values of the signal $x(n)$ that is to be modeled. The other two inputs, `p` and `q`, specify the model order, i.e., the order of the polynomials $A_p(z)$ and $B_q(z)$, respectively. The output consists of two vectors, `a` and `b`, that contain the model coefficients $a_p(k)$ and $b_q(k)$, respectively. As indicated in Eq. (A.1), the first coefficient of the vector `a` will always be equal to one.

Once the coefficients of the model have been found, the Padé approximation $\hat{x}(n)$ may be computed by evaluating the unit sample response of $H(z)$ as follows:

```
>>xhat=filter(b,a,[1 zeros(1,length(x)-1)]);
```

Alternatively, the DTFT of the Padé approximation may be plotted with the command

```
>>freqz(b,a)
```

Finally, the average squared error may be computed as follows

```
>>mse=norm(x-xhat);
```

where `xhat` is the Padé approximation that may be found as described above.

Special Notes: The model orders `p` and `q` must be non-negative integers and `p+q` must be less than the length of the vector `x`.

Prony's Method. As with the Padé approximation, Prony's method finds a model for a signal $x(n)$ of the form given in Eq. (A.1). Unlike the Padé approximation, however, the denominator coefficients $a_p(k)$ are found by minimizing the Prony error defined in Eq. (4.29). This requires solving the linear equations given in Eq. (4.35). Once the denominator coefficients have been determined, the numerator coefficients $b_q(k)$ are found using the Padé approach to match the signal exactly for the first $q + 1$ values of $x(n)$.

The m-file for Prony's method is `prony.m`, and the format of the command line is

```
>>[a,b,err]=prony(x,p,q);
```

As with `pade.m`, the inputs to `prony.m` include the vector `x` of signal values that are to be modeled, and the integers `p` and `q` that specify the model order. The outputs are the vectors `a` and `b` that contain the coefficients $a_p(k)$ and $b_q(k)$, respectively, and `err` which is the Prony error $\epsilon_{p,q}$ given in Eq. (4.44). The first coefficient of the vector `a` is always equal to one.

In the derivation of Prony's method, it is assumed that $x(n)$ is known for all $n \geq 0$. Therefore, there is an important practical issue that concerns how to address the problem of only being able to record and process a finite-length observation of $x(n)$. What is done in `prony.m` is to assume that $x(n)$ is zero for all values of n that are greater than the length of the input vector `x`. Therefore, `prony.m` uses the autocorrelation method to find the denominator coefficients, and the Padé approximation to find the numerator coefficients.

When using `prony.m` with `q=0` (an all-pole model) it is important to note that `prony.m` sets $b(0) = x(0)$. For an all-pole model, a better approach, in general, would be to set $b(0) = \sqrt{\epsilon_p}$ (an energy matching constraint) as follows,

```
>>[a,b,err]=prony(x,p,0);
>>b=sqrt(err);
```

Special Notes: The model orders p and q must be non-negative integers and p+q must be less than the length of the vector x.

Shanks' Method. Shanks' method provides an alternative to Prony's method of finding the numerator coefficients. Instead of forcing an exact fit to the data for the first $q+1$ values of n, Shanks' method performs a least squares minimization of the error

$$e'(n) = x(n) - \hat{x}(n)$$

The m-file for Shanks' method is `shanks.m` and the format used to call this m-file is

```
>>[a,b,err]=shanks(x,p,q);
```

where x is the vector that contains the signal that is to be modeled, p is the order of the denominator polynomial, and q is the order of the numerator polynomial. As with the previous two m-files, the output consists of two vectors, a and b, that contain the coefficients $a_p(k)$ and $b_q(k)$, respectively, and the model error `err`.

Special Notes: The model orders p and q must be non-negative integers and p+q must be less than the length of the vector x.

Iterative Prefiltering. Iterative prefiltering, or the method of Steiglitz and McBride, is an iterative algorithm to find a rational model for a signal $x(n)$ that minimizes the least squares error

$$\mathcal{E}_{LS} = \sum_{n=0}^{\infty} |x(n) - h(n)|^2$$

Although there is no general proof of convergence, iterative prefiltering often reaches an acceptable solution after 5 to 10 iterations. The m-file that implements the method of iterative prefiltering is `ipf.m` and may be called as follows

```
>>[a,b,err]=ipf(x,p,q,n,a0);
```

The inputs that are required are the signal vector, x, the number of poles in the model, p, the number of zeros, q, and the number of iterations n. In addition, there is an optional input vector, a0, that is used to initialize the recursion with a given set of denominator coefficients. If this input is left unspecified, then the initial condition is found using Prony's method. The outputs of the m-file are the model coefficients $a_p(k)$ and $b_q(k)$, which are stored in the vectors a and b, respectively, and the squared error, `err`.

The format of the m-file `ipf.m` allows one to continue the iteration after making an initial call to `ipf.m`. For example, typing

```
>>[a,b,err]=ipf(x,p,q,n1);
>>[a,b,err]=ipf(x,p,q,n2,a);
```

performs iterative prefiltering for n1 iterations, and then uses the vector a that is obtained after n1 iterations to initialize `ipf.m` for the next n2 iterations.

Special Notes: The model orders p and q must be non-negative integers and p+q must be less than the length of the vector x. The number of iterations, n, must be a positive integer, typically in the range from 5 to 10. The optional input vector a0 must be a non-zero vector of length p.

Autocorrelation Method. The autocorrelation method is an all-pole modeling technique that finds the all-pole coefficients $a_p(k)$ from the values of $x(n)$ for $n = 0, 1, \ldots, N$ by minimizing the Prony error

$$\mathcal{E}_p = \sum_{n=0}^{\infty} |e(n)|^2$$

where

$$e(n) = x(n) + \sum_{k=1}^{p} a_p(k)x(n-k)$$

Since $x(n)$ is assumed to be known only for $0 \leq n \leq N$, the error $e(n)$ may only be evaluated for $p \leq n \leq N$. Therefore, $\mathcal{E}_p$ cannot be minimized unless some assumptions are made about the values of $x(n)$ outside the interval $[0, N]$. With the autocorrelation method, $x(n)$ is assumed to be equal to zero for $n < 0$ and $n > N$, which is equivalent to applying a rectangular data window to $x(n)$. Although this window biases the solution, it ensures that the all-pole model will be stable. The m-file for the autocorrelation method is `acm.m` and may be called as follows

```
>>[a,err]=acm(x,p);
```

The input `x` is a vector that contains the signal values $x(n)$, and `p` is an integer that specifies the model order (number of poles). The output `a` is the vector of coefficients $a_p(k)$, and `err` is the modeling error, $\epsilon_p = \min\{\mathcal{E}_p\}$. The numerator coefficient, $b(0)$, of the all-pole model is typically selected to satisfy the energy matching constraint,

$$b(0) = \sqrt{\epsilon_p}$$

Another possibility, however, is

$$b(0) = x(0)$$

which forces the first value in the sequence to match the first value in the model.

Special Notes: The model order `p` must be a positive integer that is less than the length of the vector `x`.

Covariance Method. The covariance method is an alternative to the autocorrelation method for finding an all-pole model for a finite-length sequence $x(n)$, $n = 0, 1, \ldots, N$. Instead of assuming that the unknown values of $x(n)$ are equal to zero, the covariance method modifies the error that is to be minimized. The modification simply involves redefining the limits on the sum for $\mathcal{E}_p$ to begin at $n = p$ and end at $n = N$. Since the covariance method does not window the data, the model is generally more accurate than the autocorrelation method. However, the model is not guaranteed to be stable. The m-file for the covariance method is `covm.m` and it is called as follows:

```
>>[a,err]=covm(x,p);
```

As with the m-file `acm.m`, the inputs consist of the data vector `x` and the model order `p`, and the outputs are the model coefficients $a_p(k)$, which are stored in the vector `a`, and the error, which is stored in `err`.

Special Notes: The model order `p` must be a positive integer that is less than the length of the vector `x`.

Durbin's Method. Durbin's method is a procedure for finding a qth-order moving average model for a wide-sense stationary process $x(n)$. Thus, the model for $x(n)$ is of the form

$$x(n) = \sum_{k=0}^{q} b_q(k) w(n-k)$$

where $w(n)$ is unit variance white noise. This algorithm begins by finding a high-order ($p \gg q$) all-pole model for $x(n)$. Then, treating the model coefficients as the data, a qth-order all-pole model is found for these coefficients. The m-file for Durbin's method is `durbin.m`, which is called using a command of the form

```
>>b=durbin(x,p,q);
```

where `x` is the vector that contains the signal values, `p` is the order of the high-order all-pole model for $x(n)$, and `q` is the order of the moving average model. The output is the vector `b` of moving average coefficients.

Special Notes: The model orders `p` and `q` must be positive integers with `p` > `q`. It is generally recommended that the value of `p` be at least four times order of the moving average model, `q`. However, `p` must not exceed the length of the data vector `x`.

Least Squares Inverse Filter. Given a filter $g(n)$, the least squares inverse, $h(n)$, is the filter that produces the best approximation to a unit sample at time $n = n_0$ when convolved with $g(n)$,

$$g(n) * h(n) \approx \delta(n - n_0) \tag{A.2}$$

The m-file to find the least squares inverse filter is `spike.m`, and the format required to use this m-file is

```
>>[h,err]=spike(g,n0,n);
```

The inputs to this m-file include `g`, which is a vector that contains the coefficients of the filter that are used to design the least squares inverse, `n0`, which is the value of the delay n_0, and `n`, which is the order (number of coefficients) of the least squares inverse filter $h(n)$. The output `h` is the vector containing the coefficients of the least squares inverse filter, and `err` is the total squared error in the least squares inverse design.

Special Notes: The order of the inverse filter, `n`, must be a positive integer, and the delay, `n0`, must be a positive integer that is no larger than the length of the convolution $g(n) * h(n)$, i.e.,

```
n0 < length(g) + length(h) - 1
```

where `n = length(h)`.

A.5 LEVINSON RECURSION

In this section, we describe the m-files presented in Chapter 5, which are concerned with the Levinson and Levinson-Durbin recursions. Six of these m-files involve mappings between an autocorrelation sequence, the all-pole filter coefficients, and the reflection coefficients as illustrated in Fig. A.2. The seventh m-file is for the Levinson recursion, and may be used

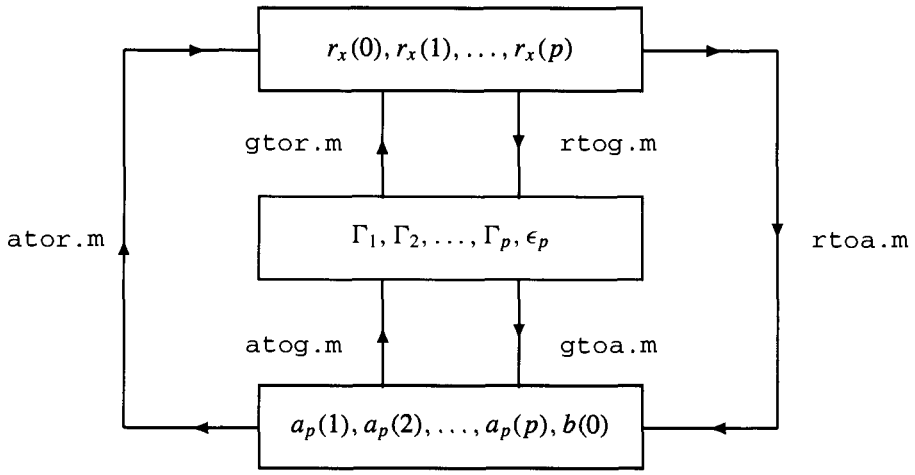


Figure A.2 *The m-files for converting between an autocorrelation sequence, the reflection coefficients, and the all-pole model.*

to find the solution to a general set of Toeplitz equations. Each of these m-files allow the inputs to be either real-valued or complex-valued vectors.

Levinson-Durbin Recursion. The Levinson-Durbin recursion is an efficient algorithm for solving a set of Hermitian Toeplitz equations of the form

$$\mathbf{R}_x \mathbf{a}_p = \epsilon_p \mathbf{u}_1$$

where $\mathbf{R}_x = \text{Toep}\{r_x(0), r_x(1), \ldots, r_x(p)\}$ is a Hermitian Toeplitz matrix and $\mathbf{u}_1$ is a unit vector with a one for the first coefficient. Since the Levinson-Durbin recursion is a mapping from a sequence of autocorrelations $r_x(k)$ to a set of model coefficients $a_p(k)$, the name of the m-file is `rtoa.m` (r-to-a).[4] The format of the command line for this m-file is

```
>>[a,epsilon]=rtoa(r);
```

and requires only one input, `r`, which is a vector that contains the sequence of autocorrelations, $r_x(k)$. The outputs include the vector `a`, the solution to the Toeplitz normal equations, and the scalar `epsilon`, which, in the case of all-pole signal modeling, represents the modeling error. It should be pointed out that `r` may be *any* complex vector, i.e., `r` need not be a valid autocorrelation sequence. However, if `r` is *not* a valid autocorrelation sequence, then $\mathbf{R}_x$ will not be positive semi-definite. In this case, `epsilon` may be negative and will not have any physical meaning in terms of a modeling error.

Although the Levinson-Durbin recursion is a mapping from a sequence of autocorrelations $r_x(k)$ to a set of model coefficients $a_p(k)$ and a set of reflection coefficients Γ_j, the m-file `rtoa.m` only finds the coefficients $a_p(k)$. A companion m-file is listed in Fig. A.3, `rtog.m`, that performs the mapping from $r_x(k)$ to the reflection coefficients Γ_j. The syntax for this m-file is

```
>>[gamma,epsilon]=rtog(r);
```

This m-file first calls `rtoa.m`, which produces the sequence $a_p(k)$, and then calls `atog.m`

[4]The inverse of this m-file is `ator.m` (a-to-r), which is described later.

```
function [gamma,epsilon] = rtog(r)
%
   [a,epsilon]=rtoa(r);
   gamma=atog(a);
   end;
```

Figure A.3 *An m-file for the Levinson-Durbin recursion that finds the set of reflection coefficients* Γ_j *from the autocorrelation sequence* $r_x(k)$.

(described below), which maps the coefficients $a_p(k)$ to the reflection coefficients Γ_j. The reflection coefficients may also be found from the sequence $r_x(k)$ as follows:

```
>>gamma=atog(rtoa(r));
```

Special Notes: None.

Step-up Recursion. The step-up recursion is an algorithm to find the direct-form coefficients $a_p(k)$ that correspond to a sequence of reflection coefficients Γ_j. The m-file for the step-up recursion is `gtoa.m` and may be called as follows:

```
>>a=gtoa(gamma);
```

The input to this m-file, `gamma`, is a vector that contains the reflection coefficients Γ_j and the output, `a`, is the vector of filter coefficients $a_p(k)$ with the leading coefficient equal to one. For example,

```
>>a=gtoa([.1, .2, .3]);
```

produces the vector `a = [1, 0.18, 0.236, 0.3]`.

Recall that the polynomial

$$A_p(z) = 1 + \sum_{k=1}^{p} a_p(k) z^{-k}$$

will be minimum phase (all of the roots inside the unit circle) if and only if the reflection coefficients are less than one in magnitude. Compare, for example, the results that are obtained with the following commands:

```
>>abs(roots(gtoa([.5, .6, .7])))
>>abs(roots(gtoa([.5, .6, 1.2])))
```

Special Notes: None.

Step-down Recursion. The step-down recursion is the inverse of the step-up recursion, producing a sequence of reflection coefficients from a set of filter coefficients. The m-file for the step-down recursion, `atog.m`, is called as follows:

```
>>g=atog(a);
```

where `a` is the vector that contains the filter coefficients $a_p(k)$, and `g` is the vector that contains the reflection coefficients Γ_j. The input `a` may be any vector of real or complex

numbers, provided that the first coefficient is nonzero. Since `atog.m` normalizes the first coefficient of `a` to one, then the following two commands result in the same sequence of reflection coefficients,

```
>>atog([1, .9, .81]);
>>atog([2, 1.8, 1.62]);
```

As discussed in Section 5.2.4, the step-down recursion may be used to check the stability of a digital filter. For example, if $H(z) = B_q(e^{j\omega})/A_p(e^{j\omega})$ and the coefficients of the denominator polynomial $A_p(e^{j\omega})$ are stored in the vector `a`, then the command

```
>>max(abs(atog(a)))
```

will return a value that is less than one if the filter is stable, and a value that is greater than one if it is unstable.

Special Notes: The first coefficient of the vector `a` must be non-zero.

Inverse Levinson-Durbin Recursion. Given an autocorrelation sequence $r_x(k)$, the Levinson-Durbin recursion produces a sequence of reflection coefficients, Γ_j, and a sequence of direct-form filter coefficients $a_p(k)$. The inverse Levinson-Durbin recursion is a mapping that recovers the autocorrelation sequence from either Γ_j or $a_p(k)$. The two m-files that implement these recursions are `gtor.m` and `ator.m`. The first, `gtor.m`, finds the autocorrelation sequence from the reflection coefficients,

```
>>r=gtor(gamma,epsilon);
```

The only required input to this m-file is the vector `gamma` that contains the sequence of reflection coefficients. The second input, `epsilon`, is an optional scalar. With only one input, the command

```
>>r=gtor(gamma);
```

will return a normalized autocorrelation sequence with $r_x(0) = 1$. For example, with the reflection coefficient sequence $\Gamma = [0.2, -0.2, 0.2]^T$ we find

```
>> r=gtor([.2, -.2, .2])

r =

    1.0000   -0.2000    0.2320   -0.2614
```

Note that when `gamma` is a vector of length p, the output `r` will be a vector of length $p+1$. If, on the other hand, a value for `epsilon` is specified, then

```
>>r=gtor(gamma,epsilon);
```

will return an autocorrelation sequence that is scaled so that `gtor` is the inverse of `rtog`, i.e.,

```
>>r=gtor(rtog(r));
```

The second m-file, `ator.m`, finds the autocorrelation sequence from a set of filter coefficients, and may be called as follows:

```
>>r=ator(a,b);
```

The only required input to this m-file is the vector a of coefficients $a_p(k)$. The second input, b, is an optional scalar that corresponds to the value $b(0)$ in the all-pole model. With only one input,

```
>>r=ator(a);
```

returns a normalized autocorrelation sequence $r_x(k)$ with $r_x(0) = 1$. Note that if a is a vector of length p then the output r will also be a vector of length p. If a value for $b(0)$ is specified in the variable b, then

```
>>r=ator(a,b);
```

will return an autocorrelation sequence, properly scaled, so that r is the inverse of `rtog.m`, i.e.,

```
>>r=ator(rtoa(r));
```

Special Notes: None.

General Levinson Recursion. The general Levinson recursion is an efficient algorithm for solving a general set of Hermitian Toeplitz equations of the form

$$\mathbf{R}_p\mathbf{x}_p = \mathbf{b} \tag{A.3}$$

where $\mathbf{R}_p = \text{Toep}\{r_x(0), r_x(1), \ldots, r_x(p)\}$ is a Hermitian Toeplitz matrix and $\mathbf{b}$ is an arbitrary vector. These equations arise in a number of problems including the Padé approximation, Shanks' method, and FIR Wiener filtering. The m-file for the general Levinson recursion is `glev.m` and is called as follows

```
>>x=glev(r,b);
```

There are two required inputs to this m-file. The first is the vector r, which is the first column of the Toeplitz matrix $\mathbf{R}_p$. The second, b, is the vector of coefficients on the right-side of Eq. (A.3). The output is the vector x, which is the solution to the Toeplitz equations.

Special Notes: The vectors r and b must have the same length.

A.6 LATTICE FILTERS

In Chapter 6 we looked at all-pole models that are based on the sequential optimization of the reflection coefficients. These modeling techniques included the forward and backward covariance methods, and the Burg algorithm. In addition, we looked at the modified covariance method, which is a non-sequential version of the Burg algorithm. In this section, we describe the m-files for these methods.

The forward covariance method finds the reflection coefficients of an all-pole model for a signal $x(n)$ by sequentially minimizing the sum of the squares of the forward prediction error. A related modeling technique, the backward covariance method, sequentially minimizes the sum of the squares of the backward prediction error. For any sequence, either the forward covariance method or the backward covariance method will produce an unstable model. The Burg algorithm, however, which sequentially minimizes the sum of the squares of the forward and backward prediction errors, is guaranteed to give a stable model for any

sequence $x(n)$. The modified covariance method, on the other hand, finds the all-pole model that minimizes the sum of the squares of the forward and backward prediction errors. Since this minimization is not performed sequentially, the modified covariance method finds the global minimum. However, the model that is found is not guaranteed to be stable.

The m-files for the lattice filter based signal models all have the same format. Each requires two inputs. The first is a vector `x` that contains the signal values that are to be modeled. The second is the order of the all-pole model, `p`. The outputs of each of these m-files include the vector `gamma`, which contains the reflection coefficients for the all-pole model, and the vector `err`, which contains the modeling errors for each model of order $k = 1$ to p.

The m-file for the forward covariance method is `fcov.m` and is called as follows:

```
>>[gamma,err] = fcov(x,p);
```

Although no m-file is given for the backward covariance method, one may be easily written by modifying `fcov.m`. Specifically, all that is required is to modify the line that computes the reflection coefficients (first line in the `for` loop) along with the line that computes the modeling error, `err(j)`, of the jth-order model (fourth line in the loop). The m-file for Burg's algorithm is `burg.m`, which has the following format:

```
>>[gamma,err] = burg(x,p);
```

The m-file for the modified covariance method on the other hand, is `mcov.m`, which may be called as follows:

```
>>[gamma,err] = mcov(x,p);
```

Once the reflection coefficients of the all-pole model have been found, the approximation $\hat{x}(n)$ of $x(n)$ may be computed as follows:

```
>>xhat=filter(1,gtoa(gamma),[1, zeros(length(x)-1)]);
```

This command first converts the reflection coefficients into the direct form filter coefficients. These coefficients are then used to find the unit sample response of the filter that is used to model $x(n)$.

Special Notes: For each m-file, the model order `p` must be a positive integer that is less than the length of the vector `x`.

A.7 OPTIMUM FILTERS

In Chapter 7 we looked at the problem of designing a filter that would produce the minimum mean-square estimate of a process $d(n)$ from a related process $x(n)$. Although no m-files were given in Chapter 7, many of the filters may be designed and implemented using standard MATLAB functions. As an example, here we will illustrate how to design an FIR Wiener filter for noise cancellation as described Sect. 7.2.3. Specifically, we will look at the filtering problem considered in Example 7.2.6.

Let $x(n) = d(n) + v_1(n)$ where $d(n)$, the desired signal, is a sinusoid of frequency $\omega = 0.05\pi$ that is observed in the presence of additive noise, $v_1(n)$. Given a related noise process, $v_2(n)$, that is correlated with $v_1(n)$, a Wiener noise cancellation filter to estimate

$d(n)$ may be designed as follows. As defined in Example 7.2.6, let

$$v_1(n) = 0.8v_1(n-1) + g(n)$$
$$v_2(n) = -0.6v_2(n-1) + g(n)$$

where $g(n)$ is a zero-mean, unit variance white Gaussian noise process. We may generate 200 samples of each of these processes with the following sequence of commands:

```
>>d=sin(0.05*pi*[1:200] + 2*pi*rand);
>>g=randn(1,200);
>>v1=filter(1,[1 -0.8],g);
>>v2=filter(1,[1 0.6],g);
>>x=d+v1;
```

The coefficients of the Wiener filter to estimate $v_1(n)$ from $v_2(n)$ are found by solving the linear equations $\mathbf{R}_{v_2}\mathbf{w} = \mathbf{r}_{xv_2}$. First, however, we must estimate the autocorrelation matrix of $v_2(n)$, and the cross-correlation between $x(n)$ and $v_2(n)$. Given $x(n)$ and $v_2(n)$ these may be estimated as follows:

```
>>Rv2=covar(v2,4);
>>rxv2=convm(x,4)'*convm(v2,4)/(length(x)-1);
```

Next, the linear equations are solved with the command

```
>>w=rxv2(1,:)/Rv2;
```

and the estimate of the additive noise $v_1(n)$ generated by filtering $v_2(n)$ as follows:

```
>>v1hat=filter(w,1,v2);
```

Finally, the estimate of $d(n)$ is the difference between $x(n)$ and $\hat{v}_1(n)$,

```
>>dhat=x-v1hat;
```

A.8 SPECTRUM ESTIMATION

In Chapter 8 we considered the problem of estimating the power spectrum $P_x(e^{j\omega})$ of a wide-sense stationary process $x(n)$ from a finite sequence of observations. In this section, we describe the m-files for spectrum estimation that are found in Chapter 8. The inputs to these m-files vary, depending on what parameters need to be specified in the spectrum estimation technique. Each, however, requires an input vector x, which may be either real-valued or complex-valued, that contains the values of the process $x(n)$ that is to be analyzed. For the classical spectrum estimation m-files, the output is a vector Px that contains 1024 samples of the estimated spectrum, $\hat{P}_x(e^{j\omega})$, over the frequency range $[0, 2\pi]$. For the remaining m-files (except the Pisarenko harmonic decomposition), the vector Px contains 1024 values of the spectrum estimate in dB, i.e., $10\log_{10}\hat{P}_x(e^{j\omega})$.

Once $\hat{P}_x(e^{j\omega})$ has been computed, making a plot of the spectrum estimate is straightforward. For example, the periodogram may be plotted as follows:

```
>>Px=periodogram(x);
>>plot(Px)
```

or, more simply, by typing

```
>>plot(periodogram(x))
```

Alternatively, to plot the spectrum in dB we may use the command

```
>>plot(10*log10(Px))
```

To label the frequency axis in Hertz, we set `FS` equal to the sampling frequency, f_s, and type the commands

```
>>f=FS*[0:1023]/1024;
>>plot(f,Px)
```

Note that if `FS=2`, then the axis will be labeled in units of π. Finally, it should be pointed out that if $x(n)$ is real, then the power spectrum is symmetric,

$$\hat{P}_x(e^{j\omega}) = \hat{P}_x(e^{-j\omega})$$

In this case, it is only necessary to plot the first half of the array `Px`,

```
>>plot(Px(1:512))
```

or,

```
>>plot(f(1:512),Px(1:512))
```

One of the experiments described in Chapter 8 is to generate an overlay plot of the spectrum estimates that are generated from an ensemble of realizations of a given process. These plots are useful in studying the variability or *stability* of a spectrum estimation technique, and in estimating the expected value of $\hat{P}_x(e^{j\omega})$. To generate such a plot, it is necessary to write an m-file similar to the one given in Fig. A.4, which generates `num` periodograms of a process consisting of a sum of sinusoids in white noise. The output of this m-file, `Px`, is a matrix having `num` columns where each column is a periodogram of one of the sequences in the ensemble. For example, the command

```
>>Px=overlay(40,[0.4*pi,0.45*pi],[5,5],1,50);
```

generates 50 periodograms using 40 samples of the process

$$x(n) = 5\sin(0.4\pi n + \phi_1) + 5\sin(0.45\pi n + \phi_2) + v(n)$$

where ϕ_1 and ϕ_2 are random variables that are uniformly distributed over the interval $[-\pi, \pi]$, and $v(n)$ is unit variance white Gaussian noise. From the matrix `Px`, an overlay plot of the periodograms may be displayed with the command

```
>>plot(10*log10(Px))
```

and the average of the spectrum estimates may be plotted as follows:

```
>>plot(10*log10(sum(Px')/num))
```

In the following sections we describe the m-files for spectrum estimation, beginning with the classical methods in Section A.8.1. In Section A.8.2, we then describe the m-files for the nonclassical methods, which include the minimum variance method, the maximum entropy method, and the model-based methods. Finally, the m-files for frequency estimation are presented in Section A.8.3.

Generating Multiple Spectrum Estimates

```
function Px=overlay(N,omega,A,sigma,num)
%  Calculates the periodogram using an ensemble of realizations of a
%  process consisting of a sum of complex exponentials in white noise.
%
%  Px=overlay(N,omega,A,sigma,num)
%       N     :  length of the signal
%       omega :  vector containing the sinusoid frequencies
%       A     :  vector containing the sinusoid amplitudes
%       sigma :  variance of the white noise
%       num   :  size of the ensemble (number of periodograms)
%
   jj=length(omega);
   n=1:N;
   for i=1:num
        x=sigma*randn(1,N);
        for j=1:jj
              phi=2*pi*rand(1);
              x=x+A(j)*sin(omega(j)*n+phi);
              end;
        Px(:,i)=periodogram(x);    % This command may be replaced
                                   % with another m-file.
        end;
```

Figure A.4 *An m-file to find the periodogram of an ensemble of realizations of a process consisting of a sum of complex exponentials in white noise. These periodograms may then be used to generate overlay plots or an estimate of the expected value of the periodogram.*

A.8.1 Classical Methods

In this section we describe the m-files for the classical approaches to spectrum estimation, which include the periodogram, the modified periodogram, Bartlett's method, Welch's method, and the Blackman-Tukey method. We begin with the m-file for the periodogram, which is used in each of the other m-files described in this section.

The Periodogram. The periodogram of $x(n)$, using samples from $n = n_1$ to $n = n_2$, is defined by

$$\hat{P}_{per}(e^{j\omega}) = \frac{1}{n_2 - n_1 + 1} \left| \sum_{n=n_1}^{n_2} x(n) e^{-jn\omega} \right|^2$$

The m-file to compute the periodogram is `periodogram.m`, and the format of the command line for this m-file is

```
>>Px=periodogram(x,n1,n2);
```

The only required input is the vector `x`, which contains the signal values $x(n)$. The inputs `n1` and `n2` are optional integers that indicate the values of $x(n)$ within the vector `x` that are to be used, i.e., `x(n1)` to `x(n2)`. If these integers are left unspecified, as in the command

```
>>Px=periodogram(x);
```

then the periodogram is computed using all of the values in the vector x. The output Px is a vector that contains 1024 samples of the periodogram from $\omega = 0$ to $\omega = 2\pi$.

Special Notes: The inputs n1 and n2 must be positive integers with n1 < n2 ≤ length(x). This program calls the m-file fft.m and, by default, an FFT of length 1024 is used. Therefore, if n2−n1+1 is greater than 1024, then periodogram.m must be modified to accommodate a longer FFT.

The Modified Periodogram. The modified periodogram allows the user to window $x(n)$ prior to computing the periodogram,

$$\hat{P}_M(e^{j\omega}) = \frac{1}{n_2 - n_1 + 1} \left| \sum_{n=n_1}^{n_2} w(n)x(n)e^{-jn\omega} \right|^2$$

The window $w(n)$ allows one to trade-off spectral resolution, which is determined by the width of the main lobe of $W(e^{j\omega})$, and spectral masking, which is due to the sidelobes of $W(e^{j\omega})$. The m-file for the modified periodogram, mper.m, is called as follows:

```
>>Px = mper(x,win,n1,n2);
```

The only required inputs are x, the vector of signal values $x(n)$, and win, an integer that specifies the type of window that is to be used. The correspondence between values of win and the type of window is given in Fig. A.5. As with the periodogram, the optional inputs n1 and n2 may be used to select a subset of the values of $x(n)$ within the array x. If n1 and n2 are left unspecified, then all of the values in the vector x will be used.

Special Notes: Same as for the m-file periodogram.m. In addition, win should be an integer within the range from win=1 to win=5.

Bartlett's Method. Bartlett's method produces a spectrum estimate by averaging the periodograms that are formed from nonoverlapping subsequences of $x(n)$. Periodogram averaging allows one to trade spectral resolution for a reduction in the variance of the estimate. The m-file for Bartlett's method is bart.m, and the format of the command line is

```
>>Px=bart(x,nsect);
```

where x is a vector of signal values and nsect is the number of subsequences that are to be extracted from $x(n)$. As with the periodogram, the output Px is a vector of length 1024 that contains the values of the estimated spectrum from $\omega = 0$ to $\omega = 2\pi$.

Special Notes: The number of subsequences, nsect, must be a positive integer. Although nsect is allowed to be as large as length(x), a much smaller value is typically used in order to ensure that the Bartlett estimate has sufficient resolution.

```
win=1 : Rectangular window
win=2 : Hamming window
win=3 : Hanning window
win=4 : Bartlett window
win=5 : Blackman window
```

Figure A.5 *The types of windows and the corresponding values for the parameter* win *in the m-files for the modified periodogram, Welch's method, and the Blackman-Tukey method.*

Welch's Method. Welch's method is a generalization of Bartlett's method that allows a window to be applied to $x(n)$, thereby replacing the periodograms in Bartlett's method with modified periodograms. In addition, Welch's method allows the subsequences that are extracted from $x(n)$ to overlap. The m-file for Welch's method is `welch.m` and the format of the command line is

```
Px = welch(x,L,over,win);
```

The inputs to this m-file include the vector `x`, which contains the signal values $x(n)$, an integer `L`, which defines the length of the sections that are to be extracted from the vector `x`, the parameter `over`, which is a real number in the range $0 \leq$ `over` < 1 that defines the amount of overlap between the subsequences, and `win`, an integer that specifies the type of window that is to be used according to the list given in Fig. A.5. The overlap that is specified by `over` is given as a percentage. For example, if `over=0`, then there is no overlap, and if `over=0.5`, then there will be a 50% overlap between subsequences. If the value of `win` is unspecified, then a rectangular window is used. If, in addition, the value of `over` is undefined, then `over` is set to zero, and

```
Px = welch(x,L);
```

is equivalent to Bartlett's method. However, this way of implementing Batlett's method differs from the command

```
Px = bartlett(x,nsect);
```

in that it allows the user to specify the section length rather than the number of sections. Finally, if `welch.m` is used with only one input as in the command

```
Px = welch(x);
```

then a rectangular window is used, the overlap is set to zero, and the section length is set equal to the length of the vector `x`. Thus, this command is equivalent to the command

```
Px = periodogram(x);
```

for computing the periodogram.

Special Notes: Same as for the m-file `periodogram.m`. In addition, the value for the overlap must satisfy $0 \leq$ `over` < 1. Overlaps of 50% (`over=.5`) and 75% (`over=.75`) are commonly used.

Blackman-Tukey Method. The Blackman-Tukey method of periodogram smoothing involves multiplying an estimated autocorrelation sequence, $\hat{r}_x(k)$, with a window, $w(k)$, and then taking the discrete-time Fourier transform,

$$\hat{P}_{BT}(e^{j\omega}) = \sum_{k=-M}^{M} \hat{r}_x(k)w(k)e^{-jk\omega}$$

The m-file for the Blackman-Tukey method is `per_smooth.m` and the format of the command line is

```
Px = per_smooth(x,win,M,n1,n2);
```

The only required inputs for this m-file are `x`, `win`, and `M`. As with the previous m-files, `x` is a vector that contains the signal values, `win` is an integer that defines the type of window that is to be used as given in Fig. A.5, and `M` is the length of the window. The inputs `n1` and

n2 are optional integers that indicate the values of $x(n)$ within the vector x that are to be used. If these integers are left unspecified then the entire data record is used.

Special Notes: Same as for the m-file for the modified periodogram, `mper.m`. In addition, the length of the window, M, must be in the range $0 < \texttt{M} \leq \texttt{length(x)}$.

A.8.2 Nonclassical Methods

In this section we describe the nonclassical spectrum estimation techniques, which include the minimum variance method, the maximum entropy method, and the model-based methods.

Minimum Variance Method. For a wide-sense stationary process $x(n)$, the minimum variance estimate of the power spectrum is

$$\hat{P}_{MV}(e^{j\omega}) = \frac{p}{\mathbf{e}^H \mathbf{R}_x^{-1} \mathbf{e}} \tag{A.4}$$

where $\mathbf{R}_x$ is the $p \times p$ autocorrelation matrix. The m-file that is used to compute the minimum variance estimate is `minvar.m`, and the format of the command line is

```
>>Px = minvar(x,p);
```

where x is a vector that contains the values $x(n)$, and p is the order of the minimum variance estimate. The order p is equal to the order of the bandpass filters used in the minimum variance filterbank, and is equal to the size of the autocorrelation matrix, $\mathbf{R}_x$. The output, Px, is a vector that contains 1024 samples of the minimum variance estimate (in dB) from $\omega = 0$ to $\omega = 2\pi$. Instead of evaluating Eq. (A.4) directly, `minvar.m` finds the minimum variance estimate using the expansion

$$\hat{P}_{MV}(e^{j\omega}) = \frac{p}{\displaystyle\sum_{i=1}^{p} \frac{1}{\lambda_i} |\mathbf{e}^H \mathbf{v}_i|^2}$$

where λ_i and $\mathbf{v}_i$ are the eigenvalues and eigenvectors, respectively, of $\mathbf{R}_x$. Note that each term in the denominator, $\mathbf{e}^H \mathbf{v}_i$, is the DTFT of the eigenvector $\mathbf{v}_i$. These terms are evaluated in `minvar.m` using a 1024-point FFT.

Special Notes: The order p must be a positive integer. Typically, $\texttt{p} \approx \texttt{N/3}$, where N is the length of the data sequence (assuming that this value is not too large to perform an eigenvalue decomposition of $\mathbf{R}_x$).

Maximum Entropy Method. The maximum entropy method of spectrum estimation finds an all-pole model for a process using the autocorrelation method, and then uses the model parameters $a_p(k)$ to estimate the spectrum as follows:

$$\hat{P}_{mem}(e^{j\omega}) = \frac{\epsilon_p}{\left| 1 + \displaystyle\sum_{k=1}^{p} a_p(k) e^{-jk\omega} \right|^2}$$

The m-file for the maximum entropy method is `mem.m`, and the format of the command line is

```
>>function Px = mem(x,p)
```

where x is the vector that contains the signal values, p is the model order, and Px is a vector that contains 1024 samples of the maximum entropy spectrum (in dB) from $\omega = 0$ to $\omega = 2\pi$.

Special Notes: The order p must be a positive integer that does not exceed the length of the vector x.

Model-based Methods. A model-based approach to spectrum estimation involves the selection of a model, the estimation of the model parameters, and the evaluation of the spectrum using the estimated parameters. The spectrum of a process $x(n)$ may be estimated using any of the signal modeling m-files described in Sect. A.4 and Sect. A.6. For example, an autoregressive spectrum estimate using Burg's method may be found as follows:

```
>>[a,err] = burg(x,p);
>>Px=10*log10(err)-20*log10(abs(fft(a,1024)));
```

A.8.3 Frequency Estimation

In this section we describe the m-files that are used for processes that consist of a sum of complex exponentials in white noise. We begin with the noise subspace methods, which are frequency estimation algorithms and include the Pisarenko harmonic decomposition, the MUSIC algorithm, the eigenvector method, and the minimum norm algorithm. Then we discuss the m-files that use a signal subspace approach, which involves a principal components analysis of the autocorrelation matrix $\mathbf{R}_x$.

Noise Subspace Methods. A noise subspace method to estimate the frequencies of p complex exponentials in noise involves the use of a frequency estimation function of the form

$$\hat{P}_x(e^{j\omega}) = \frac{1}{\sum_{i=p+1}^{M} \alpha_i |\mathbf{e}^H \mathbf{v}_i|^2} \tag{A.5}$$

where $\mathbf{v}_i$ are vectors that lie in the noise subspace of $\mathbf{R}_x$, and α_i are constants. In the Pisarenko harmonic decomposition, the frequency estimation function is

$$\hat{P}_{PHD}(e^{j\omega}) = \frac{1}{|\mathbf{e}^H \mathbf{v}_{\min}|^2}$$

where $\mathbf{v}_{\min}$ is the eigenvector of $\mathbf{R}_x$ that has the minimum eigenvalue. The m-file for Pisarenko's method is phd.m, and the format of the command line to use this m-file is

```
>>[vmin,sigma]=phd(x,p);
```

where x is a vector that contains the values of $x(n)$, and p is an integer that defines the number of complex exponentials in $x(n)$.[5] The output of this m-file is the eigenvector having the smallest eigenvalue, vmin, along with the minimum eigenvalue, sigma, which may be used as an estimate of the white noise variance. The frequencies of the complex exponentials may be estimated from vmin by finding the angles of the eigenfilter $V_{\min}(z)$ as follows:

```
>>freq=angle(roots(vmin));
```

[5]For sinusoids in noise, p is equal to twice the number of sinusoids.

Alternatively, the frequencies may be estimated from the peaks of the eigenspectrum, which may be plotted using the command

```
>>plot(abs(fft(vmin,1024)))
```

The MUSIC algorithm is another frequency estimation technique of the form given in Eq. (A.5). However, with the MUSIC algorithm, the constants α_i are equal to one and $\mathbf{v}_i$ are the $M - p$ eigenvectors of $\mathbf{R}_x$ that have the smallest eigenvalues. The m-file for the MUSIC algorithm is `music.m`, and the format of the command line is

```
>>Px = music(x,p,M);
```

where `x` is a vector that contains the signal values, `p` is a positive integer that specifies the number of complex exponentials in $x(n)$, and `M` is an integer that specifies the size of the autocorrelation matrix and, consequently, defines the number of eigenvectors, `M-p`, that are to be used to form the frequency estimation function $\hat{P}_x(e^{j\omega})$. The integer `M` must be larger than `p` and, in the special case of `M=p+1`, the MUSIC algorithm is equivalent to the Pisarenko harmonic decomposition. The output `Px` is a vector that contains 1024 equally spaced samples of the frequency estimation function (in dB) from $\omega = 0$ to $\omega = 2\pi$.

The eigenvector method is a frequency estimation algorithm that is similar to the MUSIC algorithm. As with the MUSIC algorithm, the vectors $\mathbf{v}_i$ for $i = p + 1$ to M are the eigenvectors that have the smallest eigenvalues. The constants α_i, however, are equal to the inverse of the eigenvalues, $\alpha_i = 1/\lambda_i$. The m-file for the eigenvector method is `ev.m` and the format to use this m-file is

```
>>Px = ev(x,p,M);
```

The inputs, `x`, `p`, and `M` and the output `Px` are as described for the m-file `music.m`.

The last frequency estimation algorithm is the minimum norm method, which uses a frequency estimation function of the same form as that used for the Pisarenko harmonic decomposition,

$$\hat{P}(e^{j\omega})_{MN} = \frac{1}{|\mathbf{e}^H\mathbf{a}|^2}$$

However, instead of using the eigenvector having the smallest eigenvalue, the minimum norm method uses the vector $\mathbf{a}$ in the noise subspace that has the minimum norm. The m-file for the minimum norm algorithm is `min_norm.m` and the format of the command line is

```
>>Px = min_norm(x,p,M);
```

where `x`, `p`, and `M` are as described for the MUSIC algorithm, and `Px` contains the samples of the frequency estimation function in dB.

Special Notes: For each of the noise subspace m-files, `M` and `p` must be positive integers with `M` $\geq$ `p+1`. Since the m-files are frequency estimation algorithms, the vector `Px` is not an estimate of the power spectrum, and should only be used to extract estimates of the complex exponential frequencies, ω_k.

Principal Components Spectrum Estimation. For a wide-sense stationary process consisting of p complex exponential in noise, a principal components method of spectrum estimation finds a reduced rank approximation to the autocorrelation matrix using the p

principal eigenvectors,

$$\hat{\mathbf{R}}_s = \sum_{i=1}^{p} \lambda_i \mathbf{v}_i \mathbf{v}_i^H$$

and then estimates the power spectrum from $\hat{\mathbf{R}}_x$. For example, a principal components approach using the Blackman-Tukey method and a Bartlett window yields the estimate

$$\hat{P}_{PC-BT}(e^{j\omega}) = \frac{1}{M}\mathbf{e}^H \hat{\mathbf{R}}_s \mathbf{e}$$

An m-file to find $\hat{P}_{PC-BT}(e^{j\omega})$ is `bt_pc.m`, which has a command line for the form

```
>>Px = bt_pc(x,p,M);
```

where `x` is the vector containing the signal values, `p` is the number of complex exponentials, and `M` is the size of the autocorrelation matrix $\mathbf{R}_x$ (length of the Bartlett window). The output `Px` is a vector containing 1024 samples of the estimated power spectrum (in dB) from $\omega = 0$ to $\omega = 2\pi$.

Although `bt_pc.m` is the only m-file that was given in Chapter 8 for principal components spectrum estimation, using this m-file as a model, m-files for the other principal components methods, such as the minimum variance method and the maximum entropy method, may be easily written.

A.9 ADAPTIVE FILTERING

In Chapter 9, we considered the problem of estimating a process $d(n)$ from the values of a related process $x(n)$. In this section we describe three m-files that may be used to solve this problem. One of these m-files is for the LMS algorithm, and another is for the normalized LMS algorithm. Both of these algorithms estimate $d(n)$ by filtering $x(n)$ with a time-varying tapped delay line,

$$\hat{d}(n) = \sum_{k=0}^{p-1} w_n(k) x(n-k)$$

where the coefficients $w_n(k)$ are determined using an update equation of the form

$$\mathbf{w}_{n+1} = \mathbf{w}_n + \mu(n) e(n) \mathbf{x}^*(n)$$

The third m-file is for the RLS algorithm, which finds the least squares estimate of $d(n)$. Although many other types of adaptive filtering algorithms were discussed in Chapter 9, in most cases, m-files for these algorithms may be easily implemented with only a slight modifications to one of the m-files described in this section.

LMS. With an FIR LMS adaptive filter, the coefficient update equation has the form

$$\mathbf{w}_{n+1} = \mathbf{w}_n + \mu e(n) \mathbf{x}^*(n)$$

The m-file for the LMS adaptive filter is `lms.m` and may be called as follows:

```
>> [W,E] = lms(x,d,mu,nord,a0);
```

To use this m-file, four inputs are required; the fifth is optional. The first input is the vector x, which contains the values of $x(n)$ that are to be used to estimate $d(n)$. Second is the vector d of desired signal values, $d(n)$. Third is the step size mu, which, for convergence in the mean, must be non-negative and less than $1/\lambda_{max}$ where λ_{max} is the largest eigenvalue of the autocorrelation matrix of $x(n)$. Finally, nord is an integer that defines the order of the adaptive filter (number of filter coefficients). The optional input a0 allows one to define the initial conditions of the adaptive filter. The output W is a matrix that has the coefficients of the adaptive filter at time n stored in the nth row. Thus, a plot of each filter coefficient, $w_n(k)$, as a function of n may be plotted as in Fig. 9.10 with the command

```
>> plot(W)
```

The optional output E is a vector that contains the values of the error, $e(n) = \hat{d}(n) - d(n)$, at time n.

Special Notes: The length of the vector d, the desired signal, must be at least as long as the length of the input vector x. The step size mu must be non-negative and it generally should be much smaller than $1/\lambda_{max}$ where λ_{max} is the largest eigenvalue of the autocorrelation matrix of $x(n)$. The order of the filter, nord, must be a non-negative integer and smaller than the length of the vector x. Finally, the input a0 must be a vector with a length equal to the value of nord.

Normalized LMS. The normalized LMS algorithm, which has a coefficient update equation of the form

$$\mathbf{w}_{n+1} = \mathbf{w}_n + \beta \frac{\mathbf{x}^*(n)}{\epsilon + \|\mathbf{x}(n)\|^2} e(n)$$

is simpler to use than the LMS algorithm in that it allows the step size to be selected without having to know the largest eigenvalue of the autocorrelation matrix of $x(n)$. The normalized LMS algorithm has two parameters. The first, β, is the normalized step size, and the second, ϵ, is a parameter that stabilizes the algorithm in case $\|\mathbf{x}(n)\|^2$ becomes small. The m-file for the normalized LMS algorithm is nlms.m, which may be called as follows:

```
>> [W,E] = nlms(x,d,beta,nord,a0);
```

The inputs and outputs are the same as those described for the LMS algorithm. Specifically, the input is stored in the vector x, the desired signal is stored in the vector d, the step size β is set by the value of beta, the filter order is specified by nord, and the optional initial conditions for the adaptive filter may be specified with a0. The step size β should be positive and no larger than one, $0 < \beta < 1$. In nlms.m, the parameter ϵ is set equal to 0.0001. Therefore, if another value is desired, the m-file would need to be modified accordingly. As with the LMS algorithm, the output W is a matrix with the nth row containing the adaptive filter coefficients at time n, and E is a vector that contains the errors, $e(n) = \hat{d}(n) - d(n)$.

Special Notes: Same as for the m-file lms.m except that the step size must be in the range 0 < beta < 1.

Recursive Least Squares. The recursive least squares algorithm produces the set of filter coefficients $w_n(k)$ at time n that minimize the weighted least squares error

$$\mathcal{E}(n) = \sum_{i=0}^{n} \lambda^{n-i} |e(i)|^2$$

The m-file for the RLS algorithm is `rls.m`, and is called using a command of the following form:

```
>> [W,E] = rls(x,d,lambda,nord);
```

The inputs to this m-file include the input signal vector `x`, the desired signal vector `d`, the exponential weighting factor `lambda`, and the filter order `nord`. The outputs of `rls.m` are the same as those for `lms.m` and `nlms.m`.

Special Notes: The length of the vector `d`, the desired signal, must be at least as long as the length of the input vector `x`. The exponential weighting factor, `lambda`, must be greater than zero and less than or equal to one. If `lambda` is equal to one, then the m-file implements the growing window RLS algorithm. The order of the filter, `nord`, must be a non-negative integer that is smaller than the length of the vector `x`.

SYMBOLS

OPERATIONS

$\mathbf{a}^T$, $\mathbf{A}^T$	Transpose of a vector **a** or matrix **A** (p. 21, 27)
$\mathbf{a}^H$, $\mathbf{A}^H$	Hermitian transpose of a vector **a** or matrix **A** (p. 21, 27)
$\mathbf{a}^*$, $\mathbf{A}^*$	Complex conjugate of a vector **a** or matrix **A** (p. 21, 27)
$\mathbf{a}_j^R$, $A_j^R(z)$	The reciprocal vector and its z-transform (p. 223, 224)
$\mathbf{A}^{-1}$	Inverse of **A** (p. 28)
$\mathbf{A}^+$	Pseudoinverse of **A** (p. 32, 34)
$\mathrm{Cov}(x, y)$	Covariance between x and y (p. 66)
$\det(\mathbf{A})$	Determinant of **A** (p. 29)
$E\{x\}$	Expected value of x (p. 63)
$\Pr\{A\}$	The probability of event A (p. 58)
$\mathrm{Toep}\{\mathbf{a}\}$	Hermitian Toeplitz matrix with **a** as the first column (p. 38)
$\mathrm{tr}(\mathbf{A})$	Trace of **A** (p. 30)
$\mathrm{Var}\{x\}$	Variance of x (p. 64)
∇_x	Gradient vector (p. 49)
$\|\mathbf{x}\|$	Norm of the vector **x** (p. 22)
$\langle \mathbf{a}, \mathbf{b} \rangle$	Inner product of **a** and **b** (p. 22)
$[P(z)]_+$	The causal (positive-time) part of $P(z)$ (p. 191)
$[P(z)]_-$	The anti-causal (negative-time) part of $P(z)$ (p. 191)

PRINCIPAL SYMBOLS

$\mathbf{a}_p$	Augmented vector of all-pole model coefficients (p. 148)
$\tilde{\mathbf{a}}_p$	Vector of all-pole model coefficients (p. 136)
$\mathbf{b}_q$	Vector of numerator coefficients $b_q(k)$ (p. 137)
$c_x(k)$	The autocovariance of a WSS process $x(n)$ (p. 77)
$c_x(k, l)$	The autocovariance of $x(n)$ (p. 77)
c_{xy}	The covariance between x and y (p. 66)
$c_{xy}(k, l)$	The cross-covariance between $x(n)$ and $y(n)$ (p. 79)
$\mathbf{c}_n$	Weight error vector (p. 501)
$\mathbf{C}_x$	Autocovariance matrix (p. 85)
$e_p^+(n)$, $E_p^+(z)$	Forward prediction error and its z-transform (p. 290)
$e_p^-(n)$, $E_p^-(z)$	Backward prediction error and its z-transform (p. 291)
$\mathbf{e}_i$	Vector of complex exponentials (p. 427)
$\mathcal{E}(n)$	Least squares error for RLS (p. 541)
$\mathcal{E}_p$	All-pole modeling error (p. 161)
$\mathcal{E}_{p,q}$	Prony error (p. 145)
$\mathcal{E}_p^+$	The squared p-th order forward prediction error (p. 290, 308)
$\mathcal{E}_p^-$	The squared p-th order backward prediction error (p. 292, 313)
$\mathcal{E}_p^B$	Burg error of order p (p. 317)
$\mathcal{E}_p^C$	Covariance error of order p (p. 182)

$\mathcal{E}_p^M$	Error for the modified covariance method (p. 322)
$\mathcal{E}_{LS}$	Least squares error (p. 132)
$\mathcal{E}_S$	Shanks' error (p. 156)
$f_x(\alpha)$	Probability density function (p. 61)
$F_x(\alpha)$	Probability distribution function (p. 61)
$h(n)$	Unit sample response (p. 11)
$H(e^{j\omega})$	Frequency response (p. 13)
$H(z)$	System function (p. 15)
$\mathbf{I}$	Identity matrix (p. 28)
$\mathbf{J}$	Exchange matrix (p. 36)
$\mathbf{m}_x$	The mean vector (p. 85)
$m_x(n)$	Mean of a random process $x(n)$ (p. 77)
m_x	The mean of the random variable x (p. 66) or WSS process $x(n)$ (p. 82)
$\mathcal{M}$	Figure of merit (p. 424) or level of misadjustment (p. 513)
P_A	Projection matrix (p. 34)
$\hat{P}_{AR}(e^{j\omega})$	Autoregressive spectrum estimate (p. 442)
$\hat{P}_{ARMA}(e^{j\omega})$	Autoregressive moving average spectrum estimate (p. 450)
$\hat{P}_B(e^{j\omega})$	Spectrum estimate using Bartlett's method (p. 413)
$\hat{P}_{BT}(e^{j\omega})$	Blackman-Tukey estimate (p. 421)
$\hat{P}_{EV}(e^{j\omega})$	Frequency estimation function, eigenvector method (p. 465)
$\hat{P}_M(e^{j\omega})$	Modified periodogram (p. 410)
$\hat{P}_{MA}(e^{j\omega})$	Moving average spectrum estimate (p. 448)
$\hat{P}_{mem}(e^{j\omega})$	Spectrum estimate using the maximum entropy method (p. 436)
$\hat{P}_{MN}(e^{j\omega})$	Frequency estimation function, minimum norm method (p. 466)
$\hat{P}_{MU}(e^{j\omega})$	Frequency estimation function, MUSIC algorithm (p. 464)
$\hat{P}_{MV}(e^{j\omega})$	Minimum variance spectrum estimate (p. 429)
$\hat{P}_{PC-AR}(e^{j\omega})$	Principal components autoregressive spectrum estimate (p. 472)
$\hat{P}_{PC-BT}(e^{j\omega})$	Principal components Blackman-Tukey estimate (p. 471)
$\hat{P}_{PC-MV}(e^{j\omega})$	Principal components, minimum variance estimate (p. 471)
$\hat{P}_{per}(e^{j\omega})$	Periodogram (p. 394)
$\hat{P}_{phd}(e^{j\omega})$	Pisarenko harmonic decomposition (p. 459)
$\hat{P}_W(e^{j\omega})$	Spectrum estimate using Welch's method (p. 418)
$P_x(e^{j\omega})$, $P_x(z)$	Power spectrum of $x(n)$ (p. 95)
$\text{Res}[\hat{P}_x(e^{j\omega})]$	Resolution of $\hat{P}_x(e^{j\omega})$ (p. 402)
$r_x(k)$	The autocorrelation of a WSS process $x(n)$ (p. 83)
$r_x(k,l)$	The autocorrelation of $x(n)$ (p. 77)
r_{xy}	The correlation between x and y (p. 66)
$r_{xy}(k,l)$	The cross-correlation between $x(n)$ and $y(n)$ (p. 79)
$\mathbf{R}_x$	Autocorrelation matrix (p. 85)
$\mathbf{s}_j$, $\mathbf{s}_j^\star$	Singular predictor vectors (p. 270)

$S_j(z)$, $S_j^*(z)$	Singular predictor polynomials (p. 269)
$u(n)$	Unit step (p. 8)
$u_0(\omega)$	Unit impulse (p. 13)
$\mathbf{u}_1$	Unit vector with one as its first element (p. 25)
$\mathbf{v}_{\min}$	Eigenvector having the minimum eigenvalue (p. 459)
$V_{\min}(z)$	Eigenfilter corresponding to the minimum eigenvalue (p. 459)
$\mathbf{V}$	Matrix of eigenvectors (p. 44)
$\mathcal{V}$	Variability (p. 424)
$w_R(n)$	Rectangular window (p. 178)
$W_R(e^{j\omega})$	DTFT of a rectangular window (p. 409)
$w_B(n)$	Bartlett window (p. 398)
$W_B(e^{j\omega})$	DTFT of a Bartlett window (p. 399)
$x^f(n)$, $y^f(n)$	Filtered signals used in an adaptive recursive filter (p. 538)
$x_N(n)$	Windowed signal (p. 394)
β	Normalized step size (p. 514)
γ_j	Parameter used in the Levinson-Durbin recursion (p. 217)
Γ_j	Reflection coefficient (p. 218)
$\mathbf{\Gamma}_p$	Reflection coefficient vector (p. 222)
Γ_j^B	Reflection coefficients, Burg's method (p. 317)
Γ_j^I	Reflection coefficients, Itakura's method (p. 316)
Γ_j^+	Reflection coefficients, forward covariance method (p. 308)
Γ_j^-	Reflection coefficients, backward covariance method (p. 313)
δ_j	Levinson recursion parameter (p. 265)
$\delta(n)$	Unit sample (p. 8)
ϵ_p	Minimum all-pole modeling error (p. 163)
$\epsilon_{p,q}$	Minimum Prony modeling error (p. 148)
$\varepsilon_j(n)$	Singular forward prediction error (p. 298)
λ_i	Eigenvalue (p. 41)
$\lambda_{\max}$	Maximum eigenvalue (p. 47)
$\mathbf{\Lambda}$	Diagonal matrix of eigenvalues (p. 44)
μ	Adaptive filter step size (p. 500)
ξ	Mean-square error (p. 68)
$\xi(n)$	Mean-square error at time n (p. 499)
$\xi_{\text{ex}}(n)$	Excess mean-square error (p. 511)
$\rho(\mathbf{A})$	Rank of the matrix $\mathbf{A}$ (p. 28)
ρ_{xy}	Correlation coefficient (p. 66)
σ_x^2	Variance of the random variable x (p. 64) or WSS process $x(n)$ (p. 82)
$\sigma_x^2(n)$	Variance of the random process $x(n)$ (p. 77)
τ	Adaptive filter time constant (p. 503)
τ_j, τ_j^*	Parameters that appear in the split Levinson recursion (p. 271)
χ	Condition number of a matrix (p. 503)
$\psi_k^a(n)$, $\psi_k^b(n)$	Gradient estimate used in an adaptive recursive filter (p. 538)
Ω	Sample space (p. 58)

Index