Note that these equations are singular, indicating that a lower model order is possible. Therefore, setting $a(2) = 0$ and solving for $a(1)$ we find

$$a(1) = -r_x(1, 0)/r_x(1, 1) = 1$$

Thus,

$$A(z) = 1 + z^{-1}$$

and, with $b(0) = 1$, the model is

$$H(z) = \frac{1}{1 + z^{-1}}$$

As a result, the model produces an exact fit to the data for $n = 0, 1, \ldots, N$.

Before concluding, we mention some modifications and generalizations to the covariance method that may be appropriate in certain applications. Recall that in the development of the covariance method it was assumed that $x(n)$ is only known or specified over the finite interval $[0, N - 1]$. As a result, Prony's error was modified so that it included only those values of $e(n)$ that could be evaluated from the given data. Specifically, the covariance method minimizes the error

$$\mathcal{E}_p^C = \sum_{n=p}^{N} |e(n)|^2 \tag{4.132}$$

However, if $x(n)$ begins at time $n = 0$, i.e., $x(n) = 0$ for $n < 0$, then $e(n)$ may be evaluated for all $n \leq N$. Therefore, the covariance method may be modified to minimize the error

$$\mathcal{E}_p^{C^-} = \sum_{n=0}^{N} |e(n)|^2$$

In this case, the all-pole coefficients are found by solving the covariance normal equations in Eq. (4.127) with the autocorrelations $r_x(k, l)$ computed as follows

$$r_x(k, l) = \sum_{n=0}^{N} x(n - l)x^*(n - k) \tag{4.133}$$

This method is sometimes referred to as the *prewindowed covariance method.* In a similar vein, if $x(n)$ is known to be equal to zero for $n > N$, then we may minimize the error

$$\mathcal{E}_p^{C^+} = \sum_{n=p}^{\infty} |e(n)|^2$$

which leads to the *post-windowed covariance method.* Again, the all-pole coefficients are found by solving Eq. (4.127) with the autocorrelation sequence computed as in Eq. (4.133) with the sum extending from $n = p$ to $n = \infty$.

Each form of covariance method may also be modified to accommodate the case in which $x(n)$ is given over an arbitrary interval, say $[L, U]$. For example, with the covariance method the error to be minimized would become

$$\mathcal{E}_p^C = \sum_{n=L+p}^{U} |e(n)|^2$$

and the normal equations that must be solved are the same as Eq. (4.127) with

$$r_x(k,l) = \sum_{n=L+p}^{U} x(n-l)x^*(n-k)$$

Finally, although derived only for the case of all-pole modeling, the covariance method may also be used for pole-zero modeling. In this case, Eq. (4.132) would be modified as follows

$$\mathcal{E}_{p,q}^{C} = \sum_{\max(q+1,p)}^{N} |e(n)|^2$$

and the coefficients are found by solving the normal equations Eq. (4.35) with

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

4.7 STOCHASTIC MODELS

In the preceding sections, we looked at different approaches for modeling deterministic signals. In each case, the values of $x(n)$ were known, either for all n or for values of n over some fixed finite interval. In some applications, however, it is necessary to develop models for random processes—signals whose values are unknown and may only be described probabilistically. Examples include signals whose time evolution is affected or driven by random or unknown factors, as is the case for electrocardiograms, unvoiced speech, population statistics, sunspot numbers, economic data, seismograms, and sonar data. Models for random processes differ from those for deterministic signals in two ways. First, since a random process may only be characterized statistically and the values of $x(n)$ are only known in a probabilistic sense, the errors that are minimized for deterministic signals are no longer appropriate. Recall, for example, that with Prony's method the coefficients $a_p(k)$ are found by minimizing the deterministic squared error

$$\mathcal{E}_p = \sum_{n=q+1}^{\infty} |e(n)|^2 = \sum_{n=q+1}^{\infty} \left| x(n) + \sum_{k=1}^{p} a_p(k)x(n-k) \right|^2 \tag{4.134}$$

Therefore, if $x(n)$ is only known probabilistically, then it is not feasible to consider minimizing $\mathcal{E}_p$. The second difference is in the characteristics of the signal that is used as the input to the system that is used to model $x(n)$. Whereas for deterministic signals the input signal was chosen to be a unit sample, for a random process the input signal must be a random process. Typically, this input will be taken to be unit variance white noise.

With these two differences in mind, in this section we consider the problem of modeling a wide-sense stationary random process. As we will see, the Yule-Walker equations will play an important role in the development of stochastic modeling algorithms. We begin, in Section 4.7.1, with the problem of modeling an autoregressive moving average process. We will develop two approaches, the Modified Yule-Walker Equation (MYWE) method and the least squares MYWE method. Both of these approaches are based on solving the Yule-Walker equations. Then, in Section 4.7.2, we consider the special case of autoregressive (all-pole) processes. Here we will see that all-pole modeling of a random process is similar to all-pole modeling of a deterministic signal. The primary difference is in how the autocorrelations are defined in the stochastic all-pole normal equations. Finally, in Section 4.7.3, we

consider the problem of modeling a moving average (all-zero) process and develop two new approaches. The first, *spectral factorization*, requires factoring a polynomial of order $2q$ for a qth-order moving average process. The second, *Durbin's method*, avoids polynomial rooting by modeling the moving average process with a high-order all-pole model and then forming an estimate of the moving average coefficients from the all-pole coefficients.

4.7.1 Autoregressive Moving Average Models

As we saw in Section 3.6.1, a wide-sense stationary ARMA(p, q) process may be generated by filtering unit variance white noise $v(n)$ with a causal linear shift-invariant filter having p poles and q zeros,

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

Therefore, a random process $x(n)$ may be modeled as an ARMA(p, q) process using the model shown in Fig. 4.21 where $v(n)$ is unit variance white noise. To find the filter coefficients that produce the best approximation $\hat{x}(n)$ to $x(n)$ we could take the approach used in Section 4.2, replacing the least squares error $\mathcal{E}_{LS}$ with a mean square error

$$\xi_{MS} = E\{|x(n) - \hat{x}(n)|^2\}$$

However, this would lead to a set of nonlinear equations just as it did in the least squares method of Section 4.2. Therefore, we will consider another approach.

In Section 3.6.1 (p. 110), we saw that the autocorrelation sequence of an ARMA(p, q) process satisfies the Yule-Walker equations[13]

$$\boxed{r_x(k) + \sum_{l=1}^{p} a_p(l) r_x(k-l) = c_q(k)} \tag{4.135}$$

where the sequence $c_q(k)$ is the convolution of $b_q(k)$ and $h^*(-k)$,

$$c_q(k) = b_q(k) * h^*(-k) = \sum_{l=0}^{q-k} b_q(l+k) h^*(l) \tag{4.136}$$

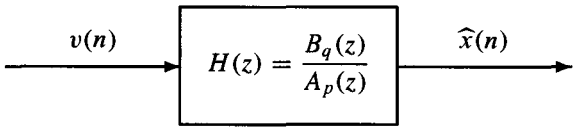


Figure 4.21 *Modeling a random process $x(n)$ as the response of a linear shift-invariant filter to unit variance white noise.*

[13]Note that σ_v^2 in Eq. (3.115) has been set equal to one since here we are assuming that $v(n)$ has unit variance.

and $r_x(k)$ is a statistical autocorrelation,

$$r_x(k) = E\{x(n)x^*(n-k)\}$$

Since $h(n)$ is assumed to be causal, then $c_q(k) = 0$ for $k > q$ and the Yule-Walker equations for $k > q$ are a function only of the coefficients $a_p(k)$,

$$\boxed{r_x(k) + \sum_{l=1}^{p} a_p(l) r_x(k-l) = 0 \quad ; \quad k > q} \tag{4.137}$$

Expressing Eq. (4.137) in matrix form for $k = q+1, q+2, \ldots, q+p$ we have

$$\begin{bmatrix} r_x(q) & r_x(q-1) & \cdots & r_x(q-p+1) \\ r_x(q+1) & r_x(q) & \cdots & r_x(q-p+2) \\ \vdots & \vdots & & \vdots \\ r_x(q+p-1) & r_x(q+p-2) & \cdots & r_x(q) \end{bmatrix} \begin{bmatrix} a_p(1) \\ a_p(2) \\ \vdots \\ a_p(p) \end{bmatrix} = - \begin{bmatrix} r_x(q+1) \\ r_x(q+2) \\ \vdots \\ r_x(q+p) \end{bmatrix} \tag{4.138}$$

which is a set of p linear equations in the p unknowns, $a_p(k)$. These equations, referred to as the *Modified Yule-Walker equations*, allow us to find the coefficients $a_p(k)$ in the filter $H(z)$ from the autocorrelation sequence $r_x(k)$ for $k = q, q+1, \ldots, q+p$. This approach, therefore, is called the *Modified Yule-Walker Equation* (MYWE) method. If the autocorrelations are unknown, then they may be replaced with estimated autocorrelations $\hat{r}_x(k)$ using a sample realization of the process.

Comparing the modified Yule-Walker equations to the Padé equations for the denominator coefficients given in Eq. (4.13), we see that the two sets of equations have exactly the same form. In fact, the only difference between them is in the definition of the "data sequence." In the modified Yule-Walker equations, the "data" consists of the sequence of autocorrelations $r_x(k)$, whereas in the Padé approximation, the data are the values of $x(n)$. As in the Padé approximation method, since the matrix in Eq. (4.138) is a nonsymmetric Toeplitz matrix, these equations may be solved efficiently for the coefficients $a_p(k)$ using the Trench algorithm [18].

Once the coefficients $a_p(k)$ have been determined, the next step is to find the MA coefficients, $b_q(k)$. There are several approaches that may be used to accomplish this. If $x(n)$ is an ARMA(p, q) process with power spectrum

$$P_x(z) = \frac{B_q(z)B_q^*(1/z^*)}{A_p(z)A_p^*(1/z^*)}$$

then filtering $x(n)$ with a linear shift-invariant filter having a system function $A_p(z)$ produces an MA(q) process, $y(n)$, that has a power spectrum

$$P_y(z) = B_q(z)B_q^*(1/z^*)$$

Therefore, the moving average parameters $b_q(k)$ may be estimated from $y(n)$ using one of the moving average modeling techniques described in Section 4.7.3. Alternatively, we may avoid filtering $x(n)$ explicitly and solve for $b_q(k)$ as follows. Given $a_p(k)$, the Yule-Walker

equations may be used to find the values of the sequence $c_q(k)$ for $k = 0, 1, \ldots, q$,

$$\begin{bmatrix} r_x(0) & r_x^*(1) & \cdots & r_x^*(p) \\ r_x(1) & r_x(0) & \cdots & r_x^*(p-1) \\ \vdots & \vdots & & \vdots \\ r_x(q) & r_x(q+1) & \cdots & r_x(0) \end{bmatrix} \begin{bmatrix} 1 \\ a_p(1) \\ \vdots \\ a_p(p) \end{bmatrix} = \begin{bmatrix} c_q(0) \\ c_q(1) \\ \vdots \\ c_q(q) \end{bmatrix}$$

which may be written as

$$\boxed{\mathbf{R}_x \mathbf{a}_p = \mathbf{c}_q} \tag{4.139}$$

Since $c_q(k) = 0$ for $k > q$ the sequence $c_q(k)$ is then known for all $k \geq 0$. We will denote the z-transform of this *causal* or *positive-time* part of $c_q(k)$ by $[C_q(z)]_+$,

$$[C_q(z)]_+ = \sum_{k=0}^{\infty} c_q(k) z^{-k}$$

Similarly, we will denote the *anticausal* or *negative-time* part by $[C_q(z)]_-$,

$$[C_q(z)]_- = \sum_{k=-\infty}^{-1} c_q(k) z^{-k} = \sum_{k=1}^{\infty} c_q(-k) z^k$$

Recall that $c_q(k)$ is the convolution of $b_q(k)$ with $h^*(-k)$. Therefore,

$$C_q(z) = B_q(z) H^*(1/z^*) = B_q(z) \frac{B_q^*(1/z^*)}{A_p^*(1/z^*)}$$

Multiplying $C_q(z)$ by $A_p^*(1/z^*)$ we have

$$\boxed{P_y(z) \equiv C_q(z) A_p^*(1/z^*) = B_q(z) B_q^*(1/z^*)} \tag{4.140}$$

which is the power spectrum of an MA(q) process. Since $a_p(k)$ is zero for $k < 0$, then $A_p^*(1/z^*)$ contains only positive powers of z. Therefore, with

$$P_y(z) = C_q(z) A_p^*(1/z^*) = [C_q(z)]_+ A_p^*(1/z^*) + [C_q(z)]_- A_p^*(1/z^*) \tag{4.141}$$

since both $[C_q(z)]_-$ and $A_p^*(1/z^*)$ are polynomials that contain only positive powers of z, then the causal part of $P_y(z)$ is equal to

$$\boxed{[P_y(z)]_+ = \left[[C_q(z)]_+ A_p^*(1/z^*) \right]_+} \tag{4.142}$$

Thus, although $c_q(k)$ is unknown for $k < 0$, the causal part of $P_y(z)$ may be computed from the causal part of $c_q(k)$ and the AR coefficients $a_p(k)$. Using the conjugate symmetry of $P_y(z)$ we may then determine $P_y(z)$ for all z. Finally, performing a spectral factorization

of $P_y(z)$,

$$\boxed{P_y(z) = B_q(z)B_q^*(1/z^*)} \tag{4.143}$$

produces the polynomial $B_q(z)$. The procedure is illustrated in the following example.

Example 4.7.1 *The MYWE Method for Modeling an ARMA(1,1) Process*

Suppose that we would like to find an ARMA(1,1) model for a real-valued random process $x(n)$ having autocorrelation values

$$r_x(0) = 26 \;\; ; \;\; r_x(1) = 7 \;\; ; \;\; r_x(2) = 7/2$$

The Yule-Walker equations are

$$\begin{bmatrix} r_x(0) & r_x(1) \\ r_x(1) & r_x(0) \\ r_x(2) & r_x(1) \end{bmatrix} \begin{bmatrix} 1 \\ a_1(1) \end{bmatrix} = \begin{bmatrix} c_1(0) \\ c_1(1) \\ 0 \end{bmatrix} \tag{4.144}$$

Thus, the modified Yule-Walker equations are

$$r_x(1)a_1(1) = -r_x(2)$$

which gives $a_1(1) = -r_x(2)/r_x(1) = -1/2$.

For the moving average coefficients we begin by computing $c_1(0)$ and $c_1(1)$ using the Yule-Walker equations as follows

$$\begin{bmatrix} r_x(0) & r_x(1) \\ r_x(1) & r_x(0) \end{bmatrix} \begin{bmatrix} 1 \\ a_1(1) \end{bmatrix} = \begin{bmatrix} c_1(0) \\ c_1(1) \end{bmatrix}$$

With the given values for $r_x(k)$, using $a_1(1) = -1/2$, we find

$$\begin{bmatrix} c_1(0) \\ c_1(1) \end{bmatrix} = \begin{bmatrix} 26 & 7 \\ 7 & 26 \end{bmatrix} \begin{bmatrix} 1 \\ -1/2 \end{bmatrix} = \begin{bmatrix} 45/2 \\ -6 \end{bmatrix}$$

and

$$[C_1(z)]_+ = \tfrac{45}{2} - 6z^{-1}$$

Multiplying by $A_1^*(1/z^*) = 1 - 0.5z$ we have

$$[C_1(z)]_+ A_1^*(1/z^*) = \left(\tfrac{45}{2} - 6z^{-1}\right)(1 - 0.5z) = -\tfrac{45}{4}z + \tfrac{51}{2} - 6z^{-1}$$

Therefore, the causal part of $P_y(z)$ is

$$[P_y(z)]_+ = \left[[C_1(z)]_+ A_1^*(1/z^*)\right]_+ = \tfrac{51}{2} - 6z^{-1}$$

Using the symmetry of $P_y(z)$, we have

$$C_1(z)A_1^*(1/z^*) = B_1(z)B_1^*(1/z^*) = -6z + \tfrac{51}{2} - 6z^{-1}$$

Performing a spectral factorization gives

$$B_1(z)B_1^*(1/z^*) = 24(1 - \tfrac{1}{4}z^{-1})(1 - \tfrac{1}{4}z)$$

so the ARMA(1,1) model is

$$H(z) = 2\sqrt{6}\,\frac{1 - 0.25z^{-1}}{1 - 0.5z^{-1}}$$

Just as with the Padé approximation, the MYWE method only uses the values of $r_x(k)$ for $k = q, q+1, \ldots, q+p$ to estimate the coefficients $a_p(k)$. If $r_x(k)$ is unknown and must be estimated from the data, then the accuracy of the model will depend on how accurately $r_x(k)$ may be estimated. However, suppose that we may estimate $r_x(k)$ for $k > p+q$. This would certainly be possible, for example, if $N \gg p+q$ where N is the length of $x(n)$. How may these estimates be used to improve the accuracy of the coefficients $a_p(k)$? As we did in extending the Padé approximation method to Prony's method, we may consider the problem of finding the set of coefficients $a_p(k)$ that produces the "best fit," in a least squares sense, to a given set of autocorrelation values. For example, given the autocorrelation sequence $r_x(k)$ for $k = 0, 1, \ldots, L$, we may form the set of *extended Yule-Walker equations* by evaluating Eq. (4.135) for $k = 0, 1, \ldots, L$ as follows:

$$\begin{bmatrix} r_x(0) & r_x^*(1) & r_x^*(2) & \cdots & r_x^*(p) \\ r_x(1) & r_x(0) & r_x^*(1) & \cdots & r_x^*(p-1) \\ \vdots & \vdots & \vdots & & \vdots \\ r_x(q) & r_x(q-1) & r_x(q-2) & \cdots & r_x(q-p) \\ r_x(q+1) & r_x(q) & r_x(q-1) & \cdots & r_x(q-p+1) \\ \vdots & \vdots & \vdots & & \vdots \\ r_x(L) & r_x(L-1) & r_x(L-2) & \cdots & r_x(L-p) \end{bmatrix} \begin{bmatrix} 1 \\ a_p(1) \\ a_p(2) \\ \vdots \\ a_p(p) \end{bmatrix} = \begin{bmatrix} c_q(0) \\ c_q(1) \\ \vdots \\ c_q(q) \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{4.145}$$

From the last $L - q$ equations we have

$$\begin{bmatrix} r_x(q) & r_x(q-1) & \cdots & r_x(q-p+1) \\ r_x(q+1) & r_x(q) & \cdots & r_x(q-p+2) \\ \vdots & \vdots & & \vdots \\ r_x(L-1) & r_x(L-2) & \cdots & r_x(L-p) \end{bmatrix} \begin{bmatrix} a_p(1) \\ a_p(2) \\ \vdots \\ a_p(p) \end{bmatrix} = - \begin{bmatrix} r_x(q+1) \\ r_x(q+2) \\ \vdots \\ r_x(L) \end{bmatrix} \tag{4.146}$$

or

$$\boxed{\mathbf{R}_q \bar{\mathbf{a}}_p = -\mathbf{r}_{q+1}} \tag{4.147}$$

which is an overdetermined set of linear equations in the unknowns $a_p(k)$. As we did in Prony's method (p. 154), we may find the least squares solution, which is found by solving the equations

$$\boxed{\left(\mathbf{R}_q^H \mathbf{R}_q\right)\bar{\mathbf{a}}_p = -\mathbf{R}_q^H \mathbf{r}_{q+1}} \tag{4.148}$$

where $\left(\mathbf{R}_q^H \mathbf{R}_q\right)$ is a $p \times p$ Hermitian Toeplitz matrix whose elements are the autocorrelations of the sequence $r_x(k)$.

4.7.2 Autoregressive Models

A wide-sense stationary autoregressive process of order p is a special case of an ARMA(p, q) process in which $q = 0$. An AR(p) process may be generated by filtering unit variance white noise, $v(n)$, with an all-pole filter of the form

$$H(z) = \frac{b(0)}{1 + \sum_{k=1}^{p} a_p(k) z^{-k}} \tag{4.149}$$

Just as with an ARMA process, the autocorrelation sequence of an AR process satisfies the Yule-Walker equations, which are

$$\boxed{r_x(k) + \sum_{l=1}^{p} a_p(l) r_x(k-l) = |b(0)|^2 \delta(k) \quad ; \quad k \geq 0} \tag{4.150}$$

Writing these equations in matrix form for $k = 1, 2, \ldots p$, using the conjugate symmetry of $r_x(k)$, we have

$$\begin{bmatrix} r_x(0) & r_x^*(1) & r_x^*(2) & \cdots & r_x^*(p-1) \\ r_x(1) & r_x(0) & r_x^*(1) & \cdots & r_x^*(p-2) \\ r_x(2) & r_x(1) & r_x(0) & \cdots & r_x^*(p-3) \\ \vdots & \vdots & \vdots & & \vdots \\ r_x(p-1) & r_x(p-2) & r_x(p-3) & \cdots & r_x(0) \end{bmatrix} \begin{bmatrix} a_p(1) \\ a_p(2) \\ a_p(3) \\ \vdots \\ a_p(p) \end{bmatrix} = - \begin{bmatrix} r_x(1) \\ r_x(2) \\ r_x(3) \\ \vdots \\ r_x(p) \end{bmatrix} \tag{4.151}$$

Therefore, given the autocorrelations $r_x(k)$ for $k = 0, 1 \ldots, p$ we may solve Eq. (4.151) for the AR coefficients $a_p(k)$. This approach is referred to as the *Yule-Walker method.* If we compare Eq. (4.151) to the normal equations for all-pole modeling of a deterministic signal using Prony's method, Eq. (4.79), we see that the two sets of equations are identical. The only difference between them is in how the autocorrelation $r_x(k)$ is defined. In the all-pole Prony method, $r_x(k)$ is a deterministic autocorrelation, whereas in the Yule-Walker method $r_x(k)$ is a statistical autocorrelation. Finally, to determine the coefficient $b(0)$ in the AR model we may use the Yule-Walker equation for $k = 0$ as follows

$$|b(0)|^2 = r_x(0) + \sum_{k=1}^{p} a_p(k) r_x(k) \tag{4.152}$$

In most applications, the statistical autocorrelation $r_x(k)$ is unknown and must be estimated from a sample realization of the process. For example, given $x(n)$ for $0 \leq n < N$, we may estimate $r_x(k)$ using the sample autocorrelation

$$\hat{r}_x(k) = \frac{1}{N} \sum_{n=0}^{N-1} x(n) x(n-k) \tag{4.153}$$

However, note that once we replace the statistical autocorrelation $r_x(k)$ with this estimate, we have come full circle back to the autocorrelation method. Therefore, in spite of the important philosophical differences between deterministic and stochastic all-pole signal modeling, the two approaches become equivalent when the autocorrelation sequence must be estimated.

4.7.3 Moving Average Models

A moving average process is another special case of ARMA(p, q) process. An MA(q) process may be generated by filtering unit variance white noise $v(n)$ with an FIR filter of order q as follows:

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

The Yule-Walker equations relating the autocorrelation sequence to the filter coefficients $b_q(k)$ are

$$r_x(k) = b_q(k) * b_q^*(-k) = \sum_{l=0}^{q-|k|} b_q(l+|k|)b_q^*(l) \tag{4.154}$$

Note that, unlike the case for an autoregressive process, these equations are nonlinear in the model coefficients, $b_q(k)$. Therefore, even if the autocorrelation sequence were known exactly, finding the coefficients $b_q(k)$ may be difficult. Instead of attempting to solve the Yule-Walker equations directly, another approach is to perform a spectral factorization of the power spectrum $P_x(z)$. Specifically, since the autocorrelation of an MA(q) process is equal to zero for $|k| > q$, the power spectrum is a polynomial of the form

$$\begin{aligned} P_x(z) &= \sum_{k=-q}^{q} r_x(k)z^{-k} = B_q(z)B_q^*(1/z^*) \\ &= |b_q(0)|^2 \prod_{k=1}^{q}(1-\beta_k z^{-1}) \prod_{k=1}^{q}(1-\beta_k^* z) \end{aligned} \tag{4.155}$$

where

$$B_q(z) = \sum_{k=0}^{q} b_q(k)z^{-k}$$

Using the spectral factorization given in Eq. (3.102), $P_x(z)$ may also be factored as follows

$$P_x(z) = \sigma_0^2\, Q(z)\, Q^*(1/z^*) = \sigma_0^2 \prod_{k=1}^{q}(1-\alpha_k z^{-1}) \prod_{k=1}^{q}(1-\alpha_k^* z) \tag{4.156}$$

where $Q(z)$ is a minimum phase monic polynomial of degree q, i.e., $|\alpha_k| \leq 1$.[14] Comparing Eqs. (4.155) and (4.156) it follows that $b_q(0) = \sigma_0$ and that $Q(z)$ is the minimum phase version of $B_q(z)$ that is formed by replacing each zero of $B_q(z)$ that lies outside the unit circle with one that lies inside the unit circle at the conjugate reciprocal location. Thus, given the autocorrelation sequence of an MA(q) process, we may find a model for $x(n)$ as follows. From the autocorrelation sequence $r_x(k)$ we form the polynomial $P_x(z)$ and factor it into a product of a minimum phase polynomial, $Q(z)$, and a maximum phase polynomial $Q^*(1/z^*)$ as in Eq. (4.156). The process $x(n)$ may then be modeled as the output of the minimum phase FIR filter

$$H(z) = \sigma_0 Q(z) = \sigma_0 \sum_{k=0}^{q} q(k)z^{-k}$$

[14] Since there is nothing that prohibits $B_q(z)$ and thus $Q(z)$ from having zeros on the unit circle, here we are allowing the factors of $Q(z)$ to have roots on the unit circle.

driven by unit variance white noise. It should be pointed out, however, that this model is not unique. For example, we may replace any one or more factors $(1-\alpha_k z^{-1})$ of $Q(z)$ with factors of the form $(1-\alpha_k^* z)$. The following example illustrates how spectral factorization may be used to find a moving average model for a simple process.

Example 4.7.2 *Moving Average Model Using Spectral Factorization*

Consider the MA(1) process that has an autocorrelation sequence given by

$$r_x(k) = 17\delta(k) + 4\big[\delta(k-1) + \delta(k+1)\big]$$

The power spectrum is the second-order polynomial

$$P_x(z) = 17 + 4z^{-1} + 4z = z\big[4 + 17z^{-1} + 4z^{-2}\big]$$

Performing a spectral factorization of $P_x(z)$ we find

$$P_x(z) = z\big(4 + z^{-1}\big)\big(1 + 4z^{-1}\big) = \big(4 + z^{-1}\big)\big(4 + z\big)$$

Therefore, $x(n)$ may be modeled as the output of an FIR filter having a system function equal to either

$$H(z) = 4 + z^{-1}$$

or

$$H(z) = 1 + 4z^{-1}$$

As an alternative to spectral factorization, a moving average model for a process $x(n)$ may also be developed using Durbin's method [5]. This approach begins by finding a high-order all-pole model $A_p(z)$ for the moving average process. Then, by considering the coefficients of the all-pole model $a_p(k)$ to be a new "data set," the coefficients of a qth-order moving average model are determined by finding a qth-order all-pole model for the sequence $a_p(k)$. More specifically, let $x(n)$ be a moving average process of order q with

$$B_q(z) = \sum_{k=0}^{q} b_q(k) z^{-k}$$

so that

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

where $w(n)$ is white noise. Suppose that we find a pth-order all-pole model for $x(n)$ and that p is large enough so that[15]

$$B_q(z) \approx \frac{1}{A_p(z)} = \frac{1}{a_p(0) + \sum_{k=1}^{p} a_p(k) z^{-k}} \tag{4.157}$$

For example, if

$$B_1(z) = b(0) - b(1)z^{-1}$$

[15]Note that the coefficient that normally appears in the numerator of the all-pole model has been absorbed into the denominator, thereby making the coefficient $a_p(0)$ some value other than one, in general.

and $|b(0)| > |b(1)|$ then $1/B_1(z)$ may be expanded in a power series as follows

$$\frac{1}{B_1(z)} = \frac{1}{b(0) - b(1)z^{-1}} = \frac{1}{b(0)} \sum_{k=0}^{\infty} \beta^k z^{-k}$$

where

$$\beta = \frac{b(1)}{b(0)}$$

Therefore, if p is sufficiently large so that $\beta^p \approx 0$ then $B_1(z)$ may be approximated by the expansion

$$B_1(z) \approx \frac{b(0)}{1 + \beta z^{-1} + \cdots + \beta^p z^{-p}} \tag{4.158}$$

Once a high-order all-pole model for $x(n)$ has been found, it is then necessary to estimate the MA coefficients $b_q(k)$ from the all-pole coefficients $a_p(k)$. From Eq. (4.157) we see that since

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

then $1/B_q(z)$ represents a qth-order all-pole model for the "data" $a_p(k)$. The coefficients of the all-pole model for $a_p(k)$ are then taken as the coefficients of the moving average model. Thus, Durbin's method may be summarized as follows.

1. Given $x(n)$ for $n = 0, 1, \ldots, N-1$ or $r_x(k)$ for $k = 0, 1, \ldots, N-1$, a pth-order all-pole model is found and the coefficients are normalized, as in Eq. (4.157), by the gain (numerator coefficient) of the all-pole model. Typically, the model order p is chosen so that it is at least four times the order q of the moving average process [17].
2. Using the AR coefficients derived in Step 1 as data, a qth-order all-pole model for the sequence $a_p(k)$ is then found. The resulting coefficients, after dividing by the gain term, are the coefficients of the moving average model.

A MATLAB program for Durbin's method that uses the autocorrelation method for both all-pole modeling problems is given in Fig. 4.22.

Durbin's Method

```
function b = durbin(x,p,q)
%
   x   = x(:);
   if p>=length(x), error('Model order too large'), end
   [a,epsilon] = acm(x,p);
   [b,epsilon] = acm(a/sqrt(epsilon),q);
   b = b/sqrt(epsilon)
   end;
```

Figure 4.22 *A* MATLAB *program for finding the moving average coefficients of an all-zero model for a signal* $x(n)$ *using Durbin's method. Note that this program calls* `acm.m` *(see Appendix).*

Example 4.7.3 *The Durbin Algorithm*

To illustrate how the Durbin algorithm is used to find a moving average model, consider the signal $x(n)$ that has a z-transform

$$X(z) = 1 - \beta z^{-1}$$

where β is a real number with $|\beta| < 1$. The first step in Durbin's method requires that we find a high-order all-pole model for $x(n)$. Since

$$\frac{1}{X(z)} = \frac{1}{1 - \beta z^{-1}} = \sum_{k=0}^{\infty} \beta^k z^{-k}$$

then the all-pole model for $x(n)$ using the autocorrelation method with $p \gg 1$ is approximately

$$H(z) = \frac{1}{1 + \beta z^{-1} + \cdots + \beta^p z^{-p}}$$

The next step is to fit a qth-order all-pole model to the sequence $a_p(k) = \beta^k$. As we saw in Example 4.6.1 (p. 179) a first-order all-pole model for this sequence is

$$H(z) = \frac{\sqrt{\epsilon_1}}{1 + a(1)z^{-1}}$$

where

$$a(1) = -\beta \frac{1 - \beta^{2p}}{1 - \beta^{2(p+1)}}$$

and

$$\epsilon_1 = \frac{1 - \beta^{4p+2}}{1 - \beta^{2p+2}}$$

Therefore, assuming that $\beta^{2p} \ll 1$, so that $\epsilon_1 \approx 1$, the first-order moving average model is

$$B_1(z) = 1 - \beta \frac{1 - \beta^{2p}}{1 - \beta^{2(p+1)}} z^{-1}$$

Note that if, instead of the autocorrelation method in the second step, we were to use the covariance method, then we would have, as we saw in Example 4.6.2 (p. 184),

$$a(1) = -\beta$$

and the first-order moving average model would be

$$B_1(z) = 1 - \beta z^{-1}$$

4.7.4 Application: Power Spectrum Estimation

Spectrum estimation is an important application of stochastic signal modeling. As we saw in Section 3.3.8, the power spectrum of a wide-sense stationary random process $x(n)$ is

$$P_x(e^{j\omega}) = \sum_{k=-\infty}^{\infty} r_x(k) e^{-jk\omega} \tag{4.159}$$

where

$$r_x(k) = E\{x(n)x^*(n-k)\} \tag{4.160}$$

is the autocorrelation of $x(n)$. Since $r_x(k)$ is generally unknown, spectrum estimation is concerned with the estimation of $P_x(e^{j\omega})$ from a sample realization of $x(n)$ for $n = 0, 1, \ldots, N$.

One approach that may be used to estimate $P_x(e^{j\omega})$ is to estimate $r_x(k)$ from $x(n)$ and then use this estimate in Eq. (4.159). However, with only $N+1$ values of $x(n)$ the autocorrelation may only be estimated for lags $|k| \leq N$, and the power spectrum estimate would have the form

$$\hat{P}_x(e^{j\omega}) = \sum_{k=-N}^{N} \hat{r}_x(k)e^{-jk\omega} \tag{4.161}$$

This estimate is limited by two factors. First, since estimated autocorrelations are used instead of the true values, the accuracy of $\hat{P}_x(e^{j\omega})$ will be limited by the accuracy of the estimates $\hat{r}_x(k)$. Second, since $\hat{P}_x(e^{j\omega})$ does not include any estimates of $r_x(k)$ for $|k| > N$, the power spectrum estimate will be limited in resolution unless $r_x(k) \approx 0$ for $|k| > N$.

The estimation of $P_x(e^{j\omega})$ may be facilitated if something is known about the process $x(n)$ in addition to the signal values. For example, suppose that $x(n)$ is known to be an autoregressive process of order p. Since the power spectrum of an AR(p) process is

$$P_x(e^{j\omega}) = \frac{|b(0)|^2}{\left|1 + \sum_{k=1}^{p} a_p(k)e^{-jk\omega}\right|^2} \tag{4.162}$$

then we may use the Yule-Walker method with estimated autocorrelations to estimate the coefficients $a_p(k)$ and $b(0)$, and then use these estimates in Eq. (4.162) as follows:

$$\hat{P}_x(e^{j\omega}) = \frac{|\hat{b}(0)|^2}{\left|1 + \sum_{k=1}^{p} \hat{a}_p(k)e^{-jk\omega}\right|^2} \tag{4.163}$$

Assuming that the estimates of the model coefficients are sufficiently accurate, this approach may result in a significantly improved spectrum estimate.

Example 4.7.4 *AR Spectrum Estimation*

Shown in Fig. 4.23 are $N = 64$ samples of an AR(4) process that is generated by filtering unit variance white noise with the fourth-order all-pole filter

$$H(z) = \frac{b(0)}{1 + \sum_{k=1}^{4} a(k)z^{-k}}$$

where $b(0) = 1$ and

$$a(1) = 0.7348 \quad ; \quad a(2) = 1.8820 \quad ; \quad a(3) = 0.7057 \quad ; \quad a(4) = 0.8851$$

which corresponds to a filter having a pair of poles at $z = 0.98e^{\pm j\pi/2}$ and a pair of poles at $z = 0.96e^{\pm j5\pi/8}$. Estimating the autocorrelation sequence for $|k| < N$ using

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

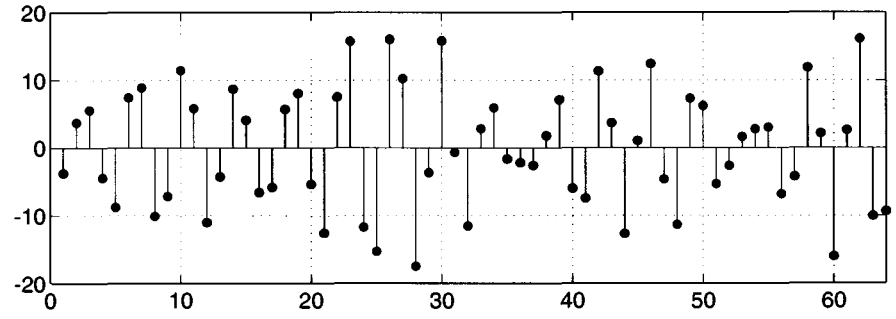


Figure 4.23 *An AR(4) random process.*

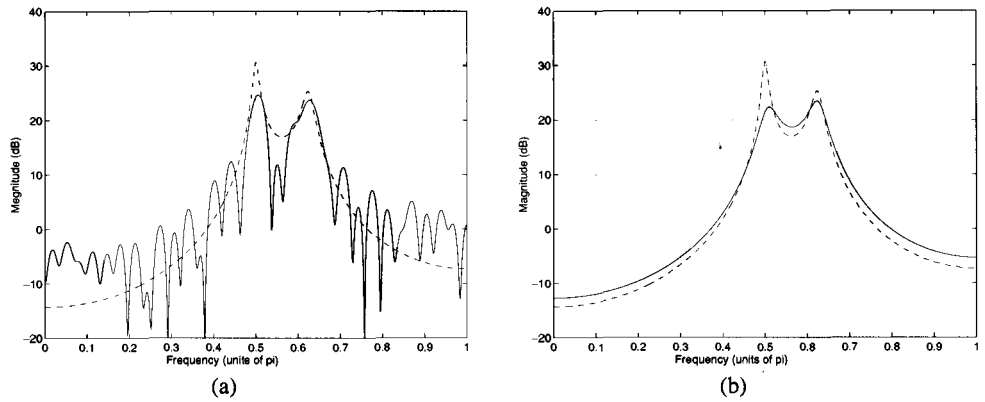


Figure 4.24 *Power spectrum estimation. (a) The estimate of the power spectrum by using estimated autocorrelations. (b) The estimate of the power spectrum using the Yule-Walker method. In both figures the true power spectrum is shown by the dashed line.*

and substituting these estimates into Eq. (4.161) we obtain the estimate of the spectrum shown in Fig. 4.24*a*.

On the other hand, using these estimates of $r_x(k)$ in the Yule-Walker equations, Eq. (4.151), we find the following estimates for $b(0)$ and $a(k)$,

$$b(0) = 1.4844 \;\; ; \;\; a(1) = 0.7100 \;\; ; \;\; a(2) = 1.6198 \;\; ; \;\; a(3) = 0.6193 \;\; ; \;\; a(4) = 0.6908$$

Incorporating these estimates into Eq. (4.162) gives the spectrum estimate shown in Fig. 4.24*b*.

As we will see in Chapter 8, there are two limitations with this model-based approach to spectrum estimation. First of all, it assumes that we know how $x(n)$ should be modeled. In Example 4.7.4, for example, it was assumed that we knew that $x(n)$ was an AR(p) process. If $x(n)$ is not an autoregressive process and cannot be sufficiently approximated as an AR process, then this approach will lead to an erroneous spectrum estimate. The second problem lies in the requirement that the order of the process be known. If p is unknown, then some criterion must be used to select an appropriate value for the model order. If this criterion does not produce a precise estimate for p, then $x(n)$ will either be overmodeled

(p too large) or undermodeled (p too small). Both cases will generally result in inaccurate spectrum estimates.

4.8 SUMMARY

In this chapter, we presented a number of different approaches to signal modeling. The first part of the chapter was concerned with modeling deterministic signals. Beginning with the least squares (direct) method, we found that this approach was only of limited use since it required finding the solution to a set of nonlinear equations. However, in those instances where it is necessary to find the true least squares solution, we saw that the method of iterative prefiltering could be used. We then looked at the Padé approximation, which only requires finding the solution to a set of linear equations and produces a model that exactly matches the data for the first $p + q + 1$ values, where p is the number of poles and q is the number of zeros in the model. In most applications, however, this property of perfect matching over a finite interval is overly restrictive and compromises the accuracy of the model for values of n outside this interval. We then developed Prony's method, which relaxes the perfect matching property of the Padé approximation and finds a model that is more accurate, on the average, for all values of n. As with the Padé approximation, Prony's method only requires finding the solution to a set of linear equations. After discussing Shanks' method, which modifies Prony's method in the way that the numerator coefficients are determined, we looked at the special case of all-pole signal modeling using Prony's method. The advantage of an all-pole model is that the normal equations are Toeplitz and may therefore be easily solved using the Levinson-Durbin recursion (Chapter 5). The Toeplitz structure also provides an advantage in terms of storage and reduces the amount of computations required to evaluate the autocorrelations in the normal equations. Finally, we considered the autocorrelation and covariance methods, which address the issue of how to determine the model when only a finite length data record is available. Here we saw that there is a tradeoff between computational efficiency and model accuracy. Specifically, whereas the autocorrelation method preserves the Toeplitz structure of the all-pole normal equations and guarantees a stable model, it requires that the data be windowed, which results in a decrease in the accuracy of the model. The covariance method, on the other hand, does not apply a window to the data and is therefore more accurate in its determination of the model coefficients, but the Toeplitz structure of the normal equations is lost along with the guaranteed stability of the model.

In the second part of the chapter, we considered briefly the problem of modeling a random process. Beginning with an ARMA process, we saw that the Yule-Walker equations could be used to find the coefficients of the model. In the case of an autoregressive process, the Yule-Walker equations are identical to the Prony all-pole normal equations. Therefore, with estimated autocorrelations, stochastic all-pole modeling is essentially equivalent to deterministic all-pole signal modeling. Finally, for a moving average process, since the Yule-Walker equations are nonlinear in the filter coefficients we looked at two methods for deriving the model parameters. The first involved a spectral factorization of the power spectrum of the process, whereas the second derived the model through a succession of all-pole models.

Before concluding this chapter it should be pointed out that an important problem in signal modeling has been, to this point, overlooked. This is the problem of model order

estimation. In each case that we have considered thus far, we have assumed that a model of a given order was to be found. In the absence of any information about the correct model order, it becomes necessary to estimate what an appropriate model order should be. As one might expect, misleading information and inaccurate models may result if an inappropriate model order is used. At this point, however, we will simply note that although a number of different model order estimation techniques are available, all of them are somewhat limited in terms of their robustness and accuracy (A more detailed discussion of model order estimation may be found in Chapter 8).

References

1. B. S. Atal and J. R. Remda, "A new model of LPC excitation for producing natural-sounding speech at low bit rates," *Proc. IEEE Int. Conf. on Acoust., Speech, Sig. Proc.*, Paris, pp. 614–617, May 1982.
2. F. Brophy and A. C. Salazar, "Considerations of the Padé approximant technique in the synthesis of recursive digital filters," *IEEE Trans. Audio and Electroacoust*, pp. 500–505, December 1973.
3. R. V. Churchill and J. W. Brown, *Introduction to Complex Variables and Applications*, 4th ed., McGraw-Hill, New York, 1984.
4. J. R. Deller, J. G. Proakis, and J. H. L. Hansen, *Discrete-time Processing of Speech Signals*, MacMillan, New York, 1993.
5. J. Durbin, "Efficient estimation of parameters in moving-average models," *Biometrica*, vol. 46, pp. 306–316, 1959.
6. J. H. McClellan, "Parametric Signal Modeling," Chapter 1 in *Advanced Topics in Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1988.
7. M. Morf, B. Dickenson, T. Kailath, and A. Vieira, "Efficient solution of covariance equations for linear prediction," *IEEE Trans. Acoust., Speech, Sig. Proc.*, vol. 25, pp. 429–433, October 1977.
8. A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1989.
9. H. E. Padé, "Sur la représentation approchée d'une fonction par des fractions rationelles," *Annales Scientifique de l'Ecole Normale Supérieure*, vol. 9, no. 3 (supplement), pp. 1–93, 1992.
10. G. R. B. Prony, "Essai expémental et analytique sur les lois de la dilatabilité de fluides elastiques et sur celles de la force expansion de la vapeur de l'alcool, à différentes températures," *Journal de l'Ecole Polytechnique* (Paris), vol. 1, no. 2, pp. 24–76, 1795.
11. L. R. Rabiner and R. W. Schafer, *Digital Processing of Speech Signals*, Prentice-Hall, Englewood Cliffs, NJ, 1978.
12. R. A. Roberts and C. T. Mullis, *Digital Signal Processing*, Addison Wesley, Reading, MA. 1987.
13. E. A. Robinson and S. Treitel, *Geophysical Signal Analysis*, Prentice-Hall, Englewood Cliffs, NJ, 1980.
14. J. L. Shanks, "Recursion filters for digital processing," *Geophysics*, vol. 32, pp. 33–51, February 1967.
15. K. Steiglitz and L. E. McBride, "A technique for the identification of linear systems," *IEEE Trans. on Automatic Control*, vol. AC-10, pp. 461–464, October 1965.
16. K. Steiglitz, "On the simultaneous estimation of poles and zeros in speech analysis," *IEEE Trans. Acoust., Speech, Sig. Proc.*, vol. 25, pp. 229–234, June 1977.
17. C. W. Therrien, *Discrete Random Signals and Statistical Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1992.
18. W. F. Trench, "An algorithm for the inversion of finite Toeplitz matrices," *J. SIAM*, vol. 12, no. 3, pp. 512–522, 1964.

4.9 PROBLEMS

4.1. Find the Padé approximation of second-order to a signal $x(n)$ that is given by

$$\mathbf{x} = [2,\ 1,\ 0,\ -1,\ 0,\ 1,\ 0,\ -1,\ 0,\ 1,\ \ldots]^T$$

i.e., $x(0) = 2$, $x(1) = 1$, $x(2) = 0$, and so on. In other words, using an approximation of the form

$$H(z) = \frac{b(0) + b(1)z^{-1} + b(2)z^{-2}}{1 + a(1)z^{-1} + a(2)z^{-2}}$$

find the coefficients $b(0)$, $b(1)$, $b(2)$, $a(1)$, and $a(2)$.

4.2. A third-order all-pole Padé approximation to a signal $x(n)$ has been found to be

$$H(z) = \frac{1}{1 + 2z^{-1} + z^{-2} + 3z^{-3}}$$

What information about $x(n)$ can be determined from this model?

4.3. Suppose that a signal $x(n)$ is known to be of the form

$$x(n) = \sum_{k=1}^{L} c_k(\lambda_k)^n u(n)$$

where the λ_k's are distinct complex numbers.

(a) Show that the Padé approximation method can be used to determine the parameters c_k and λ_k for $k = 1, 2, \ldots, L$. Is the answer unique?

(b) The first eight values of a signal $x(n)$, which is known to be of the form given above with $L = 3$, are

$$\mathbf{x} = [32,\ 16,\ 8,\ 12,\ 18,\ 33,\ 64.5,\ 128.25]^T$$

Determine c_k and λ_k for $k = 1, 2, 3$.

4.4. A consumer electronics device includes a DSP chip that contains a linear shift-invariant digital filter that is implemented in ROM. In order to perform some reverse engineering on the product, it is necessary to determine the system function of the filter. Therefore, the unit sample response is measured and it is determined that the first eight values of $h(n)$ are as listed in the following table.

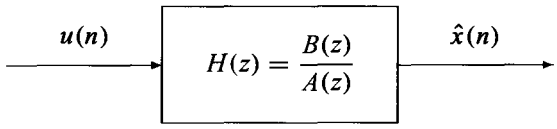
Unit sample response								
n	0	1	2	3	4	5	6	7
$h(n)$	−1	2	3	2	1	2	0	1

Having no knowledge of the order of the filter, it is assumed that $H(z)$ contains two poles and two zeros.

(a) Based on this assumption, determine a candidate system function, $H(z)$, for the filter.

(b) Based on the solution found in (a) and the given values for $h(n)$, is it possible to determine whether or not the hypothesis about the order of the system is correct? Explain.

4.5. The Padé approximation models a signal as the response of a filter to a unit sample input, $\delta(n)$. Suppose, however, that we would like to model a signal $x(n)$ as the *step response* of a filter as shown in the following figure



In the following, assume that $H(z)$ is a second-order filter having a system function of the form

$$H(z) = \frac{b(0) + b(1)z^{-1} + b(2)z^{-2}}{1 + a(1)z^{-1} + a(2)z^{-2}}$$

(a) Using the Padé approximation method with a *unit step* input, derive the set of equations that must be solved so that

$$\hat{x}(n) = x(n) \quad \text{for} \quad n = 0, 1, \ldots 4$$

(b) If the first eight values of $x(n)$ are

$$\mathbf{x} = [1,\ 0,\ 2,\ -1,\ 2,\ 0,\ 1,\ 2]^T$$

find $b(0)$, $b(1)$, $b(2)$, $a(1)$, and $a(2)$.

4.6. With a real-valued signal $x(n)$ known only for $n = 0, 1, \ldots, N$, the *backwards covariance method* finds the coefficients of the all-pole model that minimize the *backward prediction error*

$$\mathcal{E}_p^- = \sum_{n=p}^{N} [e_p^-(n)]^2$$

where

$$e_p^-(n) = x(n-p) + \sum_{k=1}^{p} a_p(k)x(n+k-p)$$

(a) Show that the coefficients $a_p(k)$ that minimize $\mathcal{E}_p^-$ satisfy a set of normal equations of the form

$$\mathbf{R}_x \bar{\mathbf{a}}_p = -\mathbf{r}_x$$

where

$$\bar{\mathbf{a}}_p = [a_p(1),\ a_p(2),\ \ldots\ , a_p(p)]^T$$

and find explicit expressions for the entries in $\mathbf{R}_x$ and $\mathbf{r}_x$.

(b) Is the solution to the backwards covariance method the same as the solution to the covariance method? Why or why not?

(c) Consider a new error that is the sum of the forward and backward prediction errors,

$$\mathcal{E}_p^B = \sum_{n=p}^{N} \left\{ [e_p^+(n)]^2 + [e_p^-(n)]^2 \right\}$$

where $e_p^-(n)$ is the backwards prediction error defined above, and $e_p^+(n)$ is the forward prediction error used in the covariance method,

$$e_p^+(n) = x(n) + \sum_{k=1}^{P} a_p(k)x(n-k)$$

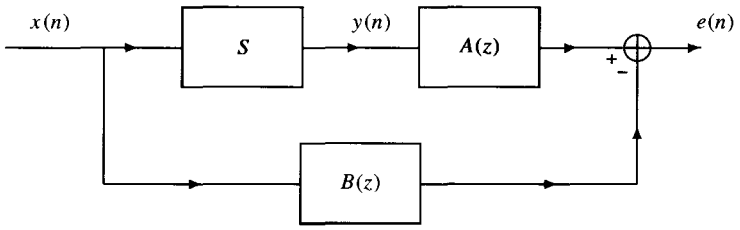
Derive the normal equations for the coefficients that minimize $\mathcal{E}_p^B$. (This approach is known as the *Modified Covariance Method.*)

(d) Consider the signal

$$x(n) = \beta^n \quad ; \quad n = 0, 1, \ldots, N$$

With $p = 1$ find the first-order all-pole model that minimizes $\mathcal{E}_p^B$ and determine the value of $\mathcal{E}_p^B$. For what values of β is the model stable? What happens to the model and the modeling error as $N \to \infty$?

4.7. Suppose that we would like to derive a rational model for an unknown system S using the approach shown in the following figure,



For a given input $x(n)$ the output of the system, $y(n)$, is observed. The coefficients of the two FIR filters $A(z)$ and $B(z)$ that minimize the sum of the squares of the error signal $e(n)$ are then to be determined. Assume that the sum is for all $n \geq 0$ as in Eq. (4.73).

(a) Derive the normal equations that define the optimal solution for the coefficients of $A(z)$ and $B(z)$.

(b) The philosophy of this method is that if the error is small then $B(z)/A(z)$ is a reasonable model for S. Suppose that S is a linear shift-invariant system with a rational system function. Show that this method will identify the parameters of S exactly assuming that the orders of the filters $A(z)$ and $B(z)$ are chosen appropriately [15].

4.8. Consider a signal, $x(n)$, which is the unit sample response of a causal all-pole filter with system function

$$H(z) = \frac{1}{(1 + 0.5z^{-1})(1 + 0.75z^{-1})(1 + 2z^{-1})}$$

We observe $x(n)$ over the interval $0 \leq n \leq N$ where $N \gg 1$.

(a) Using the covariance method, we determine a third-order all-pole model for $x(n)$. What, if anything, can you say about the location of the poles in the model? Do the pole locations depend on N? If so, where do the poles go as $N \to \infty$?

(b) Repeat part (a) for the case in which you use the autocorrelation method.

4.9. Equation (4.129) may be used to reduce the amount of computation required to set-up the covariance normal equations.

(a) Show that the elements along the main diagonal may be computed recursively beginning with $r_x(1, 1)$.

(b) Show how the elements along the lower diagonals may be computed recursively beginning with $r_x(k, 1)$. How may the terms along the upper diagonals be obtained?

(c) Determine how many multiplies and adds are necessary to set-up the covariance normal equations (do not forget the evaluation of the vector on the right-hand side).

4.10. We want to model a signal $x(n)$ using an all-pole model of the form

$$H(z) = \frac{b(0)}{1 + z^{-N}\left[\sum_{k=1}^{p} a_p(k) z^{-k}\right]}$$

For example, with $p = 2$ the model is

$$H(z) = \frac{b(0)}{1 + a(1)z^{-N-1} + a(2)z^{-N-2}}$$

Derive the normal equations that define the coefficients $a_p(k)$ that minimize the Prony error

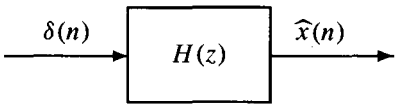
$$\mathcal{E}_p = \sum_{n=0}^{\infty} |e(n)|^2$$

where

$$e(n) = x(n) + \sum_{l=1}^{p} a_p(l) x(n - l - N)$$

and derive an expression for the minimum error.

4.11. Suppose that we would like to model a signal $x(n)$ as shown in the following figure.



where $h(n)$ is an all-pole filter that has a system function of the form

$$H(z) = \frac{b(0)}{1 + \sum_{k=1}^{p} a_p(k) z^{-2k}}$$

Modify the Prony normal equations so that one can determine the coefficients $a_p(k)$ in $H(z)$ from a sequence of signal values, $x(n)$.

4.12. Suppose that we would like to model a signal $x(n)$ that we believe to be quasiperiodic. Based on our observations of $x(n)$ we estimate the autocorrelations through lag $k = 10$ to be

$$r_x(k) = [1.0,\ 0.4,\ 0.4,\ 0.3,\ 0.2,\ 0.9,\ 0.4,\ 0.4,\ 0.2,\ 0.1,\ 0.7]^T$$

(a) In formulating an all-pole model to take into account the suspected periodicity let us

consider a two-coefficient model of the form

$$H(z) = \frac{b(0)}{1 + a(5)z^{-5} + a(10)z^{-10}}$$

Find the values for the coefficients $a(5)$ and $a(10)$ that minimize the all-pole modeling error.

(b) Compare the error obtained with the model found in (a) to the error that is obtained with a model of the form

$$H(z) = \frac{b(0)}{1 + a(1)z^{-1} + a(2)z^{-2}}$$

(c) Now consider an all-pole model of the form

$$H(z) = \frac{b(0)}{1 + a(N)z^{-N}}$$

where both $a(N)$ and N are considered to be model parameters. Find the value for $a(N)$ and N that minimize the all-pole modeling error and evaluate the modeling error.

4.13. We would like to build a predictor of digital waveforms. Such a system would form an estimate of a later sample (say n_0 samples later) by observing p consecutive data samples. Thus we would set

$$\hat{x}(n + n_0) = \sum_{k=1}^{p} a_p(k)x(n - k)$$

The predictor coefficients $a_p(k)$ are to be chosen to minimize

$$\mathcal{E}_p = \sum_{n=0}^{\infty} [x(n + n_0) - \hat{x}(n + n_0)]^2$$

(a) Derive the equations that define the optimum set of coefficients $a_p(k)$.

(b) If $n_0 = 0$, how is your formulation of this problem different from Prony's method?

4.14. You are told that it is always possible to determine whether or not a causal all-pole filter is stable from a finite number of values of its unit sample response. For example, if $H(z)$ is a pth-order all-pole filter, given $h(n)$ for $n = 0, 1, \ldots, N$, then the stability of $H(z)$ may be determined. If this is true, explain the procedure and list any conditions that must be placed on p or N. If false, explain why it cannot be done.

4.15. Let $H(z)$ be a first-order model for a real-valued signal $x(n)$,

$$H(z) = \frac{b(0)}{1 - a(1)z^{-1}}$$

and let

$$\mathcal{E}_{LS} = \sum_{n=0}^{N-1} [x(n) - h(n)]^2$$

be the error that is to be minimized. By setting the derivatives of $\mathcal{E}_{LS}$ with respect to $b(0)$ and $a(1)$ equal to zero, try to find an analytic solution for the values of $b(0)$ and $a(1)$ that minimize $\mathcal{E}_{LS}$. (This problem illustrates how difficult the direct method of signal modeling may be, even for a first-order model.)

4.16. We have a signal $x(n)$ for which we would like to obtain an all-pole model of the form

$$H(z) = \frac{b(0)}{1 + a(1)z^{-1} + a(2)z^{-2}}$$

Using the autocorrelation method, find explicit formulas for $b(0)$, $a(1)$, and $a(2)$ in terms of $r_x(0)$, $r_x(1)$, and $r_x(2)$.

4.17. If one is modeling a signal $x(n)$ whose transform, $X(z)$, contains zeros, then an all-pole model may be used to effectively model a zero with an infinite number of poles. In this problem we look at how a zero is modeled with the autocorrelation method. Let $x(n) = \delta(n) - \alpha\delta(n-1)$ where $|\alpha| < 1$ and α is real.

(a) Determine the *pth*-order all-pole model $A_p(z)$ for $x(n)$ where p is an arbitrary positive integer, and find the value of the squared error $\mathcal{E}_p$.

(b) For the all-pole model determined in part (a), what is the limit of $A_p(z)$ as $p \rightarrow \infty$? What does $\mathcal{E}_p$ converge to as $p \rightarrow \infty$? Justify your answers.

(c) Repeat parts (a) and (b) for $|\alpha| > 1$.

4.18. Find a closed-form expression for the FIR least squares inverse filter of length N for each of the following systems.

1. $G(z) = \dfrac{1}{1 - \alpha z^{-1}}$; $|\alpha| < 1$.
2. $G(z) = 1 - z^{-1}$.
3. $G(z) = \dfrac{\alpha - z^{-1}}{1 - \alpha z^{-1}}$; $|\alpha| < 1$.

4.19. An important application of least squares inverse filtering is deconvolution, which is concerned with the recovery of a signal $d(n)$ that has been convolved with a filter $g(n)$

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

The problem is to design a filter $h_N(n)$ that may be used to produce an estimate of $d(n)$ from $x(n)$,

$$\hat{d}(n) = x(n) * h_N(n)$$

One of the difficulties, however, is that noise in the observed signal may be amplified by the filter. For example, if we observe

$$y(n) = d(n) * g(n) + v(n)$$

then the filtered observations become

$$y(n) * h_N(n) = \hat{d}(n) + v(n) * h_N(n) = \hat{d}(n) + u(n)$$

where

$$u(n) = v(n) * h_N(n)$$

is the filtered noise. One way to reduce this noise is to design a least squares inverse filter that minimizes

$$\mathcal{E} = \sum_{n=0}^{\infty} |e(n)|^2 + \lambda E\{|u(n)|^2\}$$

where

$$e(n) = \delta(n - n_0) - h_N(n) * g(n)$$

and $\lambda > 0$ is a parameter that is to be selected. Note that for large values of λ, minimizing $\mathcal{E}$ will force a large reduction in the filtered noise at the expense of a decrease in resolution, i.e., larger $e(n)$, whereas smaller values of λ lead to higher resolution and larger noise.

(a) Assume that $v(n)$ is zero-mean white noise with a variance σ_v^2. Show that

$$E\{|u(n)|^2\} = \sigma_v^2 \mathbf{h}_N^H \mathbf{h}_N$$

where $\mathbf{h}_N$ is a vector containing the coefficients of the filter $h_N(n)$.

(b) Derive the normal equations that result from minimizing the error

$$\mathcal{E} = \mathbf{e}^H \mathbf{e} + \lambda \sigma_v^2 \mathbf{h}_N^H \mathbf{h}_N$$

where $\mathbf{e} = [e(0),\ e(1),\ \ldots]^T$, and show that they may be written in the form

$$(\mathbf{R}_g + \alpha \mathbf{I})\mathbf{h}_N = \mathbf{g}_{n_0}^*$$

where $\alpha > 0$ is a *prewhitening parameter* that depends upon the values of λ, and $\mathbf{g}_{n_0}^*$ is the vector on the right-side of Eq. (4.101).

4.20. We are given a signal, $x(n)$, that we want to model as the unit sample response of an all-pole filter. We have reason to believe that the signal is periodic and, consequently, the poles of the model should lie on the unit circle. Thus, assuming a second-order model for the signal, the system function is constrained to have the form

$$H(z) = \frac{b(0)}{1 + a(1)z^{-1} + z^{-2}}$$

With $|a(1)| < 2$ this model produces a pair of poles on the unit circle at an angle θ defined by

$$2\cos\theta = -a(1)$$

(a) Using the autocorrelation method, derive the normal equations that define the value of $a(1)$ that minimizes the error

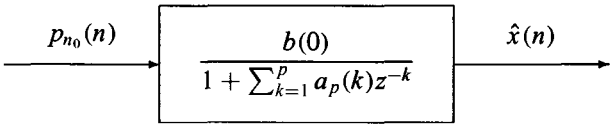
$$\mathcal{E}_p = \sum_{n=0}^{\infty} e^2(n)$$

(b) Find an expression for the minimum error, $\{\mathcal{E}_p\}_{\min}$.

4.21. Voiced speech may be modeled as the output of an all-pole filter driven by an impulse train

$$p_{n_0}(n) = \sum_{k=1}^{K} \delta(n - kn_0)$$

where the time between pulses, n_0, is known as the *pitch period.* Suppose that we have a segment of voiced speech and suppose that we know the pitch period, n_0. We extract a subsequence, $x(n)$, of length $N = 2n_0$ and model this signal as shown in the following figure



where the input, $p_{n_0}(n)$, consists of two pulses,

$$p_{n_0}(n) = \delta(n) + \delta(n - n_0)$$

Find the normal equations that define the coefficients $a_p(k)$ that minimize the error

$$\mathcal{E}_p = \sum_{n=0}^{N} e^2(n)$$

where

$$e(n) = a_p(n) * x(n) - b(n) * p_{n_0}(n)$$

and $b(n) = b(0)\delta(n)$.

4.22. You would like to design a linear predictor of a signal $x(n)$ but, due to hardware constraints, are limited to a two-tap predictor. However, since delays can be tolerated in the predictor, you decide to design a predictor of the form

$$\hat{x}(n) = a(1)x(n - N_1) + a(2)x(n - N_2)$$

where N_1 and N_2 are positive integers. The goal is to find the values of $a(1)$, $a(2)$, N_1, and N_2 that minimize the mean-square error $E\{e^2(n)\}$ where

$$e(n) = x(n) - \hat{x}(n)$$

Assuming that $x(n)$ is a zero mean wide-sense stationary process, you estimate the autocorrelation of $x(n)$ and find that the values of $r_x(k)$ for $k = 0, 1, \ldots, 7$ are as given in the following table. For $k > 7$, it is determined that the autocorrelation is approximately zero.

Autocorrelation values								
k	0	1	2	3	4	5	6	7
$r_x(k)$	1.0	−0.1	0.0	−0.5	−0.2	0.6	0.2	0.2

(a) If you were to design an optimum predictor of the form $\hat{x}(n) = a(1)x(n - 1)$, what would be the mean-square error in your prediction of $x(n)$? What about for the predictor $\hat{x}(n) = a(1)x(n - 3)$?

(b) Derive a general expression for the minimum mean-square error for a predictor of the form $\hat{x}(n) = a(1)x(n - N_1)$ with your expression given only in terms of $r_x(k)$. What value of N_1 minimizes the mean-square error?

(c) Find the values of $a(1)$, $a(2)$, N_1, and N_2 in the two-coefficient linear predictor defined above that minimize the mean-square error $E\{e^2(n)\}$.

(d) Find the value for the mean-square prediction error for your predictor designed in part (c).

4.23. If $r_x(0) = 1$, $r_x(1) = 0.5$, and $r_x(2) = 0.75$, find the values of $a(1)$, $a(2)$, and $b(0)$ in the following AR(2) model for $x(n)$,

$$x(n) + a(1)x(n-1) + a(2)x(n-2) = b(0)w(n)$$

where $w(n)$ is unit variance white noise.

4.24. Use the method of spectral factorization to find a moving average model of order 2 for a process whose autocorrelation sequence is

$$\mathbf{r}_x = [3,\ 1.5,\ 1]^T$$

4.25. Suppose that the first five values in the autocorrelation sequence for the process $x(n)$ are

$$\mathbf{r}_x = [3,\ 9/4,\ 9/8,\ 9/16,\ 9/32\ \ldots]^T$$

(a) Use the modified Yule-Walker equation method to find an ARMA(1,1) model for $x(n)$.

(b) Are the given values in the autocorrelation sequence consistent with the model that you found in part (a)?

Computer Exercises

C4.1. In this problem, you are to consider the design of lowpass filters using the Padé and Prony methods.

(a) Suppose you would like to design a lowpass filter with a cutoff frequency $\omega_c = \pi/2$. Using the Padé approximation with $p + q + 1 = 20$, compare the designs that result when $p = 0, 2, 4, \ldots, 20$. Which design is the best? (Note: one of the parameters that you will need to experiment with in your design is the delay, n_0, of the ideal unit sample response.)

(b) Repeat part (a) for a narrowband lowpass filter that has a cutoff frequency $\omega_c = \pi/16$. Which design is the best and how do these filters compare to those designed in part (a)?

(c) Repeat part (a) using Prony's method and compute the minimum error $\epsilon_{p,q}$ for each filter. Which filter is the best? Compare your designs to those obtained using Padé approximation.

(d) Using the m-file `ellip.m` in the Signal Processing Toolbox, design a tenth-order elliptic lowpass filter with a cutoff frequency $\omega_c = \pi/2$, and evaluate the Prony error. Compare this filter to the best Prony filter found in part (c) and explain what you observe.

C4.2. We have seen that the direct method of signal modeling leads to a set of nonlinear equations that need to be solved for the model coefficients. Iterative prefiltering, however, is an approach that may be used to avoid having to solve these nonlinear equations. In this exercise we look at the method of iterative prefiltering and compare it to Prony's method.

(a) Let

$$H(z) = \frac{1 - 0.9z^{-1} + 0.81z^{-2}}{1 - 1.978z^{-1} + 2.853z^{-2} - 1.877z^{-3} + 0.9036z^{-4}}$$

be the system function of a linear shift-invariant system. Generate the first 100 samples of the unit sample response $h(n)$ of this filter.

(b) Using the method of iterative prefiltering, find a two-zero, four-pole model for $h(n)$. How many iterations are required for the coefficients to converge? What happens if $h(n)$ is over-modeled using $p = q = 4$? What about $p = q = 5$?

(c) The model found in part (b) assumes exact measurements of the unit sample response $h(n)$. Suppose that the measurements of $h(n)$ are noisy and you observe

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

where $v(n)$ is white Gaussian noise with a variance σ_v^2. Repeat part (b) using these noisy measurements with $\sigma_v^2 = 0.0001, 0.001, 0.01$. Comment on the accuracy of your models and the sensitivity of the coefficients to the noise variance.

(d) Repeat parts (b) and (c) using Prony's method and compare your results with those obtained using iterative prefiltering. Which method works the best?

C4.3. In this problem, we look briefly at the problem of deconvolution using FIR least squares inverse filters. Suppose that we have recorded a signal, $y(n)$, that is known to have been blurred by a filter having a unit sample response

$$g(n) = \begin{cases} \cos(0.2[n-25])\exp\{-0.01[n-25]^2\} & ; \quad 0 \le n \le 50 \\ 0 & ; \quad \text{otherwise} \end{cases}$$

The signal that is to be recovered from $y(n)$ is a sequence of impulses,

$$x(n) = \sum_{k=1}^{10} x(k)\delta(n - n_k)$$

where the values of $x(k)$ and n_k are as listed in the following table.

n_k	25	40	55	65	85	95	110	130	140	155
$x(k)$	1	0.8	0.7	0.5	0.7	0.2	0.9	0.5	0.6	0.3

(a) Make a plot the observed signal $y(n) = g(n) * x(n)$ and determine how accurately the amplitudes and locations of the impulses in $x(n)$ may be estimated by simply looking at the peaks of $y(n)$.

(b) Using the m-file `spike.m`, design the least squares inverse filter $h_N(n)$ of length $N = 50$ that has the optimum spiking delay.

(c) Filter $y(n)$ with your optimum spiking filter and plot the output of the filter $\hat{x}(n) = h_N(n) * y(n)$. What are your estimated values for the amplitudes and locations of the impulses in $x(n)$?

(d) Your results in part (c) assume noise-free observations of $y(n) = g(n) * x(n)$. Suppose these measurements are noisy,

$$\tilde{y}(n) = g(n) * x(n) + v(n)$$

where $v(n)$ is white Gaussian noise with variance σ_v^2. Repeat part (c) using $\tilde{y}(n)$ with $\sigma_v^2 = .0001$ and $\sigma_v^2 = .001$ and comment on the accuracy of your estimates of $x(k)$ and n_k.

(e) As discussed in Problem 4.19, the effect of measurement noise may be reduced by incorporating a *prewhitening parameter* α in the design of the least squares inverse filter. Write a MATLAB m-file or modify `spike.m` to allow for noise reduction in the least squares inverse filter design. Using this m-file, repeat your experiments in part (d) using different values for the prewhitening parameter. Comment of the effectiveness of α in reducing the noise. What values for α seem to work the best?

(f) Your results in parts (b) and (c) assume perfect knowledge of $g(n)$. Repeat the design of your least squares inverse filter assuming that $g(n)$ has been measured in the presence of noise, i.e., you are given

$$\hat{g}(n) = g(n) + w(n)$$

where $w(n)$ is white noise that is uniformly distributed between $[-.005, .005]$. Filter $y(n)$ with your optimum spiking filter and plot the output of the filter $y\hat{x}(n) = h_N(n) * y(n)$. How accurate are your estimates of the amplitudes and locations of the impulses in $x(n)$?

C4.4. In this exercise, we consider the problem of finding a moving average model for a signal $x(n)$.

(a) In Fig. 4.22 is an m-file to find the moving average coefficients $b_q(k)$ of a process $x(n)$ using Durbin's method. Write an m-file to find these coefficients using the method of spectral factorization.

(b) Generate $N = 256$ samples of the process

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

where $w(n)$ is unit variance white Gaussian noise. Using the method of spectral factorization, find a second-order moving average model for $x(n)$.

(c) Repeat part (b) using Durbin's method. Compare your results with the method of spectral factorization and discuss how the accuracy of your model is affected by the order of the all-pole model used in the first step of Durbin's method.

(d) Modify the m-file `durbin.m` by replacing the autocorrelation method in the second step of Durbin's method with the covariance method. Find a second-order moving average model for $x(n)$ and compare your results with those obtained in part (b). Repeat for other moving average processes and discuss your findings.

(e) Replace the autocorrelation method in both steps of Durbin's method with the covariance method and repeat part (d).

C4.5. In this exercise, we consider the problem of finding an autoregressive moving average model for a process $x(n)$.

(a) Write an m-file to find an ARMA(p, q) model for a process $x(n)$ using the modified Yule-Walker equation method, given the autocorrelation sequence $r_x(k)$.

(b) Generate 100 samples of an ARMA(4,2) process $x(n)$ by filtering unit-variance white Gaussian noise with

$$H(z) = \frac{1 - .9z^{-1} + 0.81z^{-2}}{1 - 1.978z^{-1} + 2.853z^{-2} - 1.877z^{-3} + 0.904z^{-4}}$$

and make a plot of the power spectrum of $x(n)$.

(c) Using your m-file for the modified Yule-Walker equation method in part (a), find an ARMA(4,2) model for $x(n)$. Compare your model with the coefficients of $H(z)$. Repeat for ten different realizations of the process $x(n)$ and examine the consistency of your model coefficients. Are your estimates of $a_p(k)$ and $b_q(k)$ close to the correct values, on the average? How much variation is there from one realization to the next?

THE LEVINSON RECURSION 5

5.1 INTRODUCTION

In Chapter 4, we saw that several different signal modeling problems require finding the solution to a set of linear equations of the form

$$\mathbf{R}_x \mathbf{a}_p = \mathbf{b} \tag{5.1}$$

where $\mathbf{R}_x$ is a Toeplitz matrix. In the Padé approximation method, for example, the denominator coefficients $a_p(k)$ which are represented by the vector $\mathbf{a}_p$ are found by solving a set of Toeplitz equations of the form (5.1) where $\mathbf{R}_x$ is a *non-symmetric* Toeplitz matrix containing the signal values $x(q), x(q+1), \ldots, x(q+p-1)$ in the first column and the signal values $x(q), x(q-1), \ldots, x(q-p+1)$ in the first row. In addition, the vector $\mathbf{b}$ contains the signal values $x(q+1), x(q+2), \ldots, x(q+p)$ and is therefore tightly constrained by the values in the matrix $\mathbf{R}_x$. A similar set of linear equations is also found in the modified Yule-Walker equations used in modeling an ARMA process. We saw that Toeplitz equations also arise in all-pole modeling of deterministic signals using either Prony's method or the autocorrelation method and in all-pole modeling of stochastic processes using the Yule-Walker method. Unlike the Padé approximation, however, in these cases $\mathbf{R}_x$ is a *Hermitian* Toeplitz matrix of autocorrelation values $r_x(0), r_x(1), \ldots, r_x(p-1)$. In addition, since $\mathbf{b} = -[r_x(1), \ldots, r_x(p)]^T$, the vector on the right side of Eq. (5.1) is again tightly constrained by the values in the Toeplitz matrix $\mathbf{R}_x$. In Shanks' method for finding the numerator coefficients, we again find a set of Hermitian Toeplitz equations. In this case, however, unlike the previous examples, the vector $\mathbf{b}$ is not constrained by the values in the matrix $\mathbf{R}_x$. In Chapter 7, Toeplitz equations will again be encountered when we consider the design of FIR Wiener filters. As in Shanks' method, $\mathbf{R}_x$ will be a Hermitian Toeplitz matrix but the vector $\mathbf{b}$ will be, in general, independent of the values in $\mathbf{R}_x$.

Due to the importance of solving Toeplitz equations in a variety of different problems, in this chapter, we look at efficient algorithms for solving these equations. In the process of deriving these algorithms, we will also discover a number of interesting properties of the solutions to these equations and will gain some insight into how other approaches to signal modeling may be developed. We begin, in Section 5.2, with the derivation of the Levinson-Durbin recursion. This recursion may be used to solve the Prony all-pole normal equations and the autocorrelation normal equations. The Levinson-Durbin recursion will also lead us

to several interesting results including the lattice filter structure, the Schur-Cohn stability test for digital filters, the Cholesky decomposition of a Toeplitz matrix, and a procedure for recursively computing the inverse of a Toeplitz matrix. In Section 5.3, we develop the Levinson recursion for solving a general set of Hermitian Toeplitz equations in which the vector $\mathbf{b}$ is unconstrained. The Levinson recursion may be used in Shanks' method, and it may be used to solve the general FIR Wiener filtering problem developed in Chapter 7. Finally, in Section 5.4 we derive the split Levinson recursion. This recursion is slightly more efficient than the Levinson-Durbin recursion and introduces the idea of *singular predictor polynomials* and *line spectral pairs* that are of interest in speech processing applications.

5.2 THE LEVINSON-DURBIN RECURSION

In 1947, N. Levinson presented a recursive algorithm for solving a general set of linear symmetric Toeplitz equations $\mathbf{R}_x\mathbf{a} = \mathbf{b}$. Appearing in an expository paper on the Wiener linear prediction problem, Levinson referred to the algorithm as a "mathematically trivial procedure" [16]. Nevertheless, this recursion has led to a number of important discoveries including the lattice filter structure, which has found widespread application in speech processing, spectrum estimation, and digital filter implementations. Later, in 1961, Durbin improved the Levinson recursion for the special case in which the right-hand side of the Toeplitz equations is a unit vector [7]. In this section we develop this algorithm, known as the *Levinson-Durbin recursion.* In addition, we will explore some of the properties of the recursion, show how it leads to a lattice filter structure for digital filtering, and prove that that the all-pole model derived from the autocorrelation method is stable.

5.2.1 Development of the Recursion

All-pole modeling using Prony's method or the autocorrelation method requires that we solve the normal equations which, for a pth-order model, are

$$r_x(k) + \sum_{l=1}^{p} a_p(l) r_x(k-l) = 0 \quad ; \quad k = 1, 2, \ldots, p \tag{5.2}$$

where the modeling error is

$$\epsilon_p = r_x(0) + \sum_{l=1}^{p} a_p(l) r_x(l) \tag{5.3}$$

Combining Eqs. (5.2) and (5.3) into matrix form we have

$$\begin{bmatrix} r_x(0) & r_x^*(1) & r_x^*(2) & \cdots & r_x^*(p) \\ r_x(1) & r_x(0) & r_x^*(1) & \cdots & r_x^*(p-1) \\ r_x(2) & r_x(1) & r_x(0) & \cdots & r_x^*(p-2) \\ \vdots & \vdots & \vdots & & \vdots \\ r_x(p) & r_x(p-1) & r_x(p-2) & \cdots & r_x(0) \end{bmatrix} \begin{bmatrix} 1 \\ a_p(1) \\ a_p(2) \\ \vdots \\ a_p(p) \end{bmatrix} = \epsilon_p \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{5.4}$$

which is a set of $p+1$ linear equations in the $p+1$ unknowns $a_p(1), a_p(2), \ldots, a_p(p)$ and ϵ_p. Equivalently, Eq. (5.4) may be written as

$$\boxed{\mathbf{R}_p\mathbf{a}_p = \epsilon_p\mathbf{u}_1} \tag{5.5}$$

where $\mathbf{R}_p$ is a $(p+1) \times (p+1)$ Hermitian Toeplitz matrix and $\mathbf{u}_1 = [1, 0, \ldots, 0]^T$ is a unit vector with 1 in the first position. In the special case of real data, the $\mathbf{R}_x$ is a symmetric Toeplitz matrix.

The Levinson-Durbin recursion for solving Eq. (5.5) is an algorithm that is recursive in the model order. In other words, the coefficients of the $(j+1)$st-order all-pole model, $\mathbf{a}_{j+1}$, are found from the coefficients of the j-pole model, $\mathbf{a}_j$. We begin, therefore, by showing how the solution to the jth-order normal equations may be used to derive the solution to the $(j+1)$st-order equations. Let $a_j(i)$ be the solution to the jth-order normal equations

$$\begin{bmatrix} r_x(0) & r_x^*(1) & r_x^*(2) & \cdots & r_x^*(j) \\ r_x(1) & r_x(0) & r_x^*(1) & \cdots & r_x^*(j-1) \\ r_x(2) & r_x(1) & r_x(0) & \cdots & r_x^*(j-2) \\ \vdots & \vdots & \vdots & & \vdots \\ r_x(j) & r_x(j-1) & r_x(j-2) & \cdots & r_x(0) \end{bmatrix} \begin{bmatrix} 1 \\ a_j(1) \\ a_j(2) \\ \vdots \\ a_j(j) \end{bmatrix} = \begin{bmatrix} \epsilon_j \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{5.6}$$

which, in matrix notation is

$$\mathbf{R}_j \mathbf{a}_j = \epsilon_j \mathbf{u}_1 \tag{5.7}$$

Given $\mathbf{a}_j$, we want to derive the solution to the $(j+1)$st-order normal equations,

$$\mathbf{R}_{j+1} \mathbf{a}_{j+1} = \epsilon_{j+1} \mathbf{u}_1 \tag{5.8}$$

The procedure for doing this is as follows. Suppose that we append a zero to the vector $\mathbf{a}_j$ and multiply the resulting vector by $\mathbf{R}_{j+1}$. The result is

$$\begin{bmatrix} r_x(0) & r_x^*(1) & r_x^*(2) & \cdots & r_x^*(j) & r_x^*(j+1) \\ r_x(1) & r_x(0) & r_x^*(1) & \cdots & r_x^*(j-1) & r_x^*(j) \\ r_x(2) & r_x(1) & r_x(0) & \cdots & r_x^*(j-2) & r_x^*(j-1) \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ r_x(j) & r_x(j-1) & r_x(j-2) & \cdots & r_x(0) & r_x^*(1) \\ r_x(j+1) & r_x(j) & r_x(j-1) & \cdots & r_x(1) & r_x(0) \end{bmatrix} \begin{bmatrix} 1 \\ a_j(1) \\ a_j(2) \\ \vdots \\ a_j(j) \\ 0 \end{bmatrix} = \begin{bmatrix} \epsilon_j \\ 0 \\ 0 \\ \vdots \\ 0 \\ \gamma_j \end{bmatrix} \tag{5.9}$$

where the parameter γ_j is

$$\boxed{\gamma_j = r_x(j+1) + \sum_{i=1}^{j} a_j(i) r_x(j+1-i)} \tag{5.10}$$

Note that if $\gamma_j = 0$, then the right side of Eq. (5.9) is a scaled unit vector and $\mathbf{a}_{j+1} = [1, a_j(1), \ldots, a_j(j), 0]^T$ is the solution to the $(j+1)$st-order normal equations (5.8). In general, however, $\gamma_j \neq 0$ and $[1, a_j(1), \ldots, a_j(j), 0]^T$ is not the solution to Eq. (5.8).

The key step in the derivation of the Levinson-Durbin recursion is to note that the Hermitian Toeplitz property of $\mathbf{R}_{j+1}$ allows us to rewrite Eq. (5.9) in the equivalent form

$$\begin{bmatrix} r_x(0) & r_x(1) & r_x(2) & \cdots & r_x(j) & r_x(j+1) \\ r_x^*(1) & r_x(0) & r_x(1) & \cdots & r_x(j-1) & r_x(j) \\ r_x^*(2) & r_x^*(1) & r_x(0) & \cdots & r_x(j-2) & r_x(j-1) \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ r_x^*(j) & r_x^*(j-1) & r_x^*(j-2) & \cdots & r_x(0) & r_x(1) \\ r_x^*(j+1) & r_x^*(j) & r_x^*(j-1) & \cdots & r_x^*(1) & r_x(0) \end{bmatrix} \begin{bmatrix} 0 \\ a_j(j) \\ a_j(j-1) \\ \vdots \\ a_j(1) \\ 1 \end{bmatrix} = \begin{bmatrix} \gamma_j \\ 0 \\ 0 \\ \vdots \\ 0 \\ \epsilon_j \end{bmatrix} \tag{5.11}$$

Taking the complex conjugate of Eq. (5.11) and combining the resulting equation with Eq. (5.9), it follows that, for any (complex) constant Γ_{j+1},

$$\mathbf{R}_{j+1}\left\{\begin{bmatrix}1\\ a_j(1)\\ a_j(2)\\ \vdots\\ a_j(j)\\ 0\end{bmatrix}+\Gamma_{j+1}\begin{bmatrix}0\\ a_j^*(j)\\ a_j^*(j-1)\\ \vdots\\ a_j^*(1)\\ 1\end{bmatrix}\right\}=\begin{bmatrix}\epsilon_j\\ 0\\ 0\\ \vdots\\ 0\\ \gamma_j\end{bmatrix}+\Gamma_{j+1}\begin{bmatrix}\gamma_j^*\\ 0\\ 0\\ \vdots\\ 0\\ \epsilon_j^*\end{bmatrix} \tag{5.12}$$

Since we want to find the vector $\mathbf{a}_{j+1}$ which, when multiplied by $\mathbf{R}_{j+1}$, yields a scaled unit vector, note that if we set

$$\boxed{\Gamma_{j+1}=-\frac{\gamma_j}{\epsilon_j^*}} \tag{5.13}$$

then Eq. (5.12) becomes

$$\mathbf{R}_{j+1}\mathbf{a}_{j+1}=\epsilon_{j+1}\mathbf{u}_1$$

where

$$\mathbf{a}_{j+1}=\begin{bmatrix}1\\ a_j(1)\\ a_j(2)\\ \vdots\\ a_j(j)\\ 0\end{bmatrix}+\Gamma_{j+1}\begin{bmatrix}0\\ a_j^*(j)\\ a_j^*(j-1)\\ \vdots\\ a_j^*(1)\\ 1\end{bmatrix} \tag{5.14}$$

which is the solution to the $(j+1)$st-order normal equations. Furthermore,

$$\boxed{\epsilon_{j+1}=\epsilon_j+\Gamma_{j+1}\gamma_j^*=\epsilon_j\left[1-|\Gamma_{j+1}|^2\right]} \tag{5.15}$$

is the $(j+1)$st-order modeling error.[1] If we define $a_j(0)=1$ and $a_j(j+1)=0$ then Eq. (5.14), referred to as the *Levinson order-update equation*, may be expressed as

$$\boxed{a_{j+1}(i)=a_j(i)+\Gamma_{j+1}a_j^*(j-i+1)\quad;\quad i=0,1,\ldots j+1} \tag{5.16}$$

All that is required to complete the recursion is to define the conditions necessary to initialize the recursion. These conditions are given by the solution for the model of order $j=0$,

$$\begin{aligned}a_0(0)&=1\\ \epsilon_0&=r_x(0)\end{aligned} \tag{5.17}$$

In summary, the steps of the Levinson-Durbin recursion are as follows. The recursion is first initialized with the zeroth-order solution, Eq. (5.17). Then, for $j=0,1,\ldots,p-1$, the $(j+1)$st-order model is found from the jth-order model in three steps. The first step is to use Eqs. (5.10) and (5.13) to determine the value of Γ_{j+1}, which is referred to as

[1] Since ϵ_j is real, then the complex conjugate in Eq. (5.13) may be dropped.

Table 5.1 *The Levinson-Durbin Recursion*

1. Initialize the recursion
   (a) $a_0(0) = 1$
   (b) $\epsilon_0 = r_x(0)$
2. For $j = 0, 1, \ldots, p-1$
   (a) $\gamma_j = r_x(j+1) + \sum_{i=1}^{j} a_j(i) r_x(j-i+1)$
   (b) $\Gamma_{j+1} = -\gamma_j / \epsilon_j$
   (c) For $i = 1, 2, \ldots, j$
   $$a_{j+1}(i) = a_j(i) + \Gamma_{j+1} a_j^*(j-i+1)$$
   (d) $a_{j+1}(j+1) = \Gamma_{j+1}$
   (e) $\epsilon_{j+1} = \epsilon_j \left[1 - |\Gamma_{j+1}|^2\right]$
3. $b(0) = \sqrt{\epsilon_p}$

the $(j+1)$st *reflection coefficient*. The next step of the recursion is to use the Levinson order-update equation to compute the coefficients $a_{j+1}(i)$ from $a_j(i)$. The final step of the recursion is to update the error, ϵ_{j+1}, using Eq. (5.15). This error may also be written in two equivalent forms that will be useful in later discussions. The first is

$$\epsilon_{j+1} = \epsilon_j \left[1 - |\Gamma_{j+1}|^2\right] = r_x(0) \prod_{i=1}^{j+1} \left[1 - |\Gamma_i|^2\right] \tag{5.18}$$

and the second, which follows from Eq. (5.3), is

$$\epsilon_{j+1} = r_x(0) + \sum_{i=1}^{j+1} a_{j+1}(i) r_x(i) \tag{5.19}$$

The complete recursion is listed in Table 5.1 and a MATLAB program is given in Fig. 5.1.[2]

Example 5.2.1 ***Solving the Autocorrelation Normal Equations***

Let us use the Levinson-Durbin recursion to solve the autocorrelation normal equations and find a third-order all-pole model for a signal having autocorrelation values

$$r_x(0) = 1, \quad r_x(1) = 0.5, \quad r_x(2) = 0.5, \quad r_x(3) = 0.25$$

The normal equations for the third-order all-pole model are

$$\begin{bmatrix} 1 & 0.5 & 0.5 \\ 0.5 & 1 & 0.5 \\ 0.5 & 0.5 & 1 \end{bmatrix} \begin{bmatrix} a_3(1) \\ a_3(2) \\ a_3(3) \end{bmatrix} = - \begin{bmatrix} 0.5 \\ 0.5 \\ 0.25 \end{bmatrix}$$

Using the Levinson-Durbin recursion we have

[2]The convention used in the Levinson-Durbin recursion varies a bit from one author to another. Some authors, for example, use $-\Gamma_{j+1}$ in Eq. (5.12), while others use Γ_{j+1}^*. This results, of course, in different sign conventions for Γ_{j+1}.

The Levinson-Durbin Recursion

```
function [a,epsilon]=rtoa(r)
%
   r=r(:);
   p=length(r)-1;
   a=1;
   epsilon=r(1);
   for j=2:p+1;
       gamma=-r(2:j)'*flipud(a)/epsilon;
       a=[a;0] + gamma*[0;conj(flipud(a))];
       epsilon=epsilon*(1 - abs(gamma)^2);
       end
```

Figure 5.1 *A* MATLAB *program for solving a set of Toeplitz equations using the Levinson-Durbin Recursion. This m-file provides a mapping from the autocorrelation sequence $r_x(k)$ to the filter coefficients $a_p(k)$, hence the name* `rtoa`.

1. First-order model:

$$\gamma_0 = r_x(1)$$

$$\Gamma_1 = -\frac{\gamma_0}{\epsilon_0} = -\frac{r_x(1)}{r_x(0)} = -\tfrac{1}{2} \quad ; \quad \epsilon_1 = r_x(0)\left[1 - \Gamma_1^2\right] = \tfrac{3}{4}$$

and

$$\mathbf{a}_1 = \begin{bmatrix} 1 \\ \Gamma_1 \end{bmatrix} = \begin{bmatrix} 1 \\ -1/2 \end{bmatrix}$$

2. Second-order model:

$$\gamma_1 = r_x(2) + a_1(1)r_x(1) = \tfrac{1}{4}$$

$$\Gamma_2 = -\gamma_1/\epsilon_1 = -\tfrac{1}{3} \quad ; \quad \epsilon_2 = \epsilon_1\left[1 - \Gamma_2^2\right] = \tfrac{2}{3}$$

and

$$\mathbf{a}_2 = \begin{bmatrix} 1 \\ -1/2 \\ 0 \end{bmatrix} - 1/3 \begin{bmatrix} 0 \\ -1/2 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ -1/3 \\ -1/3 \end{bmatrix}$$

3. Third-order model:

$$\gamma_2 = r_x(3) + a_2(1)r_x(2) + a_2(2)r_x(1) = -\tfrac{1}{12}$$

$$\Gamma_3 = -\gamma_2/\epsilon_2 = \tfrac{1}{8} \quad ; \quad \epsilon_3 = \epsilon_2\left[1 - \Gamma_3^2\right] = \tfrac{21}{32}$$

and

$$\mathbf{a}_3 = \begin{bmatrix} 1 \\ -1/3 \\ -1/3 \\ 0 \end{bmatrix} + \frac{1}{8} \begin{bmatrix} 0 \\ -1/3 \\ -1/3 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ -3/8 \\ -3/8 \\ 1/8 \end{bmatrix}$$

Finally, with

$$b(0) = \sqrt{\epsilon_3} = \frac{1}{8}\sqrt{42}$$

the all-pole model becomes

$$H_3(z) = \frac{\sqrt{42}/8}{1 - \frac{3}{8}z^{-1} - \frac{3}{8}z^{-2} + \frac{1}{8}z^{-3}}$$

Having found the third-order all-pole model for $x(n)$, let us determine what the next value in the autocorrelation sequence would be if $H_3(z)$ were the correct model for $x(n)$, i.e., if $x(n)$ is the inverse z-transform of $H_3(z)$. In this case, the model will remain unchanged for $k > 3$, i.e., $\Gamma_k = 0$ and, therefore, $\gamma_k = 0$ for $k > 3$. Thus, it follows from Eq. (5.10) by setting $j = 3$ that

$$r_x(4) + \sum_{i=1}^{3} a_3(i) r_x(4 - i) = 0$$

which may be solved for $r_x(4)$ as follows

$$r_x(4) = -\sum_{i=1}^{3} a_3(i) r_x(4 - i) = 7/32$$

Successive values of $r_x(k)$ may be determined in a similar fashion. For example, if $k \geq 4$, then

$$r_x(k) = -\sum_{i=1}^{3} a_3(i) r_x(k - i)$$

which are the Yule-Walker equations for an AR(3) process.

It follows from our discussions in Section 3.6.2 of Chapter 3 that if white noise with a variance of σ_w^2 is filtered with a first-order all-pole filter

$$H(z) = \frac{1}{1 - \alpha z^{-1}}$$

then the output will be an AR(1) process with an autocorrelation sequence of the form

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

In the following example, the Levinson-Durbin recursion is used to find the sequence of reflection coefficients Γ_k and model errors ϵ_k that are associated with this process.

Example 5.2.2 *The Reflection Coefficients for an AR(1) Process*

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

$$\mathbf{r}_x = \frac{\sigma_w^2}{1 - \alpha^2}[1, \alpha, \alpha^2, \ldots, \alpha^p]^T$$

To find the reflection coefficients associated with this vector of autocorrelations we may use the Levinson-Durbin recursion as follows. Initializing the recursion with

$$\mathbf{a}_0 = 1 \quad ; \quad \epsilon_0 = r_x(0) = \frac{\sigma_w^2}{1 - \alpha^2}$$

it follows that the first reflection coefficient is

$$\Gamma_1 = -\frac{r_x(1)}{r_x(0)} = -\alpha$$

and that

$$\epsilon_1 = \epsilon_0\left[1 - \Gamma_1^2\right] = \sigma_w^2$$

Thus, for the first-order model

$$\mathbf{a}_1 = \begin{bmatrix} 1 \\ -\alpha \end{bmatrix}$$

For the second-order model

$$\gamma_1 = r_x(2) + a_1(1)r_x(1) = 0$$

and

$$\Gamma_2 = 0 \quad ; \quad \epsilon_2 = \epsilon_1 = \sigma_w^2$$

so the coefficients of the model are

$$\mathbf{a}_2 = \begin{bmatrix} 1 \\ -\alpha \\ 0 \end{bmatrix}$$

Continuing, we may show by induction that $\Gamma_j = 0$ and $\epsilon_j = \sigma_w^2$ for all $j > 1$. Specifically, suppose that at the jth step of the recursion $\Gamma_j = 0$ and

$$\mathbf{a}_j = [1, \ -\alpha, \ 0, \ldots, \ 0]^T$$

Since

$$\gamma_j = r_x(j+1) + \sum_{i=1}^{j} a_j(i)r_x(j-i+1) = r_x(j+1) - \alpha r_x(j)$$

we see from the form of the autocorrelation sequence that $\gamma_j = 0$. Therefore, $\Gamma_{j+1} = 0$ and $\mathbf{a}_{j+1}$ is formed by appending a zero to $\mathbf{a}_j$ and the result follows. In summary, for an AR(1) process,

$$\boldsymbol{\Gamma}_p = [-\alpha, \ 0, \ 0, \ \ldots, \ 0]^T$$

$$\mathbf{a}_p = [1, \ -\alpha, \ 0, \ \ldots, \ 0]^T$$

$$\boldsymbol{\epsilon}_p = \left[\frac{\sigma_w^2}{1-\alpha^2}, \ \sigma_w^2, \ \sigma_w^2, \ \ldots, \ \sigma_w^2\right]^T$$

We now look at the computational complexity of the Levinson-Durbin recursion and compare it to Gaussian elimination for solving the pth-order autocorrelation normal equations. Using Gaussian elimination to solve a set of p linear equations in p unknowns, approximately $\frac{1}{3}p^3$ multiplications and divisions are required. With the Levinson-Durbin recursion, on the other hand, at the jth step of the recursion $2j + 2$ multiplications, 1 division, and $2j + 1$ additions are necessary. Since there are p steps in the recursion, the total

number of multiplications and divisions is[3]

$$\sum_{j=0}^{p-1}(2j+3) = p^2 + 2p$$

and the number of additions is

$$\sum_{j=0}^{p-1}(2j+1) = p^2$$

Therefore, the number of multiplications and divisions is proportional to p^2 for the Levinson-Durbin recursion compared with p^3 for Gaussian elimination. Another advantage of the Levinson-Durbin recursion over Gaussian elimination is that it requires less memory for data storage. Specifically, whereas Gaussian elimination requires p^2 memory locations, the Levinson-Durbin recursion requires only $2(p+1)$ locations: $p+1$ for the autocorrelation sequence $r_x(0), \ldots, r_x(p)$, and p for the model parameters $a_p(1), \ldots, a_p(p)$, and one for the error ϵ_p.

In spite of the increased efficiency of the Levinson-Durbin recursion over Gaussian elimination, it should be pointed out that solving the normal equations may only be a small fraction of the total computational cost in the modeling process. For example, note that for a signal of length N, computing the autocorrelation values $r_x(0), \ldots, r_x(p)$ requires approximately $N(p+1)$ multiplications and additions. Therefore, if $N \gg p$, then the cost associated with finding the autocorrelation sequence will dominate the computational requirements of the modeling algorithm.

5.2.2 The Lattice Filter

One of the by-products of the Levinson-Durbin recursion is the lattice structure for digital filters. Lattice filters are used routinely in digital filter implementations as a result of a number of interesting and important properties that they possess. These properties include a modular structure, low sensitivity to parameter quantization effects, and a simple method to ensure filter stability. In this section, we show how the Levinson order-update equation may be used to derive the lattice filter structure for FIR digital filters. In Chapter 6, other lattice filters structures will be developed, including the all-pole lattice and the pole-zero lattice filters, and it will be shown how these filters may be used for signal modeling.

The derivation of the lattice filter begins with the Levinson order-update equation given in Eq. (5.16). First, however, it will be convenient to define the *reciprocal vector*, denoted by $\mathbf{a}_j^R$, which is the vector that is formed by reversing the order of the elements in $\mathbf{a}_j$ and taking the complex conjugate,

$$\mathbf{a}_j = \begin{bmatrix} 1 \\ a_j(1) \\ a_j(2) \\ \vdots \\ a_j(j-1) \\ a_j(j) \end{bmatrix} \quad \Longrightarrow \quad \begin{bmatrix} a_j^*(j) \\ a_j^*(j-1) \\ a_j^*(j-2) \\ \vdots \\ a_j^*(1) \\ 1 \end{bmatrix} = \mathbf{a}_j^R$$

[3]Recall the summations given in Table 2.3, p. 16.

or,

$$\boxed{a_j^R(i) = a_j^*(j-i)} \tag{5.20}$$

for $i = 0, 1, \ldots, j$. Using the reciprocal vector in the Levinson order-update equation, Eq. (5.16) becomes

$$a_{j+1}(i) = a_j(i) + \Gamma_{j+1} a_j^R(i-1) \tag{5.21}$$

With $A_j(z)$ the z-transform of $a_j(i)$ and $A_j^R(z)$ the z-transform of the reciprocal sequence $a_j^R(i)$, it follows from Eq. (5.20) that $A_j(z)$ is related to $A_j^R(z)$ by

$$\boxed{A_j^R(z) = z^{-j} A_j^*(1/z^*)} \tag{5.22}$$

Rewriting Eq. (5.21) in terms of $A_j(z)$ and $A_j^R(z)$ gives

$$A_{j+1}(z) = A_j(z) + \Gamma_{j+1} z^{-1} A_j^R(z) \tag{5.23}$$

which is an order-update equation for $A_j(z)$.

The next step is to derive an order-update equation for $a_{j+1}^R(i)$ and $A_{j+1}^R(z)$. Beginning with Eq. (5.21), note that if we take the complex conjugate of both sides of the equation and replace i with $j - i + 1$, we have

$$a_{j+1}^*(j-i+1) = a_j^*(j-i+1) + \Gamma_{j+1}^* a_j(i) \tag{5.24}$$

Incorporating the definition of the reversed vector, Eq. (5.20), into Eq. (5.24) gives the desired update equation for $a_{j+1}^R(i)$,

$$a_{j+1}^R(i) = a_j^R(i-1) + \Gamma_{j+1}^* a_j(i) \tag{5.25}$$

Expressed in the z-domain, Eq. (5.25) gives the desired update equation for $A_{j+1}^R(z)$,

$$A_{j+1}^R(z) = z^{-1} A_j^R(z) + \Gamma_{j+1}^* A_j(z) \tag{5.26}$$

In summary, we have a pair of coupled difference equations,

$$\begin{aligned} a_{j+1}(n) &= a_j(n) + \Gamma_{j+1} a_j^R(n-1) \\ a_{j+1}^R(n) &= a_j^R(n-1) + \Gamma_{j+1}^* a_j(n) \end{aligned} \tag{5.27}$$

for updating $a_j(n)$ and $a_j^R(n)$, along with a pair of coupled equations for updating the system functions $A_j(z)$ and $A_j^R(z)$, which may be written in matrix form as

$$\begin{bmatrix} A_{j+1}(z) \\ A_{j+1}^R(z) \end{bmatrix} = \begin{bmatrix} 1 & \Gamma_{j+1} z^{-1} \\ \Gamma_{j+1}^* & z^{-1} \end{bmatrix} \begin{bmatrix} A_j(z) \\ A_j^R(z) \end{bmatrix} \tag{5.28}$$

Both of these representations describe to the two-port network shown in Fig. 5.2. This two-port represents the basic module used to implement an FIR lattice filter. Cascading p such lattice filter modules with reflection coefficients $\Gamma_1, \Gamma_2, \ldots, \Gamma_p$, forms the pth-order lattice filter shown in Fig. 5.3. Note that the system function between the input $\delta(n)$ and the output $a_p(n)$ is $A_p(z)$, whereas the system function relating the input to the output $a_p^R(n)$ is $A_p^R(z)$.

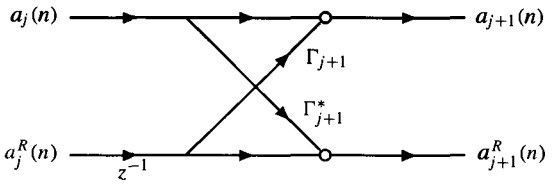


Figure 5.2 *A two-port network relating $a_j(n)$ and $a^R_j(n)$ to $a_{j+1}(n)$ and $a^R_{j+1}(n)$. This network is a single stage of an FIR lattice filter.*

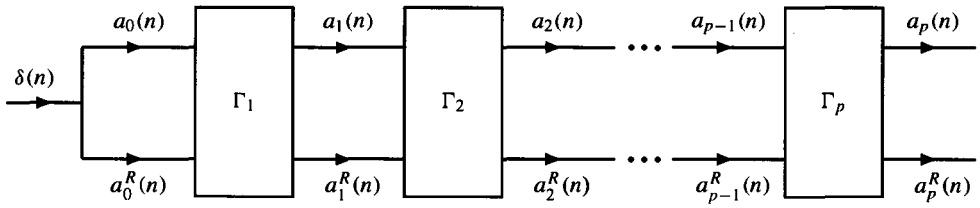


Figure 5.3 *A pth-order FIR lattice filter built up as a cascade of p two-port networks.*

5.2.3 Properties

In this section, we will establish some important properties of the reflection coefficient sequence that is generated by the Levinson-Durbin recursion and we will establish an important property of the solution to the autocorrelation normal equations. Specifically, we will show that the roots of $A_p(z)$ will be inside the unit circle if and only if the reflection coefficients are bounded by one in magnitude or, equivalently, if and only if $\mathbf{R}_p$ is positive definite. We will also show that the all-pole model obtained from the autocorrelation method is guaranteed to be stable. Finally, we will establish the *autocorrelation matching property*, which states that if we set $b(0) = \sqrt{\epsilon_p}$ in the autocorrelation method and if $h(n)$ is the inverse z-transform of

$$H(z) = \frac{b(0)}{A_p(z)}$$

then the autocorrelation of $h(n)$ is equal to the autocorrelation of $x(n)$.

The first property of the reflection coefficients is the following.

> ***Property 1.*** The reflection coefficients that are generated by the Levinson-Durbin recursion to solve the autocorrelation normal equations are bounded by one in magnitude, $|\Gamma_j| \leq 1$.

This property follows immediately from the fact that since ϵ_j is the minimum squared error,

$$\epsilon_j = \{\mathcal{E}_j\}_{\min} \tag{5.29}$$

with

$$\mathcal{E}_j = \sum_{n=0}^{\infty} |e(n)|^2$$

where $e(n)$ is the jth-order modeling error, then $\epsilon_j \geq 0$. Therefore, since

$$\epsilon_j = \epsilon_{j-1}\left[1 - |\Gamma_j|^2\right]$$

it follows that if $\epsilon_j \geq 0$ and $\epsilon_{j-1} \geq 0$, then $|\Gamma_j| \leq 1$. ■

It is important to keep in mind that the validity of Property 1 relies on the nonnegativity of ϵ_j. Implicit in this nonnegativity is the assumption that the autocorrelation values $r_x(k)$ in $\mathbf{R}_p$ are computed according to Eq. (4.121). It is not necessarily true, for example, that $\epsilon_j \geq 0$ and $|\Gamma_j| \leq 1$ if an arbitrary sequence of numbers $r_x(0), r_x(1), \ldots, r_x(p-1)$ are used in the matrix $\mathbf{R}_p$. To illustrate this point, consider what happens if we set $r_x(0) = 1$ and $r_x(1) = 2$ in the normal equations and solve for the first-order model. In this case,

$$\Gamma_1 = -r_x(1)/r_x(0) = -2$$

which has a magnitude that is greater than one. This example is not in conflict with Property 1, however, since for any valid autocorrelation sequence, $|r_x(0)| \geq |r_x(1)|$. As we will soon discover (see Property 7 on p. 253), the unit magnitude constraint on the reflection coefficients Γ_j and the nonnegativity of the sequence ϵ_j are tied to the positive definiteness of the matrix $\mathbf{R}_p$.

The next property establishes a relationship between the locations of the roots of the polynomial $A_p(z)$ and the magnitudes of the reflection coefficients Γ_j.

Property 2. If $a_p(k)$ is a set of model parameters and Γ_j is the corresponding set of reflection coefficients, then

$$A_p(z) = 1 + \sum_{k=1}^{p} a_p(k) z^{-k}$$

will be a *minimum phase polynomial* (all of the roots of $A_p(z)$ will be *inside* the unit circle) if and only if $|\Gamma_j| < 1$ for all j. Furthermore, if $|\Gamma_j| \leq 1$ for all j then the roots of $A_p(z)$ must lie either inside or on the unit circle.

There are many different ways to establish this property [14, 20, 24]. One way, as demonstrated below, is to use the *encirclement principle* (*principle of the argument*) from complex analysis [3].

Encirclement Principle. Given a rational function of z

$$P(z) = \frac{B(z)}{A(z)}$$

let C be a simple closed curve in the z plane as shown in Fig. 5.4. As the path C is traversed in a counterclockwise direction, a closed curve is generated in the $P(z)$ plane that encircles the origin $(N_z - N_p)$ times in a counterclockwise direction where N_z is the number of *zeros inside* C and N_p is the number of *poles inside* C.[4]

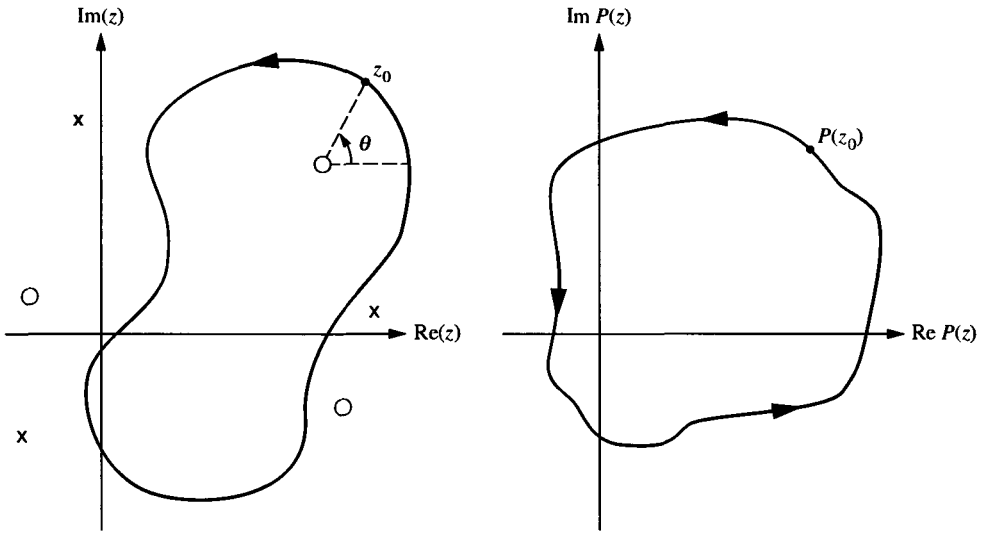


Figure 5.4 *A simple closed curve in the z-plane encircling a simple zero and the corresponding contour in the $P(z)$ plane.*

The encirclement principle may be interpreted geometrically as follows. Since C is a simple closed curve, beginning and ending at z_0, then $P(z)$ will also be a closed curve, beginning and ending at the same point, $P(z_0)$. If there is a zero contained within the contour then the angle θ from the zero to a point on the contour will increase by 2π as the contour is traversed. In the $P(z)$ plane, therefore, the curve that is generated will encircle the origin in a counterclockwise direction. Similarly, if there is a pole inside the contour, then the angle will *decrease* by 2π as C is traversed, which produces a curve in the $P(z)$ plane that encircles the origin in the clockwise direction. With N_z zeros and N_p poles inside C, the change in angle is $2\pi(N_z - N_p)$ and the encirclement principle follows. Although not stated in the encirclement principle, it should be clear that if there is a zero of $P(z)$ on the contour C, then the curve in the $P(z)$ plane will pass *through* the origin.

With the encirclement principle, we may use induction to establish Property 2. Specifically, for a first-order model,

$$A_1(z) = 1 + \Gamma_1 z^{-1}$$

and it follows that $A_1(z)$ will be minimum phase if and only if $|\Gamma_1| < 1$. Let us now assume that $A_j(z)$ is minimum phase and show that $A_{j+1}(z)$ will be minimum phase if and only if $|\Gamma_{j+1}| < 1$. To accomplish this, we will use the z-domain representation of the Levinson order-update equation given in Eq. (5.23), which is repeated below for convenience

$$A_{j+1}(z) = A_j(z) + \Gamma_{j+1} z^{-1} A_j^R(z)$$

Dividing both sides of this update equation by $A_j(z)$ we have

$$P(z) = \frac{A_{j+1}(z)}{A_j(z)} = 1 + \Gamma_{j+1} z^{-1} \frac{A_j^R(z)}{A_j(z)} \tag{5.30}$$

[4]If $N_z - N_p$ is negative then the curve $P(z)$ encircles the origin in a clockwise direction $|N_z - N_p|$ times.

We will now show that if C is a closed contour that traverses the unit circle then the number of times that $P(z)$ encircles the origin in a clockwise direction is equal to the number of zeros of $A_{j+1}(z)$ that are *outside* the unit circle. Since $A_j(z)$ is assumed to be minimum phase, $A_j(z)$ has j zeros *inside* the unit circle and j poles at $z = 0$. Now let us assume that $A_{j+1}(z)$ has l zeros *outside* the unit circle and $j + 1 - l$ zeros *inside*. Since $A_{j+1}(z)$ will have $j + 1$ poles at $z = 0$, it follows that the number of times that $P(z)$ encircles the origin in a counter clockwise direction is

$$N_z - N_p = (j + 1 - l) - (j + 1) = -l$$

or, l times in a clockwise direction. Now, note that since the contour C is the unit circle, then $z = e^{j\omega}$ and

$$\left|\Gamma_{j+1} z^{-1} \frac{A_j^R(z)}{A_j(z)}\right|_C = |\Gamma_{j+1}| \left|\frac{A_j^R(e^{j\omega})}{A_j(e^{j\omega})}\right|_C = |\Gamma_{j+1}|$$

Therefore, it follows from Eq. (5.30) that $P(z)$ traces out a curve which is a circle of radius $|\Gamma_{j+1}|$ that is centered at $z = 1$ (see Fig. 5.5). Thus, if $|\Gamma_{j+1}| < 1$, then $P(z)$ does not encircle or pass through the origin. Conversely, if $|\Gamma_{j+1}| > 1$ then $P(z)$ encircles the origin and $A_{j+1}(z)$ will not have all of its zeros inside the unit circle. Consequently, if $A_j(z)$ is minimum phase, then $A_{j+1}(z)$ will be minimum phase if and only if $|\Gamma_{j+1}| < 1$. ■

There is an intimate connection between the positive definiteness of the Toeplitz matrix, $\mathbf{R}_p$, and the minimum phase property of the solution to the normal equations. In particular, the following property states that $A_p(z)$ will be minimum phase if and only if $\mathbf{R}_p$ is positive definite (see also Section 5.2.7).

Property 3. If $\mathbf{a}_p$ is the solution to the Toeplitz normal equations $\mathbf{R}_p\mathbf{a}_p = \epsilon_p\mathbf{u}_1$, then $A_p(z)$ will be minimum phase if and only if $\mathbf{R}_p$ is positive definite, $\mathbf{R}_p > 0$.

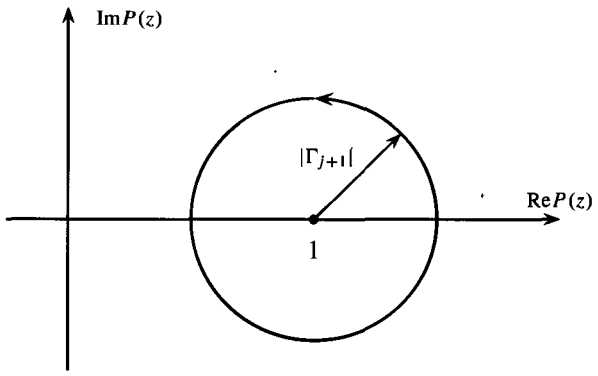


Figure 5.5 *The curve $P(z)$, which is a circle of radius $|\Gamma_{j+1}|$ that is centered at $z = 1$.*

To establish this property, let α be a root of the polynomial $A_p(z)$. In general, α will be complex and we may factor $A_p(z)$ as follows

$$A_p(z) = 1 + \sum_{k=1}^{p} a_p(k)z^{-k} = (1 - \alpha z^{-1})(1 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_{p-1} z^{-(p-1)})$$

We will show that if $\mathbf{R}_p > 0$, then $|\alpha| < 1$. Writing the vector $\mathbf{a}_p$ in factored form as

$$\mathbf{a}_p = \begin{bmatrix} 1 \\ a_p(1) \\ a_p(2) \\ \vdots \\ a_p(p) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ b_1 & 1 \\ b_2 & b_1 \\ \vdots & \vdots \\ 0 & b_{p-1} \end{bmatrix} \begin{bmatrix} 1 \\ -\alpha \end{bmatrix} = \mathbf{B} \begin{bmatrix} 1 \\ -\alpha \end{bmatrix} \tag{5.31}$$

it follows that

$$\mathbf{R}_p \mathbf{a}_p = \mathbf{R}_p \mathbf{B} \begin{bmatrix} 1 \\ -\alpha \end{bmatrix} = \epsilon_p \mathbf{u}_1 \tag{5.32}$$

Multiplying Eq. (5.32) on the left by $\mathbf{B}^H$ we have

$$\mathbf{B}^H \mathbf{R}_p \mathbf{B} \begin{bmatrix} 1 \\ -\alpha \end{bmatrix} = \epsilon_p \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

Since $\mathbf{B}$ has full rank, if $\mathbf{R}_p$ is positive definite, then $\mathbf{B}^H \mathbf{R}_p \mathbf{B}$ will be positive definite (see p. 40). Therefore,

$$\mathbf{B}^H \mathbf{R}_p \mathbf{B} = \begin{bmatrix} s_0 & s_1 \\ s_1^* & s_0 \end{bmatrix} > 0$$

which implies that

$$|s_0|^2 > |s_1|^2 \tag{5.33}$$

Since

$$\mathbf{B}^H \mathbf{R}_p \mathbf{B} \begin{bmatrix} 1 \\ -\alpha \end{bmatrix} = \begin{bmatrix} s_0 - s_1 \alpha \\ s_1^* - s_0 \alpha \end{bmatrix} = \epsilon_p \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

then $s_0 \alpha = s_1^*$ and

$$\alpha = \frac{s_1^*}{s_0} \tag{5.34}$$

Thus, Eq. (5.33) implies that $|\alpha| < 1$ and $A_p(z)$ is minimum phase. Conversely, if $A_p(z)$ is not minimum phase, then $A_p(z)$ will have a root α with $|\alpha| > 1$. Therefore, from Eq. (5.34) it follows that $|s_1| > |s_0|$ and $\mathbf{R}_p$ is not positive definite. ■

Property 1 states that the reflection coefficients that are generated in the course of solving the autocorrelation normal equations are bounded by one in magnitude. Property 2 states that a polynomial $A_p(z)$ will have all of its roots inside the unit circle if its reflection coefficients are less than one in magnitude. Combining Properties 1 and 2 leads to the following property.

Property 4. The autocorrelation method produces a stable all-pole model.

In the next property, we consider what happens to the roots of the polynomial $A_p(z)$ when $|\Gamma_p| = 1$ with the remaining terms in the reflection coefficient sequence being less than one in magnitude.

Property 5. Let $a_p(k)$ be a set of filter coefficients and Γ_j the corresponding set of reflection coefficients. If $|\Gamma_j| < 1$ for $j = 1, \ldots, p-1$ and $|\Gamma_p| = 1$, then the polynomial

$$A_p(z) = 1 + \sum_{k=1}^{p} a_p(k) z^{-k}$$

has all of its roots *on* the unit circle.

This property may be established easily as follows. Let Γ_j be a sequence of reflection coefficients with $|\Gamma_j| < 1$ for $j < p$ and $|\Gamma_p| = 1$. Since $|\Gamma_j| \leq 1$ for all j, it follows from Property 2 that all of the zeros of $A_p(z)$ are on or inside the unit circle. If we denote the zeros of $A_p(z)$ by z_k, then

$$A_p(z) = \prod_{k=1}^{p} (1 - z_k z^{-1})$$

By equating the coefficients of z^{-p} on both sides of the equation, we have

$$a_p(p) = \prod_{k=1}^{p} z_k \tag{5.35}$$

However, since $a_p(p) = \Gamma_p$, if $|\Gamma_p| = 1$, then

$$\prod_{k=1}^{p} |z_k| = 1 \tag{5.36}$$

Therefore, if there is a zero of $A_p(z)$ that has a magnitude that is less than one, then there must be *at least one* zero with a magnitude greater than one, otherwise the product of the roots would not equal one. This, however, contradicts the requirement that all of the roots must lie on or inside the unit circle and Property 5 is established. ■

This property may be used to develop a constrained lattice filter for detecting sinusoids in noise (see computer exercise C6.1 in Chapter 6). It also forms the basis for an efficient algorithm in spectrum estimation for finding the Pisarenko harmonic decomposition (Chapter 8) of a stationary random process [11].

The next property relates the autocorrelation sequence of $x(n)$ to the autocorrelation sequence of $h(n)$, the unit sample response of the all-pole filter that is used to model $x(n)$. Specifically, as we demonstrate below, if $b(0)$, the numerator coefficient in $H(z)$, is selected

so that $x(n)$ satisfies the energy matching constraint, $r_x(0) = r_h(0)$, then $b(0) = \sqrt{\epsilon_p}$ and the autocorrelation sequences are equal for $|k| \leq p$. This is the so-called *autocorrelation matching property.*

> ***Property 6—Autocorrelation matching property.*** If $b(0)$ is chosen to satisfy the energy matching constraint, then $b(0) = \sqrt{\epsilon_p}$ and the autocorrelation sequences of $x(n)$ and $h(n)$ are equal for $|k| \leq p$.

In order to establish this property, we will follow the approach given in [18]. We begin by noting that since $h(n)$ is the unit sample response of the all-pole model for $x(n)$ with

$$H(z) = \frac{b(0)}{1 + \sum_{k=1}^{p} a_p(k) z^{-k}}$$

where the coefficients $a_p(k)$ are solutions to the normal equations

$$\mathbf{R}_x \mathbf{a}_p = \epsilon_p \mathbf{u}_1 \tag{5.37}$$

then $h(n)$ satisfies the difference equation

$$h(n) + \sum_{k=1}^{p} a_p(k) h(n-k) = b(0)\delta(n) \tag{5.38}$$

Multiplying both sides of Eq. (5.38) by $h^*(n-l)$ and summing over n yields

$$\begin{aligned} \sum_{n=0}^{\infty} h(n)h^*(n-l) + \sum_{k=1}^{p} a_p(k) \sum_{n=0}^{\infty} h(n-k)h^*(n-l) &= b(0) \sum_{n=0}^{\infty} \delta(n)h^*(n-l) \\ &= b(0)h^*(-l) \end{aligned} \tag{5.39}$$

With

$$r_h(l) = \sum_{n=0}^{\infty} h(n)h^*(n-l) = r_h^*(-l)$$

we may rewrite Eq. (5.39) in terms of $r_h(l)$ as follows

$$r_h(l) + \sum_{k=1}^{p} a_p(k) r_h(l-k) = b(0)h^*(-l)$$

Since $h(n)$ is causal, then $h(n) = 0$ for $n < 0$ and $h(0) = b(0)$. Therefore, for $l \geq 0$, it follows that

$$r_h(l) + \sum_{k=1}^{p} a_p(k) r_h(l-k) = |b(0)|^2 \delta(l) \quad ; \quad l \geq 0 \tag{5.40}$$

Using the conjugate symmetry of $r_h(l)$ we may write Eq. (5.40) in matrix form as

$$\begin{bmatrix} r_h(0) & r_h^*(1) & r_h^*(2) & \cdots & r_h^*(p) \\ r_h(1) & r_h(0) & r_h^*(1) & \cdots & r_h^*(p-1) \\ r_h(2) & r_h(1) & r_h(0) & \cdots & r_h^*(p-2) \\ \vdots & \vdots & \vdots & & \vdots \\ r_h(p) & r_h(p-1) & r_h(p-2) & \cdots & r_h(0) \end{bmatrix} \begin{bmatrix} 1 \\ a_p(1) \\ a_p(2) \\ \vdots \\ a_p(p) \end{bmatrix} = |b(0)|^2 \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{5.41}$$

or

$$\mathbf{R}_h \mathbf{a}_p = |b(0)|^2 \mathbf{u}_1 \tag{5.42}$$

Therefore, $\mathbf{a}_p$ is the solution to each of the following sets of Toeplitz equations,

$$\begin{aligned} \mathbf{R}_h \mathbf{a}_p &= |b_0|^2 \mathbf{u}_1 \\ \mathbf{R}_x \mathbf{a}_p &= \epsilon_p \mathbf{u}_1 \end{aligned} \tag{5.43}$$

Now, suppose that $b(0)$ is chosen to satisfy the energy matching constraint,

$$r_x(0) = \sum_{n=0}^{\infty} |x(n)|^2 = \sum_{n=0}^{\infty} |h(n)|^2 = r_h(0) \tag{5.44}$$

By induction we may then show that $r_x(k) = r_h(k)$ for $k > 0$ as follows. Using the Levinson-Durbin recursion to solve Eq. (5.43) we have

$$a_1(1) = -\frac{r_x(1)}{r_x(0)} = -\frac{r_h(1)}{r_h(0)}$$

Since $r_x(0) = r_h(0)$, it follows that $r_x(1) = r_h(1)$. Now, assume that $r_x(k) = r_h(k)$ for $k = 1, 2, \ldots, j$. Using Eqs. (5.10) and (5.13) we have

$$\begin{aligned} r_x(j+1) &= -\Gamma_{j+1}\epsilon_j - \sum_{i=1}^{j} a_j(i) r_x(j+1-i) \\ r_h(j+1) &= -\Gamma_{j+1}\epsilon_j - \sum_{i=1}^{j} a_j(i) r_h(j+1-i) \end{aligned} \tag{5.45}$$

Therefore, since $r_x(i) = r_h(i)$ for $i = 1, 2, \ldots, j$, then $r_x(j+1) = r_h(j+1)$. Finally, from Eq. (5.41) note that

$$|b(0)|^2 = r_h(0) + \sum_{k=1}^{p} a_p(k) r_h(k)$$

With $r_h(k) = r_x(k)$ it follows that

$$|b(0)|^2 = r_x(0) + \sum_{k=1}^{p} a_p(k) r_x(k) = \epsilon_p \tag{5.46}$$

Therefore, $b(0) = \sqrt{\epsilon_p}$ and the autocorrelation matching property is established. ■

5.2.4 The Step-Up and Step-Down Recursions

The Levinson-Durbin recursion solves the autocorrelation normal equations and produces an all-pole model for a signal from the autocorrelation sequence $r_x(k)$. In addition to the model parameters, the recursion generates a set of reflection coefficients, $\Gamma_1, \Gamma_2, \ldots, \Gamma_p$,

along with the final modeling error, ϵ_p. Thus, the recursion may be viewed as a mapping

$$\{r_x(0), r_x(1), \ldots, r_x(p)\} \xrightarrow{LEV} \begin{cases} a_p(1), a_p(2), \ldots, a_p(p), b(0) \\ \Gamma_1, \Gamma_2, \ldots, \Gamma_p, \epsilon_p \end{cases}$$

from an autocorrelation sequence to a set of model parameters and to a set of reflection coefficients. There are many instances in which it would be convenient to be able to derive the reflection coefficients, Γ_j, from the filter coefficients $a_p(k)$, and vice versa. In this section, we show how one set of parameters may be derived from the other in a recursive fashion. The recursions for performing these transformations are called the *step-up* and *step-down* recursions.

The Levinson order-update equation given in Eq. (5.16) is a recursion for deriving the filter coefficients $a_p(k)$ from the reflection coefficients, Γ_j. Specifically, since

$$a_{j+1}(i) = a_j(i) + \Gamma_{j+1} a_j^*(j - i + 1) \tag{5.47}$$

then the filter coefficients $a_{j+1}(i)$ may be easily found from $a_j(i)$ and Γ_{j+1}. The recursion is initialized by setting $a_0(0) = 1$ and, after the coefficients $a_p(k)$ have been determined, the recursion is completed by setting $b(0) = \sqrt{\epsilon_p}$. Since Eq. (5.47) defines how the model parameters for a jth-order filter may be updated (stepped-up) to a $(j+1)$st-order filter given Γ_{j+1}, Eq. (5.47) is referred to as the *step-up recursion*. The recursion, represented pictorially as

$$\{\Gamma_1, \Gamma_2, \ldots, \Gamma_p, \epsilon_p\} \xrightarrow{Step-up} \{a_p(1), a_p(2), \ldots, a_p(p), b(0)\}$$

is summarized in Table 5.2 and a MATLAB program is given in Fig 5.6.

Table 5.2 *The Step-up Recursion*

1. Initialize the recursion: $a_0(0) = 1$
2. For $j = 0, 1 \ldots, p-1$
   (a) For $i = 1, 2, , \ldots, j$
   $a_{j+1}(i) = a_j(i) + \Gamma_{j+1} a_j^*(j - i + 1)$
   (b) $a_{j+1}(j+1) = \Gamma_{j+1}$
3. $b(0) = \sqrt{\epsilon_p}$

The Step-Up Recursion

```
function a=gtoa(gamma)
%
   a=1;
   gamma=gamma(:);
   p=length(gamma);
   for j=2:p+1;
      a=[a;0] + gamma(j-1)*[0;conj(flipud(a))];
      end
```

Figure 5.6 *A* MATLAB *program for the step-up recursion to find the filter coefficients $a_p(k)$ from the reflection coefficients.*

Example 5.2.3 *The Step-Up Recursion*

Given the reflection coefficient sequence

$$\mathbf{\Gamma}_2 = [\Gamma_1, \Gamma_2]^T$$

the first- and second-order all-pole models may be found from the step-up recursion as follows. From Eq. (5.14), the first-order model is

$$\mathbf{a}_1 = \begin{bmatrix} 1 \\ a_1(1) \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \Gamma_1 \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ \Gamma_1 \end{bmatrix}$$

Again using Eq. (5.14) we have for $\mathbf{a}_2$,

$$\mathbf{a}_2 = \begin{bmatrix} 1 \\ a_2(1) \\ a_2(2) \end{bmatrix} = \begin{bmatrix} 1 \\ a_1(1) \\ 0 \end{bmatrix} + \Gamma_2 \begin{bmatrix} 0 \\ a_1^*(1) \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 \\ \Gamma_1 \\ 0 \end{bmatrix} + \Gamma_2 \begin{bmatrix} 0 \\ \Gamma_1^* \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ \Gamma_1 + \Gamma_1^* \Gamma_2 \\ \Gamma_2 \end{bmatrix}$$

Therefore, in terms of the reflection coefficients Γ_1 and Γ_2, the general form for the second-order all-pole model is

$$A_2(z) = 1 + (\Gamma_1 + \Gamma_1^* \Gamma_2) z^{-1} + \Gamma_2 z^{-2}$$

Given a second-order model, $\mathbf{a}_2$, note that if a third reflection coefficient, Γ_3, is appended to the sequence, then $\mathbf{a}_3$ may be easily found from $\mathbf{a}_2$ as follows

$$\mathbf{a}_3 = \begin{bmatrix} 1 \\ a_3(1) \\ a_3(2) \\ a_3(3) \end{bmatrix} = \begin{bmatrix} 1 \\ a_2(1) \\ a_2(2) \\ 0 \end{bmatrix} + \Gamma_3 \begin{bmatrix} 0 \\ a_2^*(2) \\ a_2^*(1) \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 \\ a_2(1) + \Gamma_3 a_2^*(2) \\ a_2(2) + \Gamma_3 a_2^*(1) \\ \Gamma_3 \end{bmatrix} = \begin{bmatrix} 1 \\ (\Gamma_1 + \Gamma_1^* \Gamma_2) + \Gamma_3 \Gamma_2^* \\ \Gamma_2 + \Gamma_3 (\Gamma_1^* + \Gamma_1 \Gamma_2^*) \\ \Gamma_3 \end{bmatrix}$$

Thus, it is always possible to increase the order of the filter without having to solve again for the lower-order filters.

The step-up recursion is a mapping, $\mathcal{L} : \mathbf{\Gamma}_p \longrightarrow \mathbf{a}_p$, from a reflection coefficient sequence $\mathbf{\Gamma}_p$ to a filter coefficient sequence $\mathbf{a}_p$. In most cases, this mapping may be inverted, $\mathcal{L}^{-1} : \mathbf{a}_p \longrightarrow \mathbf{\Gamma}_p$, and the reflection coefficients determined from the model parameters, $\mathbf{a}_p$. The mapping that accomplishes this is called the *step-down* or *backward Levinson* recursion. The procedure for finding the reflection coefficients is based on the fact that since

$$\Gamma_j = a_j(j) \tag{5.48}$$

then the reflection coefficients may be computed by running the Levinson-Durbin recursion *backwards*. Specifically, beginning with $\mathbf{a}_p$ we set $\Gamma_p = a_p(p)$. Then, we recursively find

each of the lower-order models, $\mathbf{a}_j$, for $j = p-1, p-2, \ldots, 1$ and set $\Gamma_j = a_j(j)$ as illustrated below

$$\begin{array}{lcccccc}
\mathbf{a}_p & = & a_p(1) & a_p(2) & \cdots & a_p(p-2) & a_p(p-1) & \boxed{\Gamma_p} \\
\mathbf{a}_{p-1} & = & a_{p-1}(1) & a_{p-1}(2) & \cdots & a_{p-1}(p-2) & \boxed{\Gamma_{p-1}} & \\
\mathbf{a}_{p-2} & = & a_{p-2}(1) & a_{p-2}(2) & \cdots & \boxed{\Gamma_{p-2}} & & \\
\vdots & & \vdots & \vdots & & & & \\
\mathbf{a}_2 & = & a_2(1) & \boxed{\Gamma_2} & & & & \\
\mathbf{a}_1 & = & \boxed{\Gamma_1} & & & & &
\end{array}$$

To see how the jth-order model may be derived from the $(j+1)$st-order model, let us assume that the coefficients $a_{j+1}(i)$ are known. The Levinson order-update equation, Eq. (5.16), expresses $a_{j+1}(i)$ in terms of $a_j(i)$ and $a_j(j-i+1)$ as follows

$$a_{j+1}(i) = a_j(i) + \Gamma_{j+1} a_j^*(j-i+1) \tag{5.49}$$

which provides us with one equation in the two unknowns, $a_j(i)$ and $a_j^*(j-i+1)$. However, if we again use the Levinson order-update equation to express $a_{j+1}(j-i+1)$ in terms of $a_j(j-i+1)$ and $a_j^*(i)$ we have

$$a_{j+1}(j-i+1) = a_j(j-i+1) + \Gamma_{j+1} a_j^*(i)$$

which, after taking complex conjugates,

$$a_{j+1}^*(j-i+1) = a_j^*(j-i+1) + \Gamma_{j+1}^* a_j(i) \tag{5.50}$$

gives us a second linear equation in the same two unknowns. Writing Eqs. (5.49) and (5.50) in matrix form we have

$$\begin{bmatrix} a_{j+1}(i) \\ a_{j+1}^*(j-i+1) \end{bmatrix} = \begin{bmatrix} 1 & \Gamma_{j+1} \\ \Gamma_{j+1}^* & 1 \end{bmatrix} \begin{bmatrix} a_j(i) \\ a_j^*(j-i+1) \end{bmatrix} \tag{5.51}$$

If $|\Gamma_{j+1}| \neq 1$, then the matrix in Eq. (5.51) is invertible and we may solve uniquely for $a_j(i)$.[5] The solution is

$$\boxed{a_j(i) = \frac{1}{1-|\Gamma_{j+1}|^2}\left[a_{j+1}(i) - \Gamma_{j+1} a_{j+1}^*(j-i+1)\right]} \tag{5.52}$$

which is the *step-down recursion*. This recursion may also be written in vector form as follows

$$\begin{bmatrix} a_j(1) \\ a_j(2) \\ \vdots \\ a_j(j) \end{bmatrix} = \frac{1}{1-|\Gamma_{j+1}|^2} \left\{ \begin{bmatrix} a_{j+1}(1) \\ a_{j+1}(2) \\ \vdots \\ a_{j+1}(j) \end{bmatrix} - \Gamma_{j+1} \begin{bmatrix} a_{j+1}^*(j) \\ a_{j+1}^*(j-1) \\ \vdots \\ a_{j+1}^*(1) \end{bmatrix} \right\} \tag{5.53}$$

Once the sequence $a_j(i)$ has been found, we set $\Gamma_j = a_j(j)$ and the recursion continues by finding the next lower-order polynomial. The step-down recursion, represented pictorially

[5] If $|\Gamma_{j+1}| = 1$, then Eq. (5.51) is a singular set of equations and the mapping is not uniquely invertible.

as

$$\{a_p(1), a_p(2), \ldots, a_p(p), b(0)\} \xrightarrow{Step-down} \{\Gamma_1, \Gamma_2, \ldots, \Gamma_p, \epsilon_p\}$$

is summarized in Table 5.3.

Another and perhaps more elegant way to derive the step-down recursion is to use Eq. (5.28) to solve for the jth-order polynomial $A_j(z)$ in terms of $A_{j+1}(z)$. The solution is easily seen to be

$$\begin{bmatrix} A_j(z) \\ A_j^R(z) \end{bmatrix} = \frac{1}{z^{-1}(1 - |\Gamma_{j+1}|^2)} \begin{bmatrix} z^{-1} & -\Gamma_{j+1}z^{-1} \\ -\Gamma_{j+1}^* & 1 \end{bmatrix} \begin{bmatrix} A_{j+1}(z) \\ A_{j+1}^R(z) \end{bmatrix} \tag{5.54}$$

Therefore,

$$\boxed{A_j(z) = \frac{1}{1 - |\Gamma_{j+1}|^2}\left[A_{j+1}(z) - \Gamma_{j+1}A_{j+1}^R(z)\right]} \tag{5.55}$$

which is the z-domain formulation of Eq. (5.52). A MATLAB program for the step-down recursion is given in Fig. 5.7.

Table 5.3 *The Step-down Recursion*

1. Set $\Gamma_p = a_p(p)$
2. For $j = p - 1, p - 2, \ldots, 1$
   (a) For $i = 1, 2, \ldots, j$
   $$a_j(i) = \frac{1}{1 - |\Gamma_{j+1}|^2}\left[a_{j+1}(i) - \Gamma_{j+1}a_{j+1}^*(j - i + 1)\right]$$
   (b) Set $\Gamma_j = a_j(j)$
   (c) If $|\Gamma_j| = 1$, Quit.
3. $\epsilon_p = b^2(0)$

The Step-Down Recursion

```
function gamma=atog(a)
%
   a=a(:);
   p=length(a);
   a=a(2:p)/a(1);
   gamma(p-1)=a(p-1);
   for j=p-1:-1:2;
      a=(a(1:j-1) - gamma(j)*flipud(conj(a(1:j-1)))))./ ...
        (1 - abs(gamma(j))^2);
      gamma(j-1)=a(j-1);
      end
```

Figure 5.7 *A* MATLAB *program for the step-down recursion to find the reflection coefficients from a set of filter coefficients* $a_p(k)$.

Example 5.2.4 *The Step-Down Recursion*

Suppose that we would like to implement the third-order FIR filter

$$H(z) = 1 + 0.5z^{-1} - 0.1z^{-2} - 0.5z^{-3}$$

using a lattice filter structure. Using the step-down recursion, the reflection coefficients may be found from the vector

$$\mathbf{a}_3 = [1,\ 0.5,\ -0.1,\ -0.5]^T$$

and then implemented using the structure shown in Fig. 5.3. The step-down recursion begins by setting $\Gamma_3 = a_3(3) = -0.5$. Then, the coefficients of the second-order polynomial $\mathbf{a}_2 = [1, a_2(1), a_2(2)]^T$ are found from $a_3(i)$ using Eq. (5.53) with $j = 2$ as follows

$$\begin{bmatrix} a_2(1) \\ a_2(2) \end{bmatrix} = \frac{1}{1-\Gamma_3^2}\left\{\begin{bmatrix} a_3(1) \\ a_3(2) \end{bmatrix} - \Gamma_3 \begin{bmatrix} a_3(2) \\ a_3(1) \end{bmatrix}\right\}$$

From the given values for $\mathbf{a}_3$ and Γ_{j+1} we find

$$\begin{bmatrix} a_2(1) \\ a_2(2) \end{bmatrix} = \frac{1}{1-0.25}\left\{\begin{bmatrix} 0.5 \\ -0.1 \end{bmatrix} + 0.5\begin{bmatrix} -0.1 \\ 0.5 \end{bmatrix}\right\} = \begin{bmatrix} 0.6 \\ 0.2 \end{bmatrix}$$

and $\Gamma_2 = a_2(2) = 0.2$. Having found $\mathbf{a}_2$, Eq. (5.52) may again be used to find $\mathbf{a}_1$ as follows

$$a_1(1) = \frac{1}{1-\Gamma_2^2}[a_2(1) - \Gamma_2 a_2(1)] = 0.5$$

Thus, $\Gamma_1 = a_1(1) = 0.5$ and the reflection coefficient sequence is

$$\boldsymbol{\Gamma} = [0.5,\ 0.2,\ -0.5]^T$$

Therefore, a lattice filter having the given system function is as shown in Fig. 5.8.

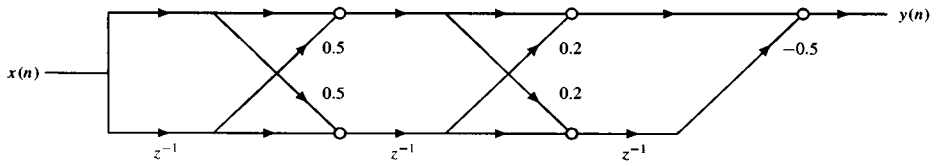


Figure 5.8 *A lattice filter implementation of the filter* $H(z) = 1 + 0.5z^{-1} - 0.1z^{-2} - 0.5z^{-3}$.

An interesting and useful application of the step-down recursion is the *Schur-Cohn stability test* for digital filters [27]. This test is based on Property 2 (p. 226), which states that the roots of a polynomial will lie inside the unit circle if and only if the magnitudes of the reflection coefficients are less than one. Therefore, given a causal, linear shift-invariant filter with a rational system function,

$$H(z) = \frac{B(z)}{A(z)}$$

the filter may be tested for stability as follows. First, the step-down recursion is applied to the coefficients of the denominator polynomial $A(z)$ to generate a reflection coefficient

sequence, Γ_j. The filter will then be stable if and only if all of the reflection coefficients are less than one in magnitude. The following example illustrates the procedure.

Example 5.2.5 *The Schur-Cohn Stability Test*

Let us use the Schur-Cohn stability test to check the stability of the filter

$$H(z) = \frac{1}{2 + 4z^{-1} - 3z^{-2} + z^{-3}}$$

First note that the leading coefficient in the denominator polynomial is not equal to one. Therefore, we begin by rewriting $H(z)$ as follows

$$H(z) = \frac{0.5}{1 + 2z^{-1} - 1.5z^{-2} + 0.5z^{-3}}$$

With $\Gamma_3 = a_3(3) = 0.5$ we then solve for the second-order polynomial $A_2(z) = 1 + a_2(1)z^{-1} + a_2(2)z^{-2}$ using Eq. (5.53) as follows

$$\begin{bmatrix} a_2(1) \\ a_2(2) \end{bmatrix} = \frac{1}{1 - \Gamma_3^2} \left\{ \begin{bmatrix} a_3(1) \\ a_3(2) \end{bmatrix} - \Gamma_3 \begin{bmatrix} a_3(2) \\ a_3(1) \end{bmatrix} \right\}$$

$$= \frac{4}{3} \left\{ \begin{bmatrix} 2 \\ -1.5 \end{bmatrix} - 0.5 \begin{bmatrix} -1.5 \\ 2 \end{bmatrix} \right\}$$

$$= \begin{bmatrix} 11/3 \\ -10/3 \end{bmatrix}$$

Therefore, since $\Gamma_2 = a_2(2) = -10/3$, then $|\Gamma_2| > 1$ and the filter is *unstable*.

5.2.5 The Inverse Levinson-Durbin Recursion

We have seen how to derive the reflection coefficients and the model parameters from an autocorrelation sequence. We have also derived recursions that map a sequence of reflection coefficients into a sequence of model parameters and vice versa. It is also possible to recursively compute the autocorrelation sequence from the reflection coefficients $\{\Gamma_1, \Gamma_2, \ldots, \Gamma_p\}$ and ϵ_p or from the coefficients of the pth-order model $a_p(k)$ and $b(0)$. The recursion for doing this is referred to as the *inverse Levinson recursion*. To see how the recursion works, assume that we are given the reflection coefficient sequence Γ_j for $j = 1, \ldots, p$ and the pth-order error ϵ_p. Since

$$\epsilon_p = r_x(0) \prod_{i=1}^{p} (1 - |\Gamma_i|^2) \tag{5.56}$$

then the first term in the autocorrelation sequence is

$$\boxed{r_x(0) = \frac{\epsilon_p}{\prod_{i=1}^{p} (1 - |\Gamma_i|^2)}} \tag{5.57}$$

Table 5.4 *The Inverse Levinson-Durbin Recursion*

1. Initialize the recursion
   (a) $r_x(0) = \epsilon_p / \prod_{i=1}^{p}(1 - |\Gamma_i|^2)$
   (b) $a_0(0) = 1$
2. For $j = 0, 1, \ldots, p-1$
   (a) For $i = 1, 2, \ldots, j$
   $a_{j+1}(i) = a_j(i) + \Gamma_{j+1} a_j^*(j - i + 1)$
   (b) $a_{j+1}(j+1) = \Gamma_{j+1}$
   (c) $r_x(j+1) = -\sum_{i=1}^{j+1} a_{j+1}(i) r_x(j+1-i)$
3. Done

This, along with the zeroth-order model

$$\boxed{\mathbf{a}_0 = 1} \tag{5.58}$$

initializes the recursion.

Now suppose that the first j terms of the autocorrelation sequence are known, along with the jth-order filter coefficients $a_j(i)$. We will now show how to find the next term in the sequence, $r_x(j+1)$. First we use the step-up recursion to find the coefficients $a_{j+1}(i)$ from Γ_{j+1} and $a_j(i)$. Then, setting $p = j+1$ and $k = j+1$ in Eq. (5.2) we have the following expression for $r_x(j+1)$:

$$\boxed{r_x(j+1) = -\sum_{i=1}^{j+1} a_{j+1}(i) r_x(j+1-i)} \tag{5.59}$$

Since the sum on the right only involves known autocorrelation values, this expression may be used to determine $r_x(j+1)$, which completes the recursion. The inverse Levinson-Durbin recursion, which may be represented pictorially as

$$\{\Gamma_1, \Gamma_2, \ldots, \Gamma_p, \epsilon_p\} \xrightarrow{(LEV)^{-1}} \{r_x(0), r_x(1), \ldots, r_x(p)\}$$

is summarized in Table 5.4.

Example 5.2.6 ***The Inverse Levinson-Durbin Recursion***

Given the sequence of reflection coefficients, $\Gamma_1 = \Gamma_2 = \Gamma_3 = \frac{1}{2}$ and a model error of $\epsilon_3 = 2(\frac{3}{4})^3$, let us find the autocorrelation sequence $\mathbf{r}_x = [r_x(0), r_x(1), r_x(2), r_x(3)]^T$. Initializing the recursion with

$$r_x(0) = \frac{\epsilon_3}{\prod_{i=1}^{3}(1 - \Gamma_i^2)} = 2$$

we begin by finding the first-order model, which is given by

$$\mathbf{a}_1 = \begin{bmatrix} 1 \\ \Gamma_1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0.5 \end{bmatrix}$$

Therefore,

$$r_x(1) = -r_x(0)\Gamma_1 = -1.0$$

Updating the model parameters using the step-up recursion we have

$$\mathbf{a}_2 = \begin{bmatrix} 1 \\ a_1(1) \\ 0 \end{bmatrix} + \Gamma_2 \begin{bmatrix} 0 \\ a_1(1) \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1/2 \\ 0 \end{bmatrix} + \frac{1}{2} \begin{bmatrix} 0 \\ 1/2 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 3/4 \\ 1/2 \end{bmatrix}$$

Thus,

$$r_x(2) = -a_2(1)r_x(1) - a_2(2)r_x(0) = 3/4 - 1 = -1/4$$

Applying the step-up recursion to $\mathbf{a}_2$ we have

$$\mathbf{a}_3 = \begin{bmatrix} 1 \\ a_2(1) \\ a_2(2) \\ 0 \end{bmatrix} + \Gamma_3 \begin{bmatrix} 0 \\ a_2(2) \\ a_2(1) \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 3/4 \\ 1/2 \\ 0 \end{bmatrix} + \frac{1}{2} \begin{bmatrix} 0 \\ 1/2 \\ 3/4 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 7/8 \\ 1/2 \end{bmatrix}$$

Finally, for $r_x(3)$ we have

$$r_x(3) = -a_3(1)r_x(2) - a_3(2)r_x(1) - a_3(3)r_x(0) = 1/4 + 7/8 - 1 = 1/8$$

and the autocorrelation sequence is

$$\mathbf{r}_x = [2, -1, -1/4, 1/8]^T$$

The inverse Levinson recursion described above provides a procedure for finding the autocorrelation sequence from the reflection coefficients and ϵ_p. If, instead of the reflection coefficients, we were given the filter coefficients $a_p(k)$, then the autocorrelation sequence may still be determined by using the step-down recursion to find the sequence of reflection coefficients and then using the inverse Levinson recursion. MATLAB programs for the inverse Levinson recursion to find $r_x(k)$ from either the reflection coefficients Γ_j or the filter coefficients $a_p(k)$ are given in Fig. 5.9.

In summary, combining the results of Sections 5.2.4 and 5.2.5 we see that there is an equivalence between three sets of parameters: the autocorrelation sequence $r_x(k)$, the all-pole model $a_p(k)$ and $b(0)$, and the reflection coefficients Γ_j along with ϵ_p. This equivalence is illustrated in Fig. 5.10, which shows how one set of parameters may be derived from another.

5.2.6 The Schur Recursion*

The Levinson-Durbin recursion provides an efficient solution to the autocorrelation normal equations. For a pth-order model, this recursion requires on the order of p^2 arithmetic operations compared to an order of p^3 operations for Gaussian elimination. Therefore, with a single processor that takes one unit of time for an arithmetic operation, the Levinson-Durbin recursion would require on the order of p^2 units of time to solve the normal equations. With the increasing use of VLSI technology and parallel processing architectures

The Inverse Levinson-Durbin Recursions

```
function r=gtor(gamma,epsilon)
%
   p=length(gamma);
   aa=gamma(1);
   r=[1 -gamma(1)];
   for j=2:p;
      aa=[aa;0]+gamma(j)*[conj(flipud(aa));1];
      r=[r -fliplr(r)*aa];
      end;
   if nargin == 2,
   r = r*epsilon/prod(1-abs(gamma).^2);
   end;

function r=ator(a,b)
%
   p=length(a)-1;
   gamma=atog(a);
   r=gtor(gamma);
   if nargin == 2,
   r = r*sqrt(b)/prod(1-abs(gamma).^2);
   end;
```

Figure 5.9 MATLAB *programs to find the autocorrelation sequence* $r_x(k)$ *from either the reflection coefficients* Γ_j *and the modeling error* ϵ_p*, or the filter coefficients* $a_p(k)$ *and* $b(0)$.

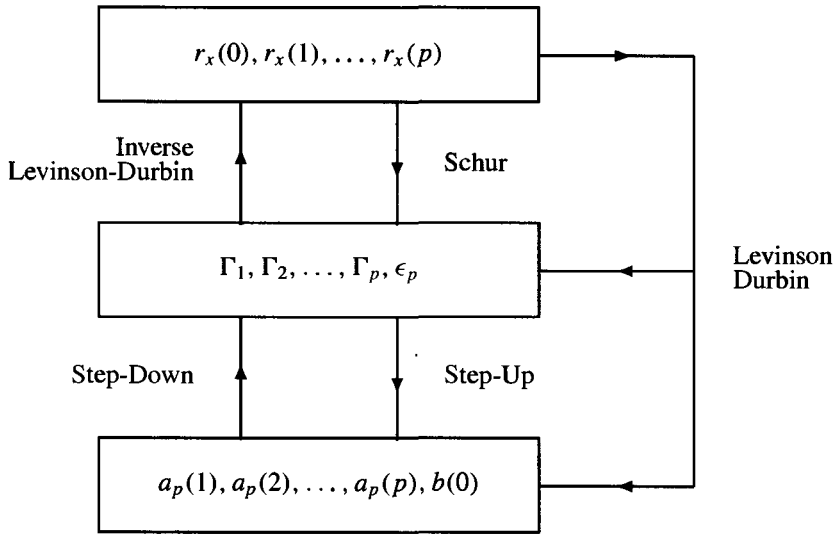


Figure 5.10 *The equivalence between the autocorrelation sequence, the all-pole model, and the reflection coefficients.*

in signal processing applications, an interesting question to ask is how much time would be necessary to solve the normal equations if we were to use p processors in parallel. Although one would hope that the computation time would be proportional to p rather than p^2, this is not the case. In fact, it is not difficult to see that with p processors the

Levinson-Durbin recursion would solve the normal equations in a time that is proportional to $p \log_2 p$ instead of p. The reason for this is the requirement to compute the inner product

$$\gamma_j = r_x(j+1) + \sum_{i=1}^{j} a_j(i) r_x(j+1-i) = [r_x(j+1),\ r_x(j),\ \ldots,\ r_x(1)] \begin{bmatrix} 1 \\ a_j(1) \\ \vdots \\ a_j(j) \end{bmatrix} \tag{5.60}$$

which is the primary computational bottleneck in the Levinson-Durbin recursion. Although the j multiplications may be performed in one unit of time using j processors, the j additions require $\log_2 j$ units of time. Therefore, with p processors, finding the solution to the pth-order normal equations requires on the order of $p \log_2 p$ units of time.

In this section, we derive another algorithm for solving the normal equations known as the *Schur recursion*. This recursion, originally presented in 1917 as a procedure for testing a polynomial to see if it is analytic and bounded in the unit disk [9, 22], avoids the inner product in the calculation of γ_j and is therefore better suited to parallel processing. Unlike the Levinson-Durbin recursion, which finds the filter coefficients $a_p(k)$ in addition to the reflection coefficients Γ_j, the Schur recursion only solves for the reflection coefficients. Thus, the Schur recursion is a mapping from an autocorrelation sequence $r_x(k)$ to a reflection coefficient sequence

$$\{r_x(0),\ r_x(1),\ \ldots, r_x(p)\} \xrightarrow{Schur} \{\Gamma_1,\ \Gamma_2,\ \ldots,\ \Gamma_p,\ \epsilon_p\} \tag{5.61}$$

By avoiding the computation of the filter coefficients, $a_p(k)$, the Schur recursion is slightly more efficient than the Levinson-Durbin recursion in terms of the number of multiplications, even in a single processor implementation of the algorithm.

The development of the Schur recursion begins with the autocorrelation normal equations for a jth-order model

$$r_x(k) + \sum_{l=1}^{j} a_j(l) r_x(k-l) = 0 \quad ; \quad k = 1, 2, \ldots, j \tag{5.62}$$

If we set $a_j(0) = 1$ then the jth-order normal equations may be written as

$$\sum_{l=0}^{j} a_j(l) r_x(k-l) = 0 \quad ; \quad k = 1, 2, \ldots, j \tag{5.63}$$

which we will refer to as the *orthogonality condition*. Note that the left side of Eq. (5.63) is the convolution of the finite length sequence $a_j(k)$ with the autocorrelation sequence $r_x(k)$. Therefore, let us define $g_j(k)$ to be the sequence that is formed by convolving $a_j(k)$ with $r_x(k)$,

$$\boxed{g_j(k) = \sum_{l=0}^{j} a_j(l) r_x(k-l) = a_j(k) * r_x(k)} \tag{5.64}$$

Thus, as shown in Fig. 5.11, $g_j(k)$ is the output of the jth-order prediction error filter $A_j(z)$ when the input is the autocorrelation sequence $r_x(k)$ and the orthogonality condition states

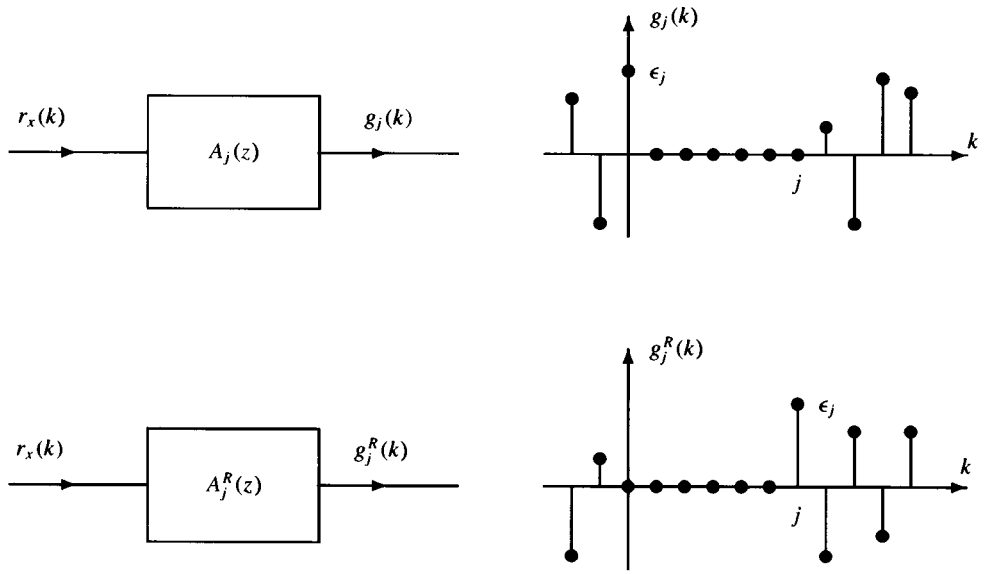


Figure 5.11 *The sequences $g_j(k)$ and $g_j^R(k)$ expressed as the output of the filter $A_j(z)$ and $A_j^R(z)$, respectively, when the input is the autocorrelation sequence $r_x(k)$. Due to the orthogonality condition, $g_j(k) = 0$ for $k = 1, \ldots, j$ and $g_j^R(k) = 0$ for $k = 0, \ldots, j-1$.*

that $g_j(k)$ is equal to zero for $k = 1, \ldots, j$,

$$g_j(k) = 0 \quad ; \quad k = 1, 2, \ldots, j \tag{5.65}$$

In addition, since

$$g_j(0) = \sum_{l=0}^{j} a_j(l) r_x(l) = \epsilon_j \tag{5.66}$$

where the last equality follows from Eq. (5.3), then $g_j(0)$ is equal to the jth-order modeling (prediction) error.

Now consider what happens when $a_j(k)$ is replaced with $a_j^R(k)$ and $g_j^R(k)$ is the convolution of $a_j^R(k)$ with $r_x(k)$,

$$\boxed{g_j^R(k) = \sum_{l=0}^{j} a_j^R(l) r_x(k-l) = a_j^R(k) * r_x(k)} \tag{5.67}$$

Thus, $g_j^R(k)$ is the response of the filter $A_j^R(z)$ to the input $r_x(k)$. Since $a_j^R(k) = a_j^*(j-k)$, then

$$g_j^R(k) = \sum_{l=0}^{j} a_j^*(j-l) r_x(k-l) = \sum_{l=0}^{j} a_j^*(l) r_x(k-j+l) \tag{5.68}$$

Using the conjugate symmetry of the autocorrelation sequence, Eq. (5.68) becomes

$$g_j^R(k) = \sum_{l=0}^{j} a_j^*(l) r_x^*([j-k]-l) = g_j^*(j-k) \tag{5.69}$$

Therefore, it follows from Eqs. (5.65) and (5.66) that

$$g_j^R(k) = 0 \quad ; \quad k = 0, 1, \ldots, j-1 \tag{5.70}$$

and

$$g_j^R(j) = \epsilon_j \tag{5.71}$$

These constraints on $g_j^R(k)$ are illustrated in Fig. 5.11.

The next step is to use the Levinson-Durbin recursion to show how the sequences $g_j(k)$ and $g_j^R(k)$ may be updated to form $g_{j+1}(k)$ and $g_{j+1}^R(k)$. Since $g_{j+1}(k) = a_{j+1}(k) * r_x(k)$, using the Levinson order-update equation for $a_{j+1}(k)$ we have

$$\begin{aligned} g_{j+1}(k) &= a_{j+1}(k) * r_x(k) = \left[a_j(k) + \Gamma_{j+1} a_j^R(k-1)\right] * r_x(k) \\ &= g_j(k) + \Gamma_{j+1} g_j^R(k-1) \end{aligned} \tag{5.72}$$

In a similar fashion, since $g_{j+1}^R(k) = a_{j+1}^R(k) * r_x(k)$, application of the Levinson order-update equation gives

$$\begin{aligned} g_{j+1}^R(k) &= a_{j+1}^R(k) * r_x(k) = \left[a_j^R(k-1) + \Gamma_{j+1}^* a_j(k)\right] * r_x(k) \\ &= g_j^R(k-1) + \Gamma_{j+1}^* g_j(k) \end{aligned} \tag{5.73}$$

Equations (5.72) and (5.73) are recursive update equations for $g_j(k)$ and $g_j^R(k)$ and are identical in form to the update equations given in Eq. (5.27) for $a_j(k)$ and $a_j^R(k)$. The only difference between the two recursions is in the initial condition. Specifically, for $a_j(k)$ we have $a_0(k) = a_0^R(k) = \delta(k)$, whereas for $g_j(k)$ we have $g_0(k) = g_0^R(k) = r_x(k)$. Taking the z-transform of Eqs. (5.72) and (5.73) and putting them in matrix form gives

$$\begin{bmatrix} G_{j+1}(z) \\ G_{j+1}^R(z) \end{bmatrix} = \begin{bmatrix} 1 & \Gamma_{j+1} z^{-1} \\ \Gamma_{j+1}^* & z^{-1} \end{bmatrix} \begin{bmatrix} G_j(z) \\ G_j^R(z) \end{bmatrix} \tag{5.74}$$

A lattice filter interpretation of these update equations is shown in Fig. 5.12.

Recall that our goal is to derive a recursion that will take a sequence of autocorrelation values and generate the corresponding sequence of reflection coefficients. Given the first j reflection coefficients, the lattice filter shown in Fig. 5.12 allows one to determine the sequences $g_j(k)$ and $g_j^R(k)$ from the autocorrelation sequence. All that remains is to derive a method to find the reflection coefficient, Γ_{j+1}, from $g_j(k)$ and $g_j^R(k)$. This may be done as follows. Since $g_{j+1}(j+1) = 0$, evaluating Eq. (5.72) for $k = j+1$ we have

$$g_{j+1}(j+1) = g_j(j+1) + \Gamma_{j+1} g_j^R(j) = 0$$

Therefore,

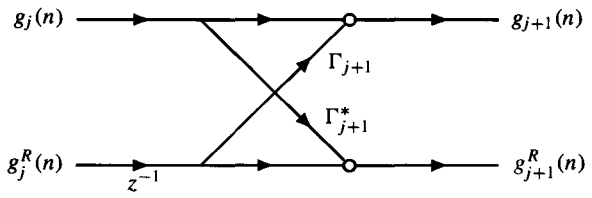
$$\boxed{\Gamma_{j+1} = -\frac{g_j(j+1)}{g_j^R(j)}} \tag{5.75}$$

and the recursion is complete. In summary, the Schur recursion begins by initializing $g_0(k)$ and $g_0^R(k)$ for $k = 1, 2, \ldots, p$ with $r_x(k)$,

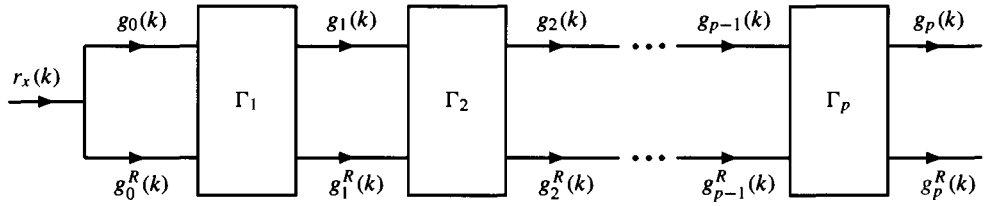
$$g_0(k) = g_0^R(k) = r_x(k)$$

The first reflection coefficient is then computed from the ratio

$$\Gamma_1 = -\frac{g_0(1)}{g_0^R(0)} = -\frac{r_x(1)}{r_x(0)}$$



(a) Lattice filter module for updating $g_j(k)$ and $g_j^R(k)$.



(b) A pth-order lattice filter for generating $g_p(k)$ and $g_p^R(k)$ from $r_x(k)$.

Figure 5.12 *The lattice filter used in the Schur recursion to generate the sequences $g_p(k)$ and $g_p^R(k)$ from the autocorrelation sequence.*

Given Γ_1, a lattice filter is then used to generate the sequences $g_1(k)$ and $g_1^R(k)$. The second reflection coefficient is then computed from the ratio

$$\Gamma_2 = -\frac{g_1(2)}{g_1^R(1)}$$

which, in turn, is used in the lattice filter to generate the sequences $g_2(k)$ and $g_2^R(k)$ and so on. In updating the sequences $g_{j+1}(k)$ and $g_{j+1}^R(k)$, note that since $g_{j+1}(k) = 0$ for $k = 1, 2, \ldots, j+1$ and $g_{j+1}^R(k) = 0$ for $k = 0, 1, \ldots, j$, not all p values need to be updated. Specifically, all that is required is to evaluate $g_{j+1}(k)$ for $k > j+1$ and $g_{j+1}^R(k)$ for $k > j$. The complete recursion is given in Table 5.5.

Additional insight into the operation of the Schur recursion may be gained if it is formulated using vector notation. Let $\mathbf{g}_j$ and $\mathbf{g}_j^R$ be the vectors that contain the sequences $g_j(k)$ and $g_j^R(k)$, respectively, and consider the $2 \times (p+1)$ matrix that has $\mathbf{g}_j$ in the first

Table 5.5 *The Schur Recursion*

1. Set $g_0(k) = g_0^R(k) = r_x(k)$ for $k = 0, 1, \ldots, p$.
2. For $j = 0, 1, \ldots, p-1$
   (a) Set $\Gamma_{j+1} = -g_j(j+1)/g_j^R(j)$
   (b) For $k = j+2, \ldots, p$
   $$g_{j+1}(k) = g_j(k) + \Gamma_{j+1} g_j^R(k-1)$$
   (c) For $k = j+1, \ldots, p$
   $$g_{j+1}^R(k) = g_j^R(k-1) + \Gamma_{j+1}^* g_j(k)$$
3. $\epsilon_p = g_p^R(p)$

row and $\mathbf{g}_j^R$ in the second row

$$\begin{bmatrix} \mathbf{g}_j^T \\ (\mathbf{g}_j^R)^T \end{bmatrix} = \begin{bmatrix} g_j(0) & g_j(1) & \cdots & g_j(p) \\ g_j^R(0) & g_j^R(1) & \cdots & g_j^R(p) \end{bmatrix}$$

From Eqs. (5.65), (5.66), and (5.70), we see that this matrix has the form[6]

$$\begin{bmatrix} \mathbf{g}_j^T \\ (\mathbf{g}_j^R)^T \end{bmatrix} = \begin{bmatrix} \epsilon_j & \cdots & 0 & 0 & g_j(j+1) & \cdots & g_j(p) \\ 0 & \cdots & 0 & g_j^R(j) & g_j^R(j+1) & \cdots & g_j^R(p) \end{bmatrix} \tag{5.76}$$

We are now going to perform a sequence of operations on this matrix that correspond to those expressed in the z-domain by Eq. (5.74). These operations will comprise one iteration of the Schur recursion. The first operation is to shift the second row of the matrix to the right by one with $g_j^R(-1)$ entering as the first element of the second row (this corresponds to a delay of $g_j^R(k)$ by one or a multiplication of $G_j^R(z)$ by z^{-1}). Note that after this shift, the ratio of the two terms in column number $(j+2)$ is equal to $-\Gamma_{j+1}$. Therefore, evaluating the reflection coefficient we may then form the matrix

$$\mathbf{\Theta}_{j+1} = \begin{bmatrix} 1 & \Gamma_{j+1} \\ \Gamma_{j+1}^* & 1 \end{bmatrix} \tag{5.77}$$

and multiply the shifted matrix by $\mathbf{\Theta}_{j+1}$ to generate the updated sequences $\mathbf{g}_{j+1}$ and $\mathbf{g}_{j+1}^R$ as follows

$$\begin{aligned} \begin{bmatrix} \mathbf{g}_{j+1}^T \\ (\mathbf{g}_{j+1}^R)^T \end{bmatrix} &= \begin{bmatrix} 1 & \Gamma_{j+1} \\ \Gamma_{j+1}^* & 1 \end{bmatrix} \begin{bmatrix} \epsilon_j & \cdots & 0 & g_j(j+1) & \cdots & g_j(p) \\ g_j^R(-1) & \cdots & 0 & g_j^R(j) & \cdots & g_j^R(p-1) \end{bmatrix} \\ &= \begin{bmatrix} \epsilon_{j+1} & \cdots & 0 & 0 & g_{j+1}(j+2) & \cdots & g_{j+1}(p) \\ 0 & \cdots & 0 & g_{j+1}^R(j+1) & g_{j+1}^R(j+2) & \cdots & g_{j+1}^R(p) \end{bmatrix} \end{aligned} \tag{5.78}$$

This completes one step of the recursion. We may, however, simplify these operations slightly by noting that the first column of the matrix in Eq. (5.76) never enters into the calculations. Therefore, we may suppress the evaluation of these entries by initializing the first element in the first column to zero and by bringing in a zero into the second row each time that it is shifted to the right. Note that since $\epsilon_j = g_j^R(j)$, then the error is not discarded with this simplification. In summary, the steps described above lead to the following matrix formulation of the Schur recursion. Beginning with

$$\mathbf{G}_0 = \begin{bmatrix} 0 & r_x(1) & r_x(2) & \cdots & r_x(p) \\ r_x(0) & r_x(1) & r_x(2) & \cdots & r_x(p) \end{bmatrix} \tag{5.79}$$

which is referred to as the *generator matrix*, a new matrix, $\widetilde{\mathbf{G}}_0$, is formed by shifting the second row of $\mathbf{G}_0$ to the right by one

$$\widetilde{\mathbf{G}}_0 = \begin{bmatrix} 0 & r_x(1) & r_x(2) & \cdots & r_x(p) \\ 0 & r_x(0) & r_x(1) & \cdots & r_x(p-1) \end{bmatrix}$$

Setting Γ_1 equal to the negative of the ratio of the two terms in the second column of $\widetilde{\mathbf{G}}_0$,

[6] We could also use Eq. (5.71) to set $g_j^R(j) = \epsilon_j$ in this matrix, but this would make the next step of computing the reflection coefficient Γ_{j+1} using Eq. (5.75) a little more obscure.

we then form the matrix

$$\Theta_1 = \begin{bmatrix} 1 & \Gamma_1 \\ \Gamma_1^* & 1 \end{bmatrix}$$

and evaluate $\mathbf{G}_1$ as follows

$$\mathbf{G}_1 = \Theta_1 \widetilde{\mathbf{G}}_0 = \begin{bmatrix} 0 & 0 & g_1(2) & \cdots & g_1(p) \\ 0 & g_1^R(1) & g_1^R(2) & \cdots & g_1^R(p) \end{bmatrix} \tag{5.80}$$

The recursion then repeats these three steps where, in general, at the jth step we

1. Shift the second row of $\mathbf{G}_j$ to the right by one,
2. Compute Γ_{j+1} as the negative of the ratio of the two terms in the $(j+2)$nd column,
3. Multiply $\widetilde{\mathbf{G}}_j$ by Θ_{j+1} to form $\mathbf{G}_{j+1}$.

The following example illustrates the procedure.

Example 5.2.7 *The Schur Recursion*

Given the autocorrelation sequence $\mathbf{r}_x = [2, -1, -1/4, 1/8]^T$, let us use the Schur recursion to find the reflection coefficients Γ_1, Γ_2, and Γ_3.

1. **Step 1:** We begin the Schur recursion by initializing the generator matrix $\mathbf{G}_0$ to

$$\mathbf{G}_0 = \begin{bmatrix} 0 & r_x(1) & r_x(2) & r_x(3) \\ r_x(0) & r_x(1) & r_x(2) & r_x(3) \end{bmatrix} = \begin{bmatrix} 0 & -1 & -1/4 & 1/8 \\ 2 & -1 & -1/4 & 1/8 \end{bmatrix}$$

2. **Step 2:** Forming the shifted matrix

$$\widetilde{\mathbf{G}}_0 = \begin{bmatrix} 0 & -1 & -1/4 & 1/8 \\ 0 & 2 & -1 & -1/4 \end{bmatrix}$$

it follows that $\Gamma_1 = 0.5$ and

$$\Theta_1 = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}$$

This step only requires one division.

3. **Step 3:** Forming the product $\Theta_1 \widetilde{\mathbf{G}}_0$ we find

$$\mathbf{G}_1 = \Theta_1 \widetilde{\mathbf{G}}_0 = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix} \begin{bmatrix} 0 & -1 & -1/4 & 1/8 \\ 0 & 2 & -1 & -1/4 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 0 & -3/4 & 0 \\ 0 & 3/2 & -9/8 & -3/16 \end{bmatrix}$$

Since we know that the second element in the first row of $\mathbf{G}_1$ will be equal to zero after multiplying $\widetilde{\mathbf{G}}_0$ by Θ_1, this step of the recursion only requires five multiplications.

4. **Step 4:** From the shifted matrix

$$\widetilde{\mathbf{G}}_1 = \begin{bmatrix} 0 & 0 & -3/4 & 0 \\ 0 & 0 & 3/2 & -9/8 \end{bmatrix}$$

we see that $\Gamma_2 = 0.5$ and

$$\Theta_2 = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}$$

Again, in this step, we have one division.

5. **Step 5:** Forming the product $\Theta_2\widetilde{\mathbf{G}}_1$ we have

$$\mathbf{G}_2 = \Theta_2\widetilde{\mathbf{G}}_1 = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}\begin{bmatrix} 0 & 0 & -3/4 & 0 \\ 0 & 0 & 3/2 & -9/8 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & -9/16 \\ 0 & 0 & 9/8 & -9/8 \end{bmatrix}$$

For this step we have three multiplications.

6. **Step 6:** Finally, forming the shifted matrix $\widetilde{\mathbf{G}}_2$,

$$\widetilde{\mathbf{G}}_2 = \begin{bmatrix} 0 & 0 & 0 & -9/16 \\ 0 & 0 & 0 & 9/8 \end{bmatrix}$$

we find that $\Gamma_3 = 0.5$,

$$\Theta_3 = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}$$

and

$$\mathbf{G}_3 = \Theta_3\widetilde{\mathbf{G}}_2 = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}\begin{bmatrix} 0 & 0 & 0 & -9/16 \\ 0 & 0 & 0 & 9/8 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 27/32 \end{bmatrix}$$

Therefore, the error is

$$\epsilon_3 = 27/32$$

In this last step, we have one division and one multiplication.

Counting the number of arithmetic operations we find that this recursion required 9 multiplications and 3 divisions.

Unlike the Levinson-Durbin recursion, the Schur recursion is well suited to parallel implementation. In particular, Kung and Hu [13] developed the pipelined structure shown in Fig. 5.13 for implementing the Schur recursion. This structure consists of a cascade of p modular cells, each containing an upper processing unit that computes the sequence values $g_j(k)$ and a lower processing unit that computes the values $g_j^R(k)$. The upper processing unit in cell 1 differs from the others in that it is a *divider cell* for computing the reflection coefficients Γ_j. The operation of this pipelined architecture is as follows.

1. The upper processing units are initialized with the autocorrelation values $r_x(1), \ldots, r_x(p)$ and the lower units with $r_x(0), \ldots, r_x(p-1)$ as shown in Fig. 5.13*a*. This step loads the cells with the values of the matrix $\widetilde{\mathbf{G}}_0$.
2. The divider cell computes the first reflection coefficient by dividing the contents in the upper unit of cell 1 with the contents of the lower unit,

$$\Gamma_1 = -r_x(1)/r_x(0)$$

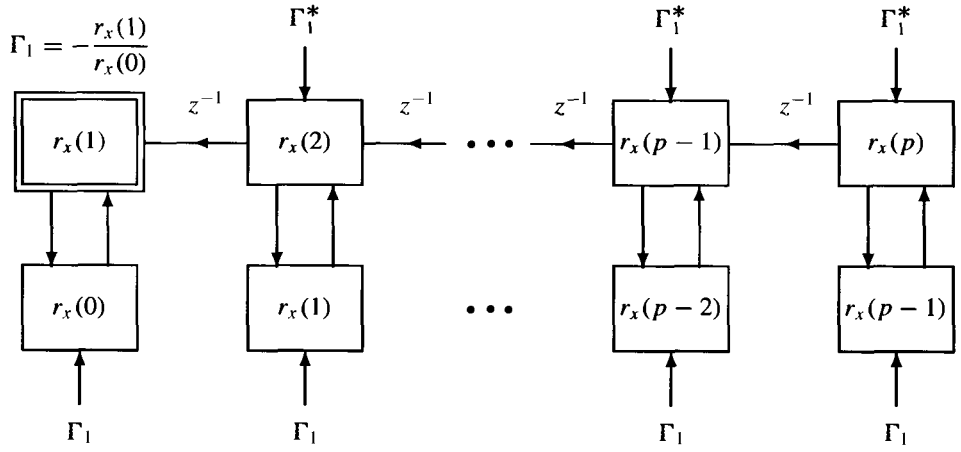
This reflection coefficient is then propagated simultaneously to each of the other cells.

3. The upper units of each cell are updated as follows

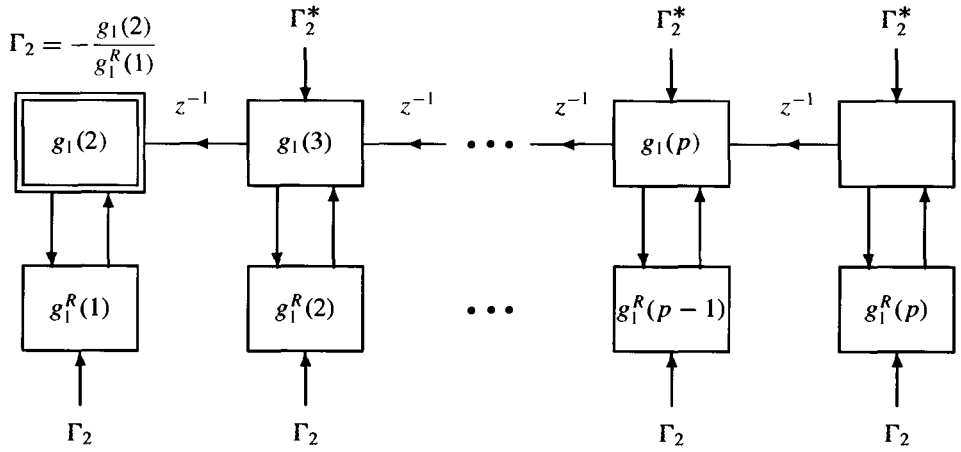
$$g_1(k) = r_x(k) + \Gamma_1 r_x(k-1) \quad ; \quad k = 1, \ldots, p$$

and the lower units updated as

$$g_1^R(k) = r_x(k-1) + \Gamma_1^* r_x(k) \quad ; \quad k = 1, \ldots, p$$



(a) Initialization of the processing units with the autocorrelation sequence $r_x(k)$. The divider cell computes the first reflection coefficient Γ_1 which is then simultaneously propagated to each of the other units.



(b) State of the Schur pipeline after the first update.

Figure 5.13 *A pipelined structure for implementing the Schur recursion.*

This operation corresponds to the updating of $\tilde{\mathbf{G}}_0$ by multiplying by Θ_1 to form the new generator matrix $\mathbf{G}_1$.

4. The contents of the upper units in each cell are then shifted to the left by one cell. Note that the last cell in the top row is left vacated.
5. The next reflection coefficient is then computed in the divider cell by dividing the contents in the upper unit of cell 1 with the contents of the lower unit. This reflection coefficient is then propagated simultaneously to each of the other cells. At this point, the pipeline structure is in the state shown in Fig. 5.13*b*.
6. The upper units of each cell are again updated using

$$g_{j+1}(k) = g_j(k) + \Gamma_{j+1} g_j^R(k-1)$$

and the lower units using

$$g_{j+1}^R(k) = g_j^R(k-1) + \Gamma_{j+1}^* g_j(k)$$

Steps 4, 5 and 6 are repeated until all of the reflection coefficients have been computed.

We conclude this discussion of the Schur recursion with an evaluation of the computational requirements of the algorithm. Note that at the jth stage of the recursion, one division is required to determine Γ_{j+1} and $p-j-1$ multiplications along with $p-j-1$ additions are necessary to evaluate the terms in the sequence $g_{j+1}(k)$ (step 2b in Table 5.5). In addition, $(p-j)$ multiplications and $(p-j)$ additions are necessary in order to evaluate the terms in the sequence $g_{j+1}^R(k)$ (step 2c in Table 5.5). With p steps in the recursion, the total number of multiplications and divisions is

$$p + 2\sum_{j=0}^{p-1}(p-j) - p = p^2 + p$$

along with p^2 additions. Thus, in comparing the Schur recursion to the Levinson-Durbin recursion (p. 223), we see that although the same number of additions are required with the Schur recursion, there are p fewer multiplications.

5.2.7 The Cholesky Decomposition

We have seen that the Levinson-Durbin recursion is an efficient algorithm for solving the autocorrelation normal equations. It may also be used to perform a Cholesky decomposition of the Hermitian Toeplitz autocorrelation matrix $\mathbf{R}_p$. The Cholesky (LDU) decomposition of a Hermitian matrix $\mathbf{C}$ is a factorization of the form

$$\mathbf{C} = \mathbf{L}\mathbf{D}\mathbf{L}^H \tag{5.81}$$

where $\mathbf{L}$ is a *lower triangular* matrix with ones along the diagonal and $\mathbf{D}$ is a *diagonal* matrix. If the diagonal terms of the matrix $\mathbf{D}$ are nonnegative ($\mathbf{C}$ is positive semidefinite), then $\mathbf{D}$ may be split into a product of two matrices by taking square roots

$$\mathbf{D} = \mathbf{D}^{1/2}\mathbf{D}^{1/2} \tag{5.82}$$

and the Cholesky decomposition of $\mathbf{C}$ may be expressed as the product of an upper triangular and a lower triangular matrix as follows

$$\mathbf{C} = \left(\mathbf{L}\mathbf{D}^{1/2}\right)\left(\mathbf{D}^{1/2}\mathbf{L}^H\right) \tag{5.83}$$

With a Cholesky decomposition of the autocorrelation matrix we will easily be able to establish the equivalence between the positive definiteness of $\mathbf{R}_p$, the positivity of the error sequence ϵ_j, and the unit magnitude constraint on the reflection coefficients Γ_j. In addition, we will be able to derive a closed-form expression for the inverse of the autocorrelation matrix as well as a recursive algorithm for inverting a Toeplitz matrix.

To derive the Cholesky decomposition of $\mathbf{R}_p$, consider the $(p+1) \times (p+1)$ *upper triangular* matrix

$$\mathbf{A}_p = \begin{bmatrix} 1 & a_1^*(1) & a_2^*(2) & \cdots & a_p^*(p) \\ 0 & 1 & a_2^*(1) & \cdots & a_p^*(p-1) \\ 0 & 0 & 1 & \cdots & a_p^*(p-2) \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix} \tag{5.84}$$

This matrix is formed from the vectors $\mathbf{a}_0, \mathbf{a}_1, \ldots, \mathbf{a}_p$ that are produced when the Levinson-Durbin recursion is applied to the autocorrelation sequence $r_x(0), \ldots, r_x(p)$. Note that the jth column of $\mathbf{A}_p$ contains the filter coefficients, $\mathbf{a}_{j-1}^R$, padded with zeros. Since

$$\mathbf{R}_j \mathbf{a}_j^R = \epsilon_j \mathbf{u}_j \tag{5.85}$$

where $\mathbf{u}_j = [0, 0, \ldots, 1]^T$ is a unit vector of length $j+1$ with a one in the final position, then

$$\mathbf{R}_p \mathbf{A}_p = \begin{bmatrix} \epsilon_0 & 0 & 0 & \cdots & 0 \\ * & \epsilon_1 & 0 & \cdots & 0 \\ * & * & \epsilon_2 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ * & * & * & \cdots & \epsilon_p \end{bmatrix} \tag{5.86}$$

which is a lower triangular matrix with the prediction errors ϵ_j along the diagonal (an asterisk is used to indicate those elements that are, in general, nonzero). Although we have not yet established Eq. (5.85), it may be inferred from the Hermitian Toeplitz property of $\mathbf{R}_p$. For example, since $\mathbf{J}\mathbf{R}_p^*\mathbf{J} = \mathbf{R}_p$ then

$$\mathbf{R}_j \mathbf{a}_j = (\mathbf{J}\mathbf{R}_j^* \mathbf{J})\mathbf{a}_j = \epsilon_j \mathbf{u}_1 \tag{5.87}$$

Multiplying on the left by $\mathbf{J}$ and using the fact that $\mathbf{J}^2 = \mathbf{I}$ and $\mathbf{J}\mathbf{a}_j = (\mathbf{a}_j^R)^*$, Eq. (5.85) follows from Eq. (5.87) after taking complex conjugates.

Since the product of two lower triangular matrices is a lower triangular matrix, if we multiply $\mathbf{R}_p\mathbf{A}_p$ on the left by the lower triangular matrix $\mathbf{A}_p^H$, then we obtain another lower triangular matrix, $\mathbf{A}_p^H \mathbf{R}_p \mathbf{A}_p$. Note that since the terms along the diagonal of $\mathbf{A}_p$ are equal to one, the diagonal of $\mathbf{A}_p^H \mathbf{R}_p \mathbf{A}_p$ will be the same as that of $\mathbf{R}_p\mathbf{A}_p$, and $\mathbf{A}_p^H \mathbf{R}_p \mathbf{A}_p$ will also have the form

$$\mathbf{A}_p^H \mathbf{R}_p \mathbf{A}_p = \begin{bmatrix} \epsilon_0 & 0 & 0 & \cdots & 0 \\ * & \epsilon_1 & 0 & \cdots & 0 \\ * & * & \epsilon_2 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ * & * & * & \cdots & \epsilon_p \end{bmatrix} \tag{5.88}$$

Even more importantly, however, is the observation that since $\mathbf{A}_p^H \mathbf{R}_p \mathbf{A}_p$ is Hermitian, then the matrix on the right side of Eq. (5.88) must also be Hermitian. Therefore, the terms below the diagonal are zero and

$$\boxed{\mathbf{A}_p^H \mathbf{R}_p \mathbf{A}_p = \mathbf{D}_p} \tag{5.89}$$

where $\mathbf{D}_p$ is a diagonal matrix

$$\mathbf{D}_p = \text{diag}\{\epsilon_0, \epsilon_1, \ldots, \epsilon_p\}$$

Since $\mathbf{A}_p^H$ is a lower triangular matrix with ones along the diagonal, then $\mathbf{A}_p^H$ is nonsingular with $\det(\mathbf{A}_p) = 1$ and the inverse of $\mathbf{A}_p^H$ is also a lower triangular matrix. Denoting the inverse of $\mathbf{A}_p^H$ by $\mathbf{L}_p$, if we multiply Eq. (5.89) on the left by $\mathbf{L}_p$ and on the right by $\mathbf{L}_p^H$

we have

$$\boxed{\mathbf{R}_p = \mathbf{L}_p \mathbf{D}_p \mathbf{L}_p^H} \tag{5.90}$$

which is the desired factorization.

An interesting and important property that may be established from Eq. (5.89) is that the determinant of the autocorrelation matrix $\mathbf{R}_p$ is equal to the product of the modeling errors,

$$\boxed{\det \mathbf{R}_p = \prod_{k=0}^{p} \epsilon_k} \tag{5.91}$$

To see this, note that if we take the determinant of both sides of Eq. (5.89) and use the fact that $\det(\mathbf{A}_p) = 1$, then

$$\begin{aligned} \det \mathbf{D}_p &= \det \left(\mathbf{A}_p^H \mathbf{R}_p \mathbf{A}_p \right) \\ &= \left(\det \mathbf{A}_p^H \right) \left(\det \mathbf{R}_p \right) \left(\det \mathbf{A}_p \right) \\ &= \det \mathbf{R}_p \end{aligned} \tag{5.92}$$

and Eq. (5.91) follows since

$$\det \mathbf{D}_p = \prod_{k=0}^{p} \epsilon_k$$

Using Eq. (5.91) we may also show that the positivity of the autocorrelation matrix $\mathbf{R}_p$ is tied to the positivity of the error sequence ϵ_k. In particular, note that $\mathbf{R}_p$ will be positive definite if and only if

$$\det \mathbf{R}_j > 0 \quad ; \quad j = 0, 1, \ldots, p \tag{5.93}$$

Therefore, since

$$\det \mathbf{R}_j = \prod_{k=0}^{j} \epsilon_k \tag{5.94}$$

it follows that $\mathbf{R}_p$ will be positive definite if and only if $\epsilon_k > 0$ for $k = 0, 1, \ldots, p$. Furthermore, $\mathbf{R}_p$ will be singular, $\det(\mathbf{R}_p) = 0$, if $\epsilon_k = 0$ for some k. This relationship between the positive definiteness of $\mathbf{R}_p$ and the positivity of the error sequence ϵ_k may also be established using Sylvester's *law of inertia* [21], which states that if $\mathbf{A}_p$ is a nonsingular matrix, then $\mathbf{A}_p^H \mathbf{R}_p \mathbf{A}_p$ has the same number of positive eigenvalues as $\mathbf{R}_p$, the same number of negative eigenvalues, and the same number of zero eigenvalues. Since $\mathbf{A}_p^H \mathbf{R}_p \mathbf{A}_p = \mathbf{D}_p$ with $\mathbf{A}_p$ being nonsingular, then $\mathbf{R}_p$ and $\mathbf{D}_p$ have the same number of positive, negative, and zero eigenvalues. Since the eigenvalues of $\mathbf{D}_p$ are equal to ϵ_k it follows that $\mathbf{R}_p > 0$ if and only if $\epsilon_k > 0$ for all k. In summary, we have established the following fundamental property of Hermitian Toeplitz matrices.

Property 7. For any $(p+1) \times (p+1)$ Hermitian Toeplitz matrix $\mathbf{R}_p$, the following are equivalent:

1. $\mathbf{R}_p > 0$
2. $\epsilon_j > 0, \quad j = 1, \ldots, p$
3. $|\Gamma_j| < 1, \quad j = 1, \ldots, p$

Furthermore, this equivalence remains valid if the strict inequalities are replaced with less than or equal to inequalities.

In the following example we use this property to show how a Toeplitz matrix may be tested to see whether or not it is positive definite.

Example 5.2.8 ***Testing for Positive Definiteness***

Consider the 3×3 symmetric Toeplitz matrix

$$\mathbf{R} = \begin{bmatrix} 1 & \alpha & \beta \\ \alpha & 1 & \alpha \\ \beta & \alpha & 1 \end{bmatrix}$$

Let us find the values of α and β for which $\mathbf{R}$ is positive definite.

The first row of $\mathbf{R}$ may be considered to be the first three terms of an autocorrelation sequence, $\mathbf{r} = [1, \alpha, \beta]^T$. Using the Levinson-Durbin recursion we may express the values of the reflection coefficients, Γ_1 and Γ_2, in terms of α and β. Since

$$\Gamma_1 = -r(1)/r(0) = -\alpha$$

then

$$\mathbf{a}_1 = [1, -\alpha]^T \quad \text{and} \quad \epsilon_1 = 1 - \alpha^2$$

Therefore,

$$\gamma_1 = r_x(2) + a_1(1) r_x(1) = \beta - \alpha^2$$

and

$$\Gamma_2 = -\frac{\gamma_1}{\epsilon_1} = \frac{\alpha^2 - \beta}{1 - \alpha^2}$$

Now, in order for $\mathbf{R}$ to be positive definite it is necessary that $|\Gamma_1| < 1$ and $|\Gamma_2| < 1$, which implies that

$$\text{(i)} \qquad |\alpha| < 1$$

$$\text{(ii)} \qquad \left| \frac{\alpha^2 - \beta}{1 - \alpha^2} \right| < 1$$

From the second condition, it follows that

$$-1 < \frac{\alpha^2 - \beta}{1 - \alpha^2} < 1$$

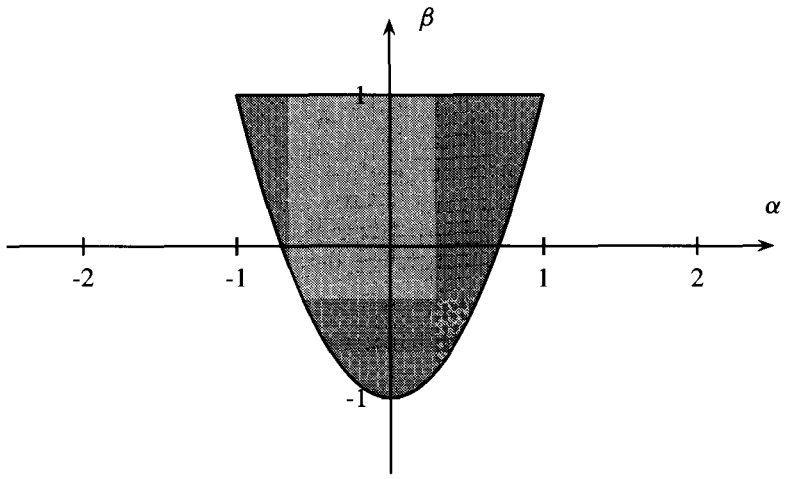


Figure 5.14 *The values for α and β for which the autocorrelation matrix associated with the autocorrelation sequence $\mathbf{r} = [1, \alpha, \beta]^T$ is positive definite.*

which, since $|\alpha| < 1$, becomes

$$\alpha^2 - 1 < \alpha^2 - \beta < 1 - \alpha^2$$

The left-hand inequality gives

$$\beta < 1$$

and the right-hand inequality is equivalent to

$$\beta > 2\alpha^2 - 1$$

Therefore, the allowable values for α and β lie within the shaded region shown in Fig. 5.14.

5.2.8 The Autocorrelation Extension Problem

In this section, we consider the *autocorrelation extension problem*, which addresses the following question: Given the first $(p + 1)$ values of an autocorrelation sequence, $r_x(k)$ for $k = 0, 1, \ldots, p$, how may we extend (extrapolate) this *partial autocorrelation sequence* for $k > p$ in such a way that the extrapolated sequence is a valid autocorrelation sequence? In Section 3.3.5 of Chapter 3 we saw that in order for a finite sequence of numbers to be a valid partial autocorrelation sequence, the autocorrelation matrix formed from this sequence must be nonnegative definite, $\mathbf{R}_p \geq 0$. Therefore, any extension must preserve this nonnegative definite property, i.e., $\mathbf{R}_{p+1} \geq 0$, and $\mathbf{R}_{p+2} \geq 0$, and so on. In Section 3.6.1 of Chapter 3 we discovered one possible way to extrapolate a partial autocorrelation sequence. Specifically, given $r_x(k)$ for $k = 0, 1, \ldots, p$, if $r_x(k)$ is extrapolated for $k > p$ according to the recursion

$$r_x(k) = \sum_{l=1}^{p} a_p(l) r_x(k - l) \tag{5.95}$$

where the coefficients $a_p(l)$ are the solution to the Yule-Walker (normal) equations

$$\mathbf{R}_p \mathbf{a}_p = \epsilon_p \mathbf{u}_1 \tag{5.96}$$

then the resulting sequence will be a valid autocorrelation sequence. This follows from the fact that this extrapolation generates the autocorrelation sequence of the AR(p) process that is consistent with the given autocorrelation values, i.e., $r_x(k)$ for $|k| \leq p$. Thus, a finite sequence of numbers is *extendible* if and only if $\mathbf{R}_p \geq 0$ and, if extendible, one possible extension is given by Eq. (5.95). We have not yet addressed, however, the question of whether or not other extensions are possible. Equivalently, given a partial autocorrelation sequence $r_x(k)$ for $k = 0, 1, \ldots, p$, what values of $r_x(p+1)$ will produce a valid partial autocorrelation sequence with $\mathbf{R}_{p+1} \geq 0$? The answers to these questions may be deduced from Property 7 (p. 253), which states that if $\mathbf{R}_p$ is nonnegative definite, then $\mathbf{R}_{p+1}$ will be nonnegative definite if and only if $|\Gamma_{p+1}| \leq 1$. Therefore, by expressing $r_x(p+1)$ in terms of the reflection coefficient Γ_{p+1} we may place a bound on the allowable values for $r_x(p+1)$. The desired relationship follows from Eqs. (5.10) and (5.13), which gives, for $r_x(p+1)$,

$$r_x(p+1) = -\Gamma_{p+1}\epsilon_p - \sum_{k=1}^{p} a_p(k) r_x(p-k+1) \tag{5.97}$$

With Γ_{p+1} a complex number that is bounded by one in magnitude, $|\Gamma_{p+1}| \leq 1$, Eq. (5.97) implies that $r_x(p+1)$ is constrained to lie within or on a circle of radius ϵ_p that is centered at $C = -\sum_{k=1}^{p} a_p(k) r_x(p-k+1)$, as shown in Fig. 5.15. In the real case, the range of allowable values is

$$-\epsilon_p - \sum_{k=1}^{p} a_p(k) r_x(p-k+1) \leq r_x(p+1) \leq \epsilon_p - \sum_{k=1}^{p} a_p(k) r_x(p-k+1) \tag{5.98}$$

There are two special cases of autocorrelation sequence extension that are worth some discussion. The first occurs when we set $\Gamma_{p+1} = 0$. In this case, the extrapolation is performed according to Eq. (5.95) with

$$r_x(p+1) = \sum_{l=1}^{p} a_p(l) r_x(p+1-l)$$

Furthermore, if the extrapolation of $r_x(k)$ is continued in this manner for all $k > p$, then the resulting autocorrelation sequence will correspond to an AR(p) process that has a power

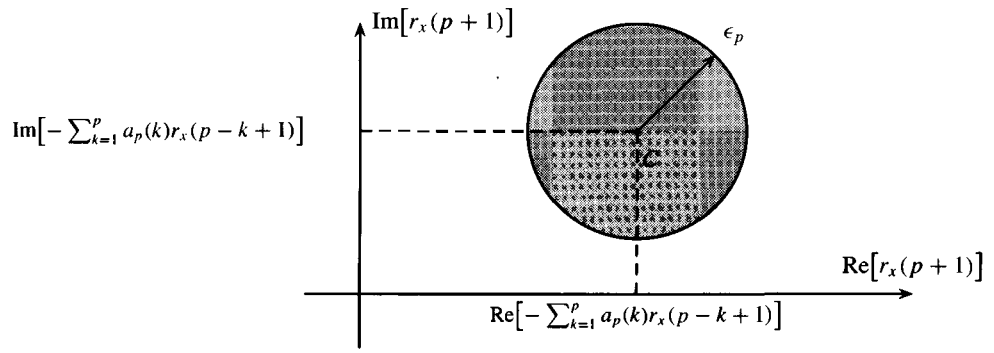


Figure 5.15 *The allowable values for the autocorrelation coefficient $r_x(p+1)$ given the partial autocorrelation sequence $r_x(k)$ for $k = 0, 1, \ldots, p$ are shown to lie within the circle of radius ϵ_p centered at C.*

spectrum equal to

$$P_x(e^{j\omega}) = \frac{|b(0)|^2}{\left|1 + \sum_{k=1}^{p} a_p(k)e^{-jk\omega}\right|^2} \tag{5.99}$$

The second case occurs when $|\Gamma_{p+1}| = 1$. When the magnitude of Γ_{p+1} is equal to one, $r_x(p+1)$ lies *on* the circle shown in Fig. 5.15, the $(p+1)$st-order model error is zero,

$$\epsilon_{p+1} = \epsilon_p\left[1 - |\Gamma_{p+1}|^2\right] = 0 \tag{5.100}$$

the matrix $\mathbf{R}_{p+1}$ is singular, and the predictor polynomial $A_{p+1}(z)$ has all of its roots on the unit circle (see Property 5, p. 230). As we will see in Section 8.6 of Chapter 8, this case will be important for frequency estimation and for modeling signals as a sum of complex exponentials.

Example 5.2.9 *Autocorrelation Extension*

Given the partial autocorrelation sequence $r_x(0) = 1$ and $r_x(1) = 0.5$, let us find the set of allowable values for $r_x(2)$, assuming that $r_x(2)$ is real. For the first-order model we have

$$\Gamma_1 = -\frac{r_x(1)}{r_x(0)} = -0.5$$

and $a_1(1) = \Gamma_1 = -0.5$. Thus, with a first-order modeling error of

$$\epsilon_1 = r_x(0)\left[1 - |\Gamma_1|^2\right] = 0.75$$

we have

$$r_x(2) = -\Gamma_2\epsilon_1 - a_1(1)r_x(1) = -0.75\Gamma_2 + 0.25$$

Therefore, with $|\Gamma_2| \leq 1$ it follows that

$$-0.5 \leq r_x(2) \leq 1$$

In the special case of $\Gamma_2 = 0$, $r_x(2) = 0.25$ and, in the extreme cases of $\Gamma_2 = \pm 1$, the autocorrelation values are $r_x(2) = -0.5$ and $r_x(2) = 1$.

5.2.9 Inverting a Toeplitz Matrix*

In this section, we will derive a recursive algorithm for inverting a Toeplitz matrix $\mathbf{R}_p$ and show how this recursion may be used to derive the Levinson recursion for solving a general set of Toeplitz equations

$$\mathbf{R}_p\mathbf{x} = \mathbf{b}$$

We begin by showing how the decomposition derived in Section 5.2.7 may be used to express the inverse of a Toeplitz matrix in terms of the vectors $\mathbf{a}_j$ and the errors ϵ_j. The Levinson-Durbin recursion will then be applied to this expression for $\mathbf{R}_p^{-1}$ to derive the *Toeplitz matrix inversion recursion.*

Let $\mathbf{R}_p$ be a nonsingular Hermitian Toeplitz matrix. Using the decomposition given in Eq. (5.89), taking the inverse of both sides, we have

$$\left(\mathbf{A}_p^H\mathbf{R}_p\mathbf{A}_p\right)^{-1} = \mathbf{A}_p^{-1}\mathbf{R}_p^{-1}\mathbf{A}_p^{-H} = \mathbf{D}_p^{-1} \tag{5.101}$$

Multiplying both sides of this equation by $\mathbf{A}_p$ on the left and by $\mathbf{A}_p^H$ on the right gives the

desired expression for $\mathbf{R}_p^{-1}$,

$$\mathbf{R}_p^{-1} = \mathbf{A}_p \mathbf{D}_p^{-1} \mathbf{A}_p^H \tag{5.102}$$

Since $\mathbf{D}_p$ is a diagonal matrix, $\mathbf{D}_p^{-1}$ is easily computed. Therefore, finding the inverse of $\mathbf{R}_p$ simply involves applying the Levinson-Durbin recursion to the sequence $r_x(0), \ldots, r_x(p)$, forming the matrix $\mathbf{A}_p$, and performing the matrix product in Eq. (5.102).

Example 5.2.10 *Inverting a Toeplitz Matrix*

Let us find the inverse of the 3×3 autocorrelation matrix

$$\mathbf{R}_2 = \begin{bmatrix} 2 & 0 & 1 \\ 0 & 2 & 0 \\ 1 & 0 & 2 \end{bmatrix}$$

We begin by using the Levinson-Durbin recursion to find the vectors $\mathbf{a}_j$ and the errors ϵ_j for $j = 0, 1, 2$. For $j = 0$, we have $\mathbf{a}_0 = 1$ and $\epsilon_0 = r_x(0) = 2$. With

$$\Gamma_1 = -r_x(1)/r_x(0) = 0$$

the first-order model is

$$\mathbf{a}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

with an error

$$\epsilon_1 = \epsilon_0 \left[1 - \Gamma_1^2\right] = 2$$

For the second-order model,

$$\gamma_1 = r_x(2) + a_1(1) r_x(1) = 1$$

and

$$\Gamma_2 = -\frac{\gamma_1}{\epsilon_1} = -1/2$$

Thus,

$$\mathbf{a}_2 = \begin{bmatrix} 1 \\ 0 \\ -1/2 \end{bmatrix}$$

and

$$\epsilon_2 = \epsilon_1 \left[1 - \Gamma_2^2\right] = 3/2$$

Finally, using Eq. (5.102) we have for the inverse of $\mathbf{R}_2$,

$$\mathbf{R}_2^{-1} = \mathbf{A}_2 \mathbf{D}_2^{-1} \mathbf{A}_2^T = \begin{bmatrix} 1 & 0 & -1/2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1/2 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & 2/3 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1/2 & 0 & 1 \end{bmatrix}$$

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

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

Note that the inverse of a Toeplitz matrix is not, in general, Toeplitz. However, as discussed in Section 2.3.7 (p. 38) and as illustrated in this example, the inverse of a symmetric Toeplitz matrix is centrosymmetric [17].

In the next example, Eq. (5.102) is used to find the inverse autocorrelation matrix of a first-order autoregressive process.

Example 5.2.11 ***The Inverse Autocorrelation Matrix for an AR(1) Process***

Let $\mathbf{R}_p$ be the autocorrelation matrix of an AR(1) process that is generated by the difference equation

$$x(n) = \alpha x(n-1) + w(n)$$

where $|\alpha| < 1$ and $w(n)$ is zero mean white noise with a variance σ_w^2. Thus, the autocorrelation sequence of $x(n)$ is

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

and the autocorrelation matrix is

$$\mathbf{R}_p = \frac{\sigma_w^2}{1-\alpha^2}\begin{bmatrix} 1 & \alpha & \alpha^2 & \cdots & \alpha^p \\ \alpha & 1 & \alpha & \cdots & \alpha^{p-1} \\ \alpha^2 & \alpha & 1 & \cdots & \alpha^{p-2} \\ \vdots & \vdots & \vdots & & \vdots \\ \alpha^p & \alpha^{p-1} & \alpha^{p-2} & \cdots & 1 \end{bmatrix}$$

From Example 5.2.2 (p. 221) we know that the jth-order model for $x(n)$ is

$$\mathbf{a}_j = [1, -\alpha, 0, \ldots, 0]^T$$

and that the sequence of modeling errors is

$$\epsilon_0 = \frac{\sigma_w^2}{1-\alpha^2}$$

and

$$\epsilon_k = \sigma_w^2 \quad ; \quad k \geq 1$$

Therefore,

$$\mathbf{A}_p = \begin{bmatrix} 1 & -\alpha & 0 & \cdots & 0 & 0 \\ 0 & 1 & -\alpha & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & -\alpha \\ 0 & 0 & 0 & \cdots & 0 & 1 \end{bmatrix}$$

and

$$\mathbf{D}_p^{-1} = \text{diag}\left\{\epsilon_0^{-1}, \epsilon_1^{-1}, \ldots, \epsilon_{p-1}^{-1}, \epsilon_p^{-1}\right\} = \frac{1}{\sigma_w^2}\text{diag}\left\{(1-\alpha^2),\ 1,\ 1, \ldots,\ 1\right\}$$

From Eq. (5.102) it follows that

$$\mathbf{R}_p^{-1} = \frac{1}{\sigma_w^2}\begin{bmatrix} 1 & -\alpha & 0 & \cdots & 0 & 0 \\ 0 & 1 & -\alpha & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & -\alpha \\ 0 & 0 & 0 & \cdots & 0 & 1 \end{bmatrix}\begin{bmatrix} 1-\alpha^2 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & 0 & 1 \end{bmatrix}$$

$$\times\begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ -\alpha & 1 & \cdots & 0 & 0 \\ 0 & -\alpha & \cdots & 0 & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & -\alpha & 1 \end{bmatrix}$$

$$= \frac{1}{\sigma_w^2}\begin{bmatrix} 1-\alpha^2 & -\alpha & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & -\alpha \\ 0 & 0 & \cdots & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ -\alpha & 1 & \cdots & 0 & 0 \\ 0 & -\alpha & \cdots & 0 & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & -\alpha & 1 \end{bmatrix}$$

$$= \frac{1}{\sigma_w^2}\begin{bmatrix} 1 & -\alpha & \cdots & 0 & 0 \\ -\alpha & 1+\alpha^2 & \cdots & 0 & 0 \\ 0 & -\alpha & \cdots & 0 & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \cdots & 1+\alpha^2 & -\alpha \\ 0 & 0 & \cdots & -\alpha & 1 \end{bmatrix}$$

Thus, we see that $\mathbf{R}_p^{-1}$ is centrosymmetric and *tri-diagonal.*

Equation (5.102) provides a closed form expression for the inverse of a Hermitian Toeplitz matrix in terms of the matrices $\mathbf{A}_p$ and $\mathbf{D}_p$. However, since these matrices may be built up recursively using the Levinson-Durbin recursion, we may fold this recursion into the evaluation of Eq. (5.102) and derive a method for recursively computing the inverse of $\mathbf{R}_p$. In particular, given

$$\mathbf{R}_j^{-1} = \mathbf{A}_j\mathbf{D}_j^{-1}\mathbf{A}_j^H \tag{5.103}$$

we may find

$$\mathbf{R}_{j+1}^{-1} = \mathbf{A}_{j+1}\mathbf{D}_{j+1}^{-1}\mathbf{A}_{j+1}^H \tag{5.104}$$

To see how this may be accomplished, let us partition $\mathbf{A}_{j+1}$ and $\mathbf{D}_{j+1}$ as follows[7]

$$\mathbf{A}_{j+1} = \left[\begin{array}{c|c} \mathbf{A}_j & \begin{array}{c} a^*_{j+1}(j+1) \\ \vdots \\ a^*_{j+1}(1) \end{array} \\ \hline 0 \quad \cdots \quad 0 & 1 \end{array}\right] = \left[\begin{array}{c|c} \mathbf{A}_j & \bar{\mathbf{a}}^R_{j+1} \\ \hline \mathbf{0}^T & 1 \end{array}\right] \tag{5.105}$$

and

$$\mathbf{D}^{-1}_{j+1} = \left[\begin{array}{c|c} \mathbf{D}^{-1}_j & \begin{array}{c} 0 \\ \vdots \\ 0 \end{array} \\ \hline 0 \quad \cdots \quad 0 & \epsilon^{-1}_{j+1} \end{array}\right] \tag{5.106}$$

Incorporating Eqs. (5.105) and (5.106) into Eq. (5.104) we have

$$\begin{aligned} \mathbf{R}^{-1}_{j+1} &= \left[\begin{array}{c|c} \mathbf{A}_j & \bar{\mathbf{a}}^R_{j+1} \\ \hline \mathbf{0}^T & 1 \end{array}\right] \left[\begin{array}{c|c} \mathbf{D}^{-1}_j & \mathbf{0} \\ \hline \mathbf{0}^T & \epsilon^{-1}_{j+1} \end{array}\right] \left[\begin{array}{c|c} \mathbf{A}^H_j & \mathbf{0} \\ \hline (\bar{\mathbf{a}}^R_{j+1})^H & 1 \end{array}\right] \\ &= \left[\begin{array}{c|c} \mathbf{A}_j & \bar{\mathbf{a}}^R_{j+1} \\ \hline \mathbf{0}^T & 1 \end{array}\right] \left[\begin{array}{c|c} \mathbf{D}^{-1}_j \mathbf{A}^H_j & \mathbf{0} \\ \hline \epsilon^{-1}_{j+1}(\bar{\mathbf{a}}^R_{j+1})^H & \epsilon^{-1}_{j+1} \end{array}\right] \\ &= \left[\begin{array}{c|c} \mathbf{A}_j \mathbf{D}^{-1}_j \mathbf{A}^H_j + \epsilon^{-1}_{j+1} \bar{\mathbf{a}}^R_{j+1}(\bar{\mathbf{a}}^R_{j+1})^H & \epsilon^{-1}_{j+1} \bar{\mathbf{a}}^R_{j+1} \\ \hline \epsilon^{-1}_{j+1}(\bar{\mathbf{a}}^R_{j+1})^H & \epsilon^{-1}_{j+1} \end{array}\right] \end{aligned} \tag{5.107}$$

Splitting this expression for $\mathbf{R}^{-1}_{j+1}$ into a sum as follows

$$\mathbf{R}^{-1}_{j+1} = \left[\begin{array}{c|c} \mathbf{A}_j \mathbf{D}^{-1}_j \mathbf{A}^H_j & \mathbf{0} \\ \hline \mathbf{0}^T & 0 \end{array}\right] + \epsilon^{-1}_{j+1} \left[\begin{array}{c|c} \bar{\mathbf{a}}^R_{j+1}(\bar{\mathbf{a}}^R_{j+1})^H & \bar{\mathbf{a}}^R_{j+1} \\ \hline (\bar{\mathbf{a}}^R_{j+1})^H & 1 \end{array}\right] \tag{5.108}$$

we see that $\mathbf{R}^{-1}_j$ appears in the upper-left corner of the first term and that the second term may be written as an outer product

$$\begin{aligned} \epsilon^{-1}_{j+1} \left[\begin{array}{c|c} \bar{\mathbf{a}}^R_{j+1}(\bar{\mathbf{a}}^R_{j+1})^H & \bar{\mathbf{a}}^R_{j+1} \\ \hline (\bar{\mathbf{a}}^R_{j+1})^H & 1 \end{array}\right] &= \epsilon^{-1}_{j+1} \left[\begin{array}{c} \bar{\mathbf{a}}^R_{j+1} \\ \hline 1 \end{array}\right] \left[\begin{array}{c|c} (\bar{\mathbf{a}}^R_{j+1})^H & 1 \end{array}\right] \\ &= \frac{1}{\epsilon_{j+1}} \mathbf{a}^R_{j+1}(\mathbf{a}^R_{j+1})^H \end{aligned} \tag{5.109}$$

[7]**Note to the reader:** Although there is nothing difficult about the following derivation from a mathematical point of view, from a notational point of view it is a bit complex. Since there is not much to be gained in going through this derivation, other than a feeling of accomplishment, the reader may wish to jump to the main results which are contained in Eqs. (5.110) and (5.114). Although the discussion following Eq. (5.114) uses these results to derive the general Levinson recursion, this recursion is re-derived in the following section, without using these results.

Therefore, we have

$$\mathbf{R}_{j+1}^{-1} = \left[\begin{array}{c|c} \mathbf{R}_j^{-1} & \mathbf{0} \\ \hline \mathbf{0}^T & 0 \end{array}\right] + \frac{1}{\epsilon_{j+1}} \mathbf{a}_{j+1}^R (\mathbf{a}_{j+1}^R)^H \tag{5.110}$$

which is the desired update equation for $\mathbf{R}_j^{-1}$. The recursion is initialized by setting $\mathbf{R}_0^{-1} = 1$, $\epsilon_0 = r_x(0)$, and $\mathbf{a}_0 = 1$. Then, from $\mathbf{R}_j^{-1}$, ϵ_j, and $\mathbf{a}_j$, the Levinson-Durbin recursion is used to determine $\mathbf{a}_{j+1}$ and ϵ_{j+1} which, in turn, are then used in Eq. (5.110) to perform the update of $\mathbf{R}_j^{-1}$. The following example illustrates the procedure.

Example 5.2.12 *Recursive Evaluation of an Inverse Autocorrelation Matrix*

In this example we show how to recursively compute the inverse of the 3×3 autocorrelation matrix

$$\mathbf{R}_2 = \begin{bmatrix} 1 & 1/2 & 1/2 \\ 1/2 & 1 & 1/2 \\ 1/2 & 1/2 & 1 \end{bmatrix}$$

Initializing the recursion we have

$$\mathbf{a}_0 = 1 \quad ; \quad \epsilon_0 = r_x(0) = 1 \quad ; \quad \mathbf{R}_0^{-1} = 1$$

Then, using the Levinson-Durbin recursion to calculate the first-order model

$$\Gamma_1 = -r_x(1)/r_x(0) = -1/2$$

and

$$\mathbf{a}_1 = \begin{bmatrix} 1 \\ -1/2 \end{bmatrix}$$

Computing the first-order error

$$\epsilon_1 = \epsilon_0 \left[1 - \Gamma_1^2\right] = 3/4$$

we find the inverse of $\mathbf{R}_1$ to be

$$\begin{aligned} \mathbf{R}_1^{-1} &= \begin{bmatrix} \mathbf{R}_0^{-1} & 0 \\ 0 & 0 \end{bmatrix} + \frac{1}{\epsilon_1} \mathbf{a}_1^R (\mathbf{a}_1^R)^T \\ &= \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} + \frac{4}{3} \begin{bmatrix} -1/2 \\ 1 \end{bmatrix} \begin{bmatrix} -1/2 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 4/3 & -2/3 \\ -2/3 & 4/3 \end{bmatrix} \end{aligned} \tag{5.111}$$

Using the Levinson-Durbin we next generate the second-order model. With

$$\begin{aligned}\gamma_1 &= r_x(2) + a_1(1)r_x(1) = 1/4\\ \Gamma_2 &= -\gamma_1/\epsilon_1 = -1/3\\ \epsilon_2 &= \epsilon_1\left[1 - \Gamma_2^2\right] = 2/3\end{aligned}$$

we find, for $\mathbf{a}_2$,

$$\mathbf{a}_2 = \begin{bmatrix} 1 \\ -1/2 \\ 0 \end{bmatrix} - \frac{1}{3}\begin{bmatrix} 0 \\ -1/2 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ -1/3 \\ -1/3 \end{bmatrix}$$

Therefore the inverse of $\mathbf{R}_2$ is

$$\begin{aligned}\mathbf{R}_2^{-1} &= \begin{bmatrix} \mathbf{R}_1^{-1} & 0 \\ 0 & 0 \end{bmatrix} + \frac{1}{\epsilon_2}\mathbf{a}_2^R(\mathbf{a}_2^R)^T\\ &= \begin{bmatrix} 4/3 & -2/3 & 0 \\ -2/3 & 4/3 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \frac{3}{2}\begin{bmatrix} -1/3 \\ -1/3 \\ 1 \end{bmatrix}\begin{bmatrix} -1/3 & -1/3 & 1 \end{bmatrix}\\ &= \begin{bmatrix} 3/2 & -1/2 & -1/2 \\ -1/2 & 3/2 & -1/2 \\ -1/2 & -1/2 & 3/2 \end{bmatrix}\end{aligned} \tag{5.112}$$

which is the desired result.

A slight variation of the recursion given in Eq. (5.110) may be derived by exploiting the Toeplitz structure of $\mathbf{R}_p$. Specifically, since

$$\mathbf{J}\mathbf{R}_p\mathbf{J} = \mathbf{R}_p^*$$

where $\mathbf{J}$ is the exchange matrix, taking the inverse of both sides of this equation, using the fact that $\mathbf{J}^{-1} = \mathbf{J}$, we find

$$\mathbf{R}_p^{-1} = \mathbf{J}(\mathbf{R}_p^{-1})^*\mathbf{J}$$

Applying this transformation to the expression for $\mathbf{R}_{j+1}^{-1}$ given in Eq. (5.110) we have

$$\mathbf{R}_{j+1}^{-1} = \mathbf{J}(\mathbf{R}_{j+1}^{-1})^*\mathbf{J} = \mathbf{J}\left[\begin{array}{c|c} (\mathbf{R}_j^{-1})^* & \mathbf{0} \\ \hline \mathbf{0}^T & 0 \end{array}\right]\mathbf{J} + \frac{1}{\epsilon_{j+1}}\mathbf{J}(\mathbf{a}_{j+1}^R)^*(\mathbf{a}_{j+1}^R)^T\mathbf{J} \tag{5.113}$$

Simplifying the expression on the right by multiplying by the exchange matrices, we find

$$\boxed{\mathbf{R}_{j+1}^{-1} = \left[\begin{array}{c|c} 0 & \mathbf{0}^T \\ \hline \mathbf{0} & \mathbf{R}_j^{-1} \end{array}\right] + \frac{1}{\epsilon_{j+1}} \mathbf{a}_{j+1} \mathbf{a}_{j+1}^H} \tag{5.114}$$

which is the desired recursion.

In the next section, we derive the Levinson recursion which may be used to solve a general set of Toeplitz equations of the form

$$\mathbf{R}_p \mathbf{x}_p = \mathbf{b}_p \tag{5.115}$$

where $\mathbf{b}_p = [b(0), b(1), \ldots, b(p)]^T$ is an arbitrary vector. Here, we will derive an equivalent form of the recursion that is based on the recursion given in Eq. (5.110). The derivation proceeds as follows. Suppose that we are given the solution to the Toeplitz equations $\mathbf{R}_j \mathbf{x}_j = \mathbf{b}_j$ and that we want to derive the solution to

$$\mathbf{R}_{j+1} \mathbf{x}_{j+1} = \mathbf{b}_{j+1} \tag{5.116}$$

where

$$\mathbf{b}_{j+1} = \left[\begin{array}{c} \mathbf{b}_j \\ \hline b_{j+1} \end{array}\right]$$

Applying Eq. (5.110) to Eq. (5.116) we have

$$\begin{aligned} \mathbf{x}_{j+1} &= \left[\begin{array}{c|c} \mathbf{R}_j^{-1} & \mathbf{0} \\ \hline \mathbf{0}^T & 0 \end{array}\right] \mathbf{b}_{j+1} + \frac{1}{\epsilon_{j+1}} \mathbf{a}_{j+1}^R \left(\mathbf{a}_{j+1}^R\right)^H \mathbf{b}_{j+1} \\ &= \left[\begin{array}{c} \mathbf{x}_j \\ \hline 0 \end{array}\right] + \frac{1}{\epsilon_{j+1}} \left[\left(\mathbf{a}_{j+1}^R\right)^H \mathbf{b}_{j+1}\right] \mathbf{a}_{j+1}^R \end{aligned} \tag{5.117}$$

or

$$\boxed{\mathbf{x}_{j+1} = \left[\begin{array}{c} \mathbf{x}_j \\ \hline 0 \end{array}\right] + \frac{\alpha_{j+1}}{\epsilon_{j+1}} \mathbf{a}_{j+1}^R} \tag{5.118}$$

where

$$\boxed{\alpha_{j+1} = \left(\mathbf{a}_{j+1}^R\right)^H \mathbf{b}_{j+1} = \sum_{k=0}^{j+1} b_{j+1}(k) a_{j+1}(j+1-k)} \tag{5.119}$$

Equations (5.118) and (5.119) along with the Levinson-Durbin recursion for performing the updates of $\mathbf{a}_j$ constitute the Levinson recursion.

5.3 THE LEVINSON RECURSION

We have seen how the Toeplitz structure of the all-pole normal equations is exploited in the Levinson-Durbin recursion to efficiently solve a set of equations of the form

$$\mathbf{R}_p\mathbf{a}_p = \epsilon_p\mathbf{u}_1 \tag{5.120}$$

where $\mathbf{R}_p$ is a $(p+1) \times (p+1)$ Hermitian Toeplitz matrix and $\mathbf{u}_1$ is a unit vector. There are many applications, however, that require solving a more general set of Toeplitz equations

$$\mathbf{R}_p\mathbf{x}_p = \mathbf{b} \tag{5.121}$$

where the vector on the right-hand side is arbitrary. For example, as we saw in Chapter 4, Shanks' method requires solving a set of linear equations of this form. In addition, we will see in Chapter 7 that the design of an FIR digital Wiener filter requires finding the solution to the Wiener-Hopf equation

$$\mathbf{R}_p\mathbf{a}_p = \mathbf{r}_{dx}$$

where $\mathbf{R}_p$ is a Toeplitz matrix and $\mathbf{r}_{dx}$ is a vector of cross-correlations that is, in general, unrelated to the elements in $\mathbf{R}_p$. Therefore, in this section, we derive the original recursion developed by N. Levinson in 1947 for solving a general set of Hermitian Toeplitz equations of the form given in Eq. (5.121) where $\mathbf{b}$ is an arbitrary vector. This recursion, known as the Levinson recursion, simultaneously solves the following two sets of linear equations

$$\begin{bmatrix} r_x(0) & r_x^*(1) & r_x^*(2) & \cdots & r_x^*(j) \\ r_x(1) & r_x(0) & r_x^*(1) & \cdots & r_x^*(j-1) \\ r_x(2) & r_x(1) & r_x(0) & \cdots & r_x^*(j-2) \\ \vdots & \vdots & \vdots & & \vdots \\ r_x(j) & r_x(j-1) & r_x(j-2) & \cdots & r_x(0) \end{bmatrix} \begin{bmatrix} 1 \\ a_j(1) \\ a_j(2) \\ \vdots \\ a_j(j) \end{bmatrix} = \begin{bmatrix} \epsilon_j \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{5.122}$$

and

$$\begin{bmatrix} r_x(0) & r_x^*(1) & r_x^*(2) & \cdots & r_x^*(j) \\ r_x(1) & r_x(0) & r_x^*(1) & \cdots & r_x^*(j-1) \\ r_x(2) & r_x(1) & r_x(0) & \cdots & r_x^*(j-2) \\ \vdots & \vdots & \vdots & & \vdots \\ r_x(j) & r_x(j-1) & r_x(j-2) & \cdots & r_x(0) \end{bmatrix} \begin{bmatrix} x_j(0) \\ x_j(1) \\ x_j(2) \\ \vdots \\ x_j(j) \end{bmatrix} = \begin{bmatrix} b(0) \\ b(1) \\ b(2) \\ \vdots \\ b(j) \end{bmatrix} \tag{5.123}$$

for $j = 0, 1, \ldots, p$. Since Eq. (5.122) corresponds to the all-pole normal equations, we will find that, embedded within the Levinson recursion, is the Levinson-Durbin recursion. In Section 5.2.9 we used the Toeplitz matrix inverse recursion to derive one form of the Levinson recursion. Here, we will rederive the recursion without using this matrix inverse recursion. The derivation, in fact, will closely resemble that used for the Levinson-Durbin recursion in Section 5.2.

The Levinson recursion begins by finding the solution to Eqs. (5.122) and (5.123) for $j = 0$. The result is clearly seen to be

$$\epsilon_0 = r_x(0) \quad \text{and} \quad x_0(0) = b(0)/r_x(0)$$

Now let us assume that we know the solution to the jth-order equations and derive the solution to the $(j+1)$st-order equations. Given the solution $\mathbf{a}_j$ to Eq. (5.122) the Levinson-

Durbin update equation (5.14) gives us the solution $\mathbf{a}_{j+1}$ to the $(j+1)$st-order equations

$$\begin{bmatrix} r_x(0) & r_x^*(1) & \cdots & r_x^*(j) & r_x^*(j+1) \\ r_x(1) & r_x(0) & \cdots & r_x^*(j-1) & r_x^*(j) \\ \vdots & \vdots & & \vdots & \vdots \\ r_x(j) & r_x(j-1) & \cdots & r_x(0) & r_x^*(1) \\ r_x(j+1) & r_x(j) & \cdots & r_x(1) & r_x(0) \end{bmatrix} \begin{bmatrix} 1 \\ a_{j+1}(1) \\ \vdots \\ a_{j+1}(j) \\ a_{j+1}(j+1) \end{bmatrix} = \begin{bmatrix} \epsilon_{j+1} \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix} \tag{5.124}$$

Exploiting the Hermitian Toeplitz property of $\mathbf{R}_{j+1}$, if we take the complex conjugate of Eq. (5.124) and reverse the order of the rows and columns of $\mathbf{R}_{j+1}$, Eq. (5.124) may be written in the equivalent form

$$\begin{bmatrix} r_x(0) & r_x^*(1) & \cdots & r_x^*(j) & r_x^*(j+1) \\ r_x(1) & r_x(0) & \cdots & r_x^*(j-1) & r_x^*(j) \\ \vdots & \vdots & & \vdots & \vdots \\ r_x(j) & r_x(j-1) & \cdots & r_x(0) & r_x^*(1) \\ r_x(j+1) & r_x(j) & \cdots & r_x(1) & r_x(0) \end{bmatrix} \begin{bmatrix} a_{j+1}^*(j+1) \\ a_{j+1}^*(j) \\ \vdots \\ a_{j+1}^*(1) \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ \epsilon_{j+1}^* \end{bmatrix} \tag{5.125}$$

Since it is assumed that the solution $\mathbf{x}_j$ to Eq. (5.123) is known, suppose that we append a zero to $\mathbf{x}_j$ and then multiply this extended vector by $\mathbf{R}_{j+1}$, as we did in the Levinson-Durbin recursion. This leads to the following set of linear equations

$$\begin{bmatrix} r_x(0) & r_x^*(1) & \cdots & r_x^*(j) & r_x^*(j+1) \\ r_x(1) & r_x(0) & \cdots & r_x^*(j-1) & r_x^*(j) \\ \vdots & \vdots & & \vdots & \vdots \\ r_x(j) & r_x(j-1) & \cdots & r_x(0) & r_x^*(1) \\ r_x(j+1) & r_x(j) & \cdots & r_x(1) & r_x(0) \end{bmatrix} \begin{bmatrix} x_j(0) \\ x_j(1) \\ \vdots \\ x_j(j) \\ 0 \end{bmatrix} = \begin{bmatrix} b(0) \\ b(1) \\ \vdots \\ b(j) \\ \delta_j \end{bmatrix} \tag{5.126}$$

where δ_j is given by

$$\delta_j = \sum_{i=0}^{j} r_x(j+1-i)x_j(i)$$

Since δ_j is not, in general, equal to $b(j+1)$, the extended vector $[\mathbf{x}_j^T, 0]$ is not the solution to the $(j+1)$st-order equations. Consider what happens, however, if we form the sum

$$\mathbf{R}_{j+1}\left\{\begin{bmatrix} x_j(0) \\ x_j(1) \\ \vdots \\ x_j(j) \\ 0 \end{bmatrix} + q_{j+1}\begin{bmatrix} a_{j+1}^*(j+1) \\ a_{j+1}^*(j) \\ \vdots \\ a_{j+1}^*(1) \\ 1 \end{bmatrix}\right\} = \begin{bmatrix} b(0) \\ b(1) \\ \vdots \\ b(j) \\ \delta_j + q_{j+1}\epsilon_{j+1}^* \end{bmatrix} \tag{5.127}$$

where q_{j+1} is an arbitrary complex constant. If we set

$$\boxed{q_{j+1} = \frac{b(j+1) - \delta_j}{\epsilon^*_{j+1}}} \tag{5.128}$$

then

$$\delta_j + q_{j+1}\epsilon^*_{j+1} = b(j+1)$$

and the vector

$$\mathbf{x}_{j+1} = \begin{bmatrix} x_j(0) \\ x_j(1) \\ \vdots \\ x_j(j) \\ 0 \end{bmatrix} + q_{j+1} \begin{bmatrix} a^*_{j+1}(j+1) \\ a^*_{j+1}(j) \\ \vdots \\ a^*_{j+1}(1) \\ 1 \end{bmatrix} \tag{5.129}$$

is the desired solution. Note that since ϵ_j is real, the complex conjugate in Eq. (5.128) may be neglected. Thus, the Levinson recursion, as summarized in Table 5.6, is defined by Eqs. (5.128) and (5.129) along with the Levinson-Durbin recursion.

Example 5.3.1 *The Levinson Recursion*

Let us use the Levinson recursion to solve the following set of Toeplitz equations,

$$\begin{bmatrix} 4 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 4 \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \end{bmatrix} = \begin{bmatrix} 9 \\ 6 \\ 12 \end{bmatrix}$$

The solution is as follows.

1. Initialization of the recursion.

$$\begin{aligned} \epsilon_0 &= r(0) = 4 \\ x_0(0) &= b(0)/r(0) = 9/4 \end{aligned}$$

2. For $j = 0$,

$$\begin{aligned} \gamma_0 &= r(1) = 2 \\ \Gamma_1 &= -\gamma_0/\epsilon_0 = -1/2 \end{aligned}$$

Therefore we have, for the first-order model,

$$\mathbf{a}_1 = \begin{bmatrix} 1 \\ \Gamma_1 \end{bmatrix} = \begin{bmatrix} 1 \\ -1/2 \end{bmatrix}$$

Continuing we have,

$$\begin{aligned} \epsilon_1 &= \epsilon_0\left[1 - |\Gamma_1|^2\right] = 4(1 - 1/4) = 3 \\ \delta_0 &= x_0(0)r(1) = 2(9/4) = 9/2 \\ q_1 &= \left[b(1) - \delta_0\right]/\epsilon_1 = (6 - 9/2)/3 = 1/2 \end{aligned}$$

Table 5.6 *The Levinson Recursion*

1. Initialize the recursion
   - (a) $a_0(0) = 1$
   - (b) $x_0(0) = b(0)/r_x(0)$
   - (c) $\epsilon_0 = r_x(0)$
2. For $j = 0, 1, \ldots, p-1$
   - (a) $\gamma_j = r_x(j+1) + \sum_{i=1}^{j} a_j(i) r_x(j-i+1)$
   - (b) $\Gamma_{j+1} = -\gamma_j/\epsilon_j$
   - (c) For $i = 1, 2, \ldots, j$
     $$a_{j+1}(i) = a_j(i) + \Gamma_{j+1} a_j^*(j-i+1)$$
   - (d) $a_{j+1}(j+1) = \Gamma_{j+1}$
   - (e) $\epsilon_{j+1} = \epsilon_j[1 - |\Gamma_{j+1}|^2]$
   - (f) $\delta_j = \sum_{i=0}^{j} x_j(i) r_x(j-i+1)$
   - (g) $q_{j+1} = [b(j+1) - \delta_j]/\epsilon_{j+1}$
   - (h) For $i = 0, 1, \ldots, j$
     $$x_{j+1}(i) = x_j(i) + q_{j+1} a_{j+1}^*(j-i+1)$$
   - (i) $x_{j+1}(j+1) = q_{j+1}$

which leads to the first-order solution

$$\mathbf{x}_1 = \begin{bmatrix} x_0(0) \\ 0 \end{bmatrix} + q_1 \begin{bmatrix} a_1(1) \\ 1 \end{bmatrix} = \begin{bmatrix} 9/4 \\ 0 \end{bmatrix} + 1/2 \begin{bmatrix} -1/2 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 1/2 \end{bmatrix}$$

3. For $j = 1$ we find that

$$\gamma_1 = r(2) + a_1(1) r(1) = 1 + 2(-1/2) = 0$$

Therefore, $\Gamma_2 = 0$ and the second-order model is simply

$$\mathbf{a}_2 = \begin{bmatrix} \mathbf{a}_1 \\ 0 \end{bmatrix}$$

In addition, $\epsilon_2 = \epsilon_1$ and

$$\delta_1 = \begin{bmatrix} x_1(0), & x_1(1) \end{bmatrix} \begin{bmatrix} r(2) \\ r(1) \end{bmatrix} = 2 + 2(1/2) = 3$$

$$q_2 = \begin{bmatrix} b(2) - \delta_1 \end{bmatrix} / \epsilon_2 = (12 - 3)/3 = 3$$

Thus, the desired solution is

$$\mathbf{x}_2 = \begin{bmatrix} x_1(0) \\ x_1(1) \\ 0 \end{bmatrix} + q_2 \begin{bmatrix} a_2(2) \\ a_2(1) \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 1/2 \\ 0 \end{bmatrix} + 3 \begin{bmatrix} 0 \\ -1/2 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ -1 \\ 3 \end{bmatrix}$$

The General Levinson Recursion

```
function x=glev(r,b)
%
   r=r(:);
   p=length(b);
   a=1;
   x=b(1)/r(1);
   epsilon=r(1);
   for j=2:p;
       g=r(2:j)'*flipud(a);
       gamma=-g/epsilon;
       a=[a;0] + gamma*[0;conj(flipud(a))];
       epsilon=epsilon*(1 - abs(gamma)^2);
       delta=r(2:j)'*flipud(x);
       q=(b(j)-delta)/epsilon;
       x=[x;0] + q*[conj(flipud(a))];
       end
```

Figure 5.16 *A* MATLAB *program for solving a set of Toeplitz equations of the form* $\mathbf{R}_p\mathbf{x}_p = \mathbf{b}$ *using the general Levinson recursion.*

We conclude this section by looking at the computational complexity of the Levinson recursion. From Table 5.6 we see that at jth step of the recursion there are two divisions, $4(j+1)$ multiplications, and $4j+3$ additions. Since there are p steps in the recursion, the total number of multiplications and divisions that are required is

$$\sum_{j=0}^{p-1}(4j+6) = 2p^2 + 4p$$

and the number of additions is

$$\sum_{j=0}^{p-1}(4j+3) = 2p^2 + p$$

Therefore, the number of multiplications and divisions is proportional to p^2 for the Levinson recursion compared with p^3 for Gaussian elimination, and twice that required for the Levinson-Durbin recursion. A MATLAB program for the general Levinson recursion is provided in Fig. 5.16.

5.4 THE SPLIT LEVINSON RECURSION*

The Levinson-Durbin recursion provides an efficient algorithm for solving the pth-order autocorrelation normal equations given in Eq. (5.5) using $p^2 + 2p$ multiplications and p^2 additions. It is possible, however, to reduce the number of multiplications by approximately a factor of 2 while maintaining approximately the same number of additions using what is known as the *Split Levinson Recursion* [5]. The key to achieving this reduction is to replace the predictor polynomials $\mathbf{a}_j$ with *symmetric* vectors, $\mathbf{s}_j$, and use these instead of $\mathbf{a}_j$ in a "Levinson-like" recursion. Due to the symmetry of $\mathbf{s}_j$, only half of the coefficients need to

be computed. What needs to be done to make this procedure work is to develop a recursion for the symmetric vectors along with a procedure to find $\mathbf{a}_p$ from the symmetric vectors. The split Levinson algorithm was originally developed by Delsarte and Genin for the case of real data [5]. Later they extended this work by developing a split Schur recursion along with a split lattice filter structure [6]. Although split algorithms have also been developed for the complex case [1, 2, 12], these algorithms are a bit more complicated. Therefore, in this section we will restrict our attention to the case of real data and follow the development originally presented in [5]. In the following chapter, we derive the split lattice filter structure.

From our development of the Levinson-Durbin recursion in Section 5.2.1, we have seen that the solution to the pth-order normal equations is built up recursively from the sequence of reflection coefficients Γ_j for $j = 1, \ldots, p$ and the prediction error filters $A_j(z)$ as follows

$$A_j(z) = A_{j-1}(z) + \Gamma_j z^{-1} A_{j-1}^R(z) \quad ; \quad A_0(z) = 1 \tag{5.130}$$

where $A_j^R(z) = z^{-j} A_j(z^{-1})$ is the *reciprocal polynomial.*[8] If we replace z with z^{-1} in Eq. (5.130) and multiply both sides by z^{-j} we find, as we did in Section 5.2.1, that $A_j^R(z)$ satisfies a similar recursion given by

$$A_j^R(z) = z^{-1} A_{j-1}^R(z) + \Gamma_j A_{j-1}(z) \quad ; \quad A_0^R(z) = 1 \tag{5.131}$$

Let us now define a second set of polynomials, $S_j(z)$ and $S_j^\star(z)$, that are formed by setting the reflection coefficient in Eq. (5.130) equal to $\Gamma_j = 1$ and $\Gamma_j = -1$, respectively,

$$S_j(z) = A_{j-1}(z) + z^{-1} A_{j-1}^R(z) \tag{5.132}$$

$$S_j^\star(z) = A_{j-1}(z) - z^{-1} A_{j-1}^R(z) \tag{5.133}$$

These polynomials are referred to as the *singular predictor polynomials.*[9] By adding together Eqs. (5.130) and (5.131) we may relate $S_j(z)$ to $A_j(z)$ and $A_j^R(z)$ as follows,

$$(1 + \Gamma_j) S_j(z) = A_j(z) + A_j^R(z) \tag{5.134}$$

Similarly, if we subtract Eq. (5.131) from Eq. (5.130) we find that the singular polynomial $S_j^\star(z)$ is related to $A_j(z)$ and $A_j^R(z)$ by

$$(1 - \Gamma_j) S_j^\star(z) = A_j(z) - A_j^R(z) \tag{5.135}$$

If we denote the coefficients of the singular predictor polynomials $S_j(z)$ and $S_j^\star(z)$ by $s_j(i)$ and $s_j^\star(i)$, respectively,

$$S_j(z) = \sum_{i=0}^{j} s_j(i) z^{-i}$$

and

$$S_j^\star(z) = \sum_{i=0}^{j} s_j^\star(i) z^{-i}$$

[8]Recall that we are assuming that the coefficients $a_j(k)$ are real. Therefore, $A_j(z)$ is conjugate symmetric, $A_j(z) = A_j^*(z^*)$ and, as a result, the reciprocal polynomial has the form given here [compare this to the definition of $A_j^R(z)$ given in Eq. (5.22)].

[9]It is interesting to note that the singular predictor polynomials are identical to the *line spectral pairs* that are of interest in speech processing [4,23].

and let $\mathbf{s}_j = [s_j(0), \ldots, s_j(j)]^T$ and $\mathbf{s}_j^\star = [s_j^\star(0), \ldots, s_j^\star(j)]^T$ be the singular predictor vectors, then it follows from Eqs. (5.134) and (5.135) that

$$(1 + \Gamma_j)\mathbf{s}_j = \begin{bmatrix} 1 \\ a_j(1) \\ \vdots \\ a_j(j-1) \\ a_j(j) \end{bmatrix} + \begin{bmatrix} a_j(j) \\ a_j(j-1) \\ \vdots \\ a_j(1) \\ 1 \end{bmatrix} \tag{5.136}$$

and

$$(1 - \Gamma_j)\mathbf{s}_j^\star = \begin{bmatrix} 1 \\ a_j(1) \\ \vdots \\ a_j(j-1) \\ a_j(j) \end{bmatrix} - \begin{bmatrix} a_j(j) \\ a_j(j-1) \\ \vdots \\ a_j(1) \\ 1 \end{bmatrix} \tag{5.137}$$

Therefore, $\mathbf{s}_j$ is a *symmetric* vector that is formed by taking the symmetric (even) part of the vector $\mathbf{a}_j$, and $\mathbf{s}_j^\star$ is an *antisymmetric* vector corresponding to the antisymmetric (odd) part of $\mathbf{a}_j$.[10] Thus, $S_j(z)$ is a *symmetric* polynomial

$$S_j(z) = z^{-j} S_j(z^{-1})$$

and $S_j^\star(z)$ is an *antisymmetric* polynomial,

$$S_j^\star(z) = -z^{-j} S_j^\star(z^{-1})$$

and as a result, both have linear phase.[11]

We now want to show that $\mathbf{s}_j$ and $\mathbf{s}_j^\star$ satisfy a set of Toeplitz equations. Clearly from Eq. (5.136) it follows that

$$(1 + \Gamma_j)\mathbf{R}_j\mathbf{s}_j = \mathbf{R}_j\mathbf{a}_j + \mathbf{R}_j\mathbf{a}_j^R$$

Therefore, since

$$\mathbf{R}_j\mathbf{a}_j = \epsilon_j[1, 0, \ldots, 0]^T$$

and

$$\mathbf{R}_j\mathbf{a}_j^R = \epsilon_j[0, \ldots, 0, 1]^T$$

then

$$\boxed{\mathbf{R}_j\mathbf{s}_j = \tau_j\left[1, 0, \ldots, 0, 1\right]^T} \tag{5.138}$$

[10] It is for this reason that the algorithm we are developing is called a "split" Levinson recursion, i.e., it *splits* the predictor polynomial $\mathbf{a}_j$ into its *symmetric* and *antisymmetric* parts.

[11] We may also see this from the definitions of $S_j(z)$ and $S_j^\star(z)$ given in Eqs. (5.132) and (5.133).

where we have defined

$$\tau_j = \frac{\epsilon_j}{1+\Gamma_j} \tag{5.139}$$

Note that τ_j may be evaluated from $r_x(i)$ and the coefficients of the singular predictor vector $\mathbf{s}_j$ as follows

$$\boxed{\tau_j = \sum_{i=0}^{j} r_x(i)s_j(i)} \tag{5.140}$$

Similarly for $\mathbf{s}_j^\star$ we find that $\mathbf{s}_j^\star$ satisfies the Toeplitz equations

$$\boxed{\mathbf{R}_j\mathbf{s}_j^\star = \tau_j^\star\left[1, 0, \ldots, 0, -1\right]^T} \tag{5.141}$$

where

$$\tau_j^\star = \frac{\epsilon_j}{1-\Gamma_j} \tag{5.142}$$

Given that $\mathbf{s}_j$ and $\mathbf{s}_j^\star$ satisfy a set of Toeplitz equations, we may now derive a recursion for the singular predictor polynomials $S_j(z)$, similar to what we did for $A_j(z)$ and $A_j^R(z)$ in the Levinson-Durbin recursion. To do this, we must first show how to express $A_j(z)$ in terms of $S_j(z)$ and $S_{j+1}(z)$. Beginning with Eq. (5.134) we have

$$A_j(z) = (1+\Gamma_j)S_j(z) - A_j^R(z) \tag{5.143}$$

Replacing j by $(j+1)$ in Eq. (5.132) and solving for $A_j^R(z)$ gives

$$A_j^R(z) = zS_{j+1}(z) - zA_j(z) \tag{5.144}$$

Substituting Eq. (5.144) into Eq. (5.143) we find

$$A_j(z) = (1+\Gamma_j)S_j(z) - zS_{j+1}(z) + zA_j(z)$$

Therefore, solving for $A_j(z)$,

$$\boxed{(1-z)A_j(z) = (1+\Gamma_j)S_j(z) - zS_{j+1}(z)} \tag{5.145}$$

which is the desired relation. Using Eq. (5.145) we may now derive a recursion for $S_j(z)$. We begin with the Levinson-Durbin recursion

$$A_j(z) = A_{j-1}(z) + \Gamma_j z^{-1}A_{j-1}^R(z) \tag{5.146}$$

and note from Eq. (5.132) that $A_{j-1}^R(z)$ may be expressed in terms of $A_{j-1}(z)$ and $S_j(z)$ as follows,

$$A_{j-1}^R(z) = zS_j(z) - zA_{j-1}(z)$$

Substituting this expression for $A_{j-1}^R(z)$ into Eq. (5.146) we have

$$\begin{aligned} A_j(z) &= A_{j-1}(z) + \Gamma_j z^{-1}\left[zS_j(z) - zA_{j-1}(z)\right] \\ &= (1-\Gamma_j)A_{j-1}(z) + \Gamma_j S_j(z) \end{aligned} \tag{5.147}$$

Now, as we saw in Eq. (5.145), $A_j(z)$ may be written in terms of $S_j(z)$ and $S_{j+1}(z)$. Therefore, multiplying both sides of Eq. (5.147) by $(1 - z)$

$$(1 - z)A_j(z) = (1 - \Gamma_j)(1 - z)A_{j-1}(z) + \Gamma_j(1 - z)S_j(z)$$

and using Eq. (5.145) to eliminate $(1-z)A_j(z)$ and $(1-z)A_{j-1}(z)$ we have, after combining together common terms,

$$S_{j+1}(z) = (1 + z^{-1})S_j(z) - \delta_j z^{-1} S_{j-1}(z) \tag{5.148}$$

where we have defined

$$\delta_j = (1 - \Gamma_j)(1 + \Gamma_{j-1}) \tag{5.149}$$

In the time-domain, Eq. (5.148) becomes

$$\boxed{s_{j+1}(i) = s_j(i) + s_j(i - 1) - \delta_j s_{j-1}(i - 1)} \tag{5.150}$$

which is referred to as the *three-term recurrence.*[12] Expressed in terms of the singular predictor vectors, this three-term recurrence is

$$\begin{bmatrix} s_{j+1}(0) \\ s_{j+1}(1) \\ \vdots \\ s_{j+1}(j) \\ s_{j+1}(j+1) \end{bmatrix} = \begin{bmatrix} s_j(0) \\ s_j(1) \\ \vdots \\ s_j(j) \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ s_j(0) \\ \vdots \\ s_j(j-1) \\ s_j(j) \end{bmatrix} - \delta_j \begin{bmatrix} 0 \\ s_{j-1}(0) \\ \vdots \\ s_{j-1}(j-1) \\ 0 \end{bmatrix} \tag{5.151}$$

Equation (5.148) or, equivalently, Eq. (5.151), gives us a method for computing the singular predictor polynomial $S_{j+1}(z)$ given the previous two polynomials, $S_j(z)$ and $S_{j-1}(z)$, and the parameter δ_j. What is needed to complete the recursion is a method for computing δ_j. Since the reflection coefficients Γ_j are not known, we cannot use Eq. (5.149) directly. However, using Eq. (5.139) we see that

$$\frac{\tau_j}{\tau_{j-1}} = \frac{\epsilon_j}{1 + \Gamma_j} \frac{1 + \Gamma_{j-1}}{\epsilon_{j-1}}$$

However, since $\epsilon_j = \epsilon_{j-1}(1 - \Gamma_j^2)$, then

$$\frac{\tau_j}{\tau_{j-1}} = \frac{(1 - \Gamma_j^2)(1 + \Gamma_{j-1})}{1 + \Gamma_j} = (1 - \Gamma_j)(1 + \Gamma_{j-1}) \tag{5.152}$$

which, from Eq. (5.149), is equal to δ_j. Therefore, δ_j may be evaluated using

$$\boxed{\delta_j = \frac{\tau_j}{\tau_{j-1}}} \tag{5.153}$$

[12]This relation is called a three-term recurrence since there are three terms that are used to update the singular predictor vector. By contrast, note that the Levinson-Durbin order update equation is a *two-term recurrence.*

along with Eq. (5.140). Equations (5.140), (5.150), and (5.153) constitute the split Levinson recursion for generating the singular predictor polynomials. The initial conditions required to begin the recursion are

$$\begin{aligned} S_0(z) &= \frac{1}{1+\Gamma_0}\left[A_0(z) + A_0^R(z)\right] = 2 \\ S_1(z) &= A_0(z) + z^{-1}A_0^R(z) = 1 + z^{-1} \end{aligned} \tag{5.154}$$

Now that we have a recursion for $S_j(z)$, all that is left is to show how to derive the prediction error filter $A_p(z)$ from the singular predictor polynomials. From Eq. (5.145) with $j = p$ we have

$$(1 - z)A_p(z) = (1 + \Gamma_p)S_p(z) - zS_{p+1}(z) \tag{5.155}$$

Multiplying both sides by $-z^{-1}$ and solving for $A_p(z)$ gives

$$A_p(z) = z^{-1}A_p(z) + S_{p+1}(z) - z^{-1}(1 + \Gamma_p)S_p(z) \tag{5.156}$$

which, in the coefficient domain becomes

$$\boxed{a_p(j) = a_p(j-1) + s_{p+1}(j) - (1 + \Gamma_p)s_p(j-1)} \tag{5.157}$$

With the initial condition $a_p(0) = 1$ we thus have a recursion for the coefficients $a_p(j)$. All that is missing is an expression for $(1 + \Gamma_p)$. However, this may be computed from $\mathbf{s}_{p+1}$ and $\mathbf{s}_p$ as follows. Setting $z = 1$ in Eq. (5.155) we have

$$(1 + \Gamma_p)S_p(z)\Big|_{z=1} - S_{p+1}(z)\Big|_{z=1} = 0 \tag{5.158}$$

Therefore, since

$$S_p(z)\Big|_{z=1} = \sum_{i=0}^{p} s_p(i) \quad \text{and} \quad S_{p+1}(z)\Big|_{z=1} = \sum_{i=0}^{p+1} s_{p+1}(i)$$

then

$$\boxed{(1 + \Gamma_p) = \frac{\displaystyle\sum_{i=0}^{p+1} s_{p+1}(i)}{\displaystyle\sum_{i=0}^{p} s_p(i)}} \tag{5.159}$$

The complete *Split Levinson Recursion* is summarized in Table 5.7. It is presented, however, in a form that does not take full advantage of the symmetries in the polynomials $S_j(z)$. In its most efficient form, for example, the evaluation of the coefficients τ_j would be accomplished as follows

$$\tau_j = \begin{cases} \displaystyle\sum_{i=0}^{(j-1)/2} \left[r_x(i) + r_x(j-i)\right]s_j(i) & ; \quad j \text{ odd} \\ \displaystyle\sum_{i=0}^{(j-1)/2} \left[r_x(i) + r_x(j-i)\right]s_j(i) + r_x(j/2)s_j(j/2) & ; \quad j \text{ even} \end{cases}$$

Table 5.7 *The Split Levinson Recursion*

1. Initialize the recursion
   (a) $\mathbf{s}_0 = 2$ and $\mathbf{s}_1 = [1, 1]^T$
   (b) $\tau_0 = r_x(0)$
2. For $j = 1, 2, \ldots, p$
   (a) $\tau_j = \sum_{i=0}^{j} r_x(i)s_j(i)$
   (b) $\delta_j = \tau_j/\tau_{j-1}$
   (c) $s_{j+1}(0) = s_{j+1}(j+1) = 1$
   (d) For $k = 1, 2, \ldots, j$
   $$s_{j+1}(k) = s_j(k) + s_j(k-1) - \delta_j s_{j-1}(k-1)$$
3. Set $(1 + \Gamma_p) = \sum_{i=0}^{p+1} s_{p+1}(i) / \sum_{i=0}^{p} s_p(i)$
4. For $j = 1, 2, \ldots, p$
   (a) $a_p(j) = a_p(j-1) + s_{p+1}(j) - (1 + \Gamma_p)s_p(j-1)$
5. Done

In addition, due to the symmetry of s_j, only half of the coefficients need to be evaluated, thereby reducing the number of multiplications and additions by approximately a factor of two.

Although not included as a part of the split Levinson recursion in Table 5.7, it is also possible to compute the reflection coefficient recursively from the coefficients δ_j. Specifically, solving Eq. (5.152) for Γ_j we find that

$$\Gamma_j = 1 - \frac{\delta_j}{1 + \Gamma_{j-1}} \tag{5.160}$$

The initial condition for this recursion is $\Gamma_0 = 0$.

Example 5.4.1 ***The Split Levinson Recursion***

Let us use the split Levinson recursion to find the predictor polynomial corresponding to the autocorrelation sequence

$$\mathbf{r} = [4, 1, 1, -2]^T$$

In order to make the discussion easier to follow, we will not employ the symmetry of $\mathbf{s}_j$ in our calculations.

1. Initialization of the recursion.

$$\tau_0 = r(0) = 4$$
$$\mathbf{s}_0 = 2$$
$$\mathbf{s}_1 = [1, 1]^T$$

2. For $j = 1$,

$$\tau_1 = [4,\ 1]\begin{bmatrix} 1 \\ 1 \end{bmatrix} = 5 \quad ; \quad \delta_1 = \tau_1/\tau_0 = 5/4$$

Now, using Eq.(5.151) we have

$$\mathbf{s}_2 = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} - 5/4 \begin{bmatrix} 0 \\ 2 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ -1/2 \\ 1 \end{bmatrix}$$

3. For $j = 2$ we have

$$\tau_2 = [4,\ 1,\ 1]\begin{bmatrix} 1 \\ -1/2 \\ 1 \end{bmatrix} = 9/2 \quad ; \quad \delta_2 = \tau_2/\tau_1 = (9/2)(1/5) = 9/10$$

Again using Eq.(5.151) to find $\mathbf{s}_3$ we have

$$\mathbf{s}_3 = \begin{bmatrix} 1 \\ -1/2 \\ 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ -1/2 \\ 1 \end{bmatrix} - 9/10 \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ -2/5 \\ -2/5 \\ 1 \end{bmatrix}$$

4. Finally, for $j = 3$ we have

$$\tau_3 = [4,\ 1,\ 1,\ -2]\begin{bmatrix} 1 \\ -2/5 \\ -2/5 \\ 1 \end{bmatrix} = 6/5$$

$$\delta_3 = \tau_3/\tau_2 = (6/5)(2/9) = 4/15$$

Again using Eq.(5.151) to find $\mathbf{s}_4$ we have

$$\mathbf{s}_4 = \begin{bmatrix} 1 \\ -2/5 \\ -2/5 \\ 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ -2/5 \\ -2/5 \\ 1 \end{bmatrix} - 4/15 \begin{bmatrix} 0 \\ 1 \\ -1/2 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1/3 \\ -2/3 \\ 1/3 \\ 1 \end{bmatrix}$$

Finally, we compute the predictor polynomial with the help of Eqs. (5.157) and (5.159).

First, using Eq. (5.159) we have

$$(1+\Gamma_3) = \frac{\sum_{i=0}^{4} s_4(i)}{\sum_{i=0}^{3} s_3(i)} = 5/3$$

Then, from Eq. (5.157) we have

$$\mathbf{a}_3 = \begin{bmatrix} 1 \\ a_3(1) \\ a_3(2) \\ a_3(3) \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ a_3(1) \\ a_3(2) \end{bmatrix} + \begin{bmatrix} 1 \\ s_4(1) \\ s_4(2) \\ s_4(3) \end{bmatrix} - (1+\Gamma_3) \begin{bmatrix} 0 \\ 1 \\ s_3(1) \\ s_3(2) \end{bmatrix}$$

Incorporating the values for the singular predictor vectors gives

$$\begin{bmatrix} 1 \\ a_3(1) \\ a_3(2) \\ a_3(3) \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ a_3(1) \\ a_3(2) \end{bmatrix} + \begin{bmatrix} 1 \\ -4/3 \\ 0 \\ 1 \end{bmatrix}$$

Solving this equation recursively for the coefficients $a_3(i)$ we find

$$\mathbf{a}_3 = \begin{bmatrix} 1 \\ -1/3 \\ -1/3 \\ 2/3 \end{bmatrix}$$

5.5 SUMMARY

In this chapter, we have looked at several different algorithms for recursively solving a set of Toeplitz equations. Each of these algorithms exploits the symmetries and redundancies of the Toeplitz matrix. This chapter began with a derivation of the Levinson-Durbin recursion for solving a set of Hermitian Toeplitz equations in which the right-hand side is a unit vector,

$$\mathbf{R}_p \mathbf{a}_p = \epsilon_p \mathbf{u}_1 \tag{5.161}$$

Compared with $O(p^3)$ operations that are required to solve a set of p linear equations using Gaussian elimination, the Levinson-Durbin recursion requires $O(p^2)$ operations. Using the Levinson-Durbin recursion we were able to establish some important properties of the solution $\mathbf{a}_p$ to these equations. It was shown, for example, that $\mathbf{a}_p$ generates a minimum phase polynomial if and only if the Toeplitz matrix $\mathbf{R}_p$ is positive definite and, in

this case, the reflection coefficients Γ_j that are generated by the Levinson-Durbin recursion are bounded by one in magnitude, $|\Gamma_j| < 1$. As a result, we were able to establish that the all-pole models that are formed using Prony's method and the autocorrelation method are guaranteed to be stable. We were also able to demonstrate that if $\mathbf{R}_p > 0$, then $r_x(k)$ for $k = 0, 1, \ldots, p$ represents a valid partial autocorrelation sequence and may be extended for $k > p$. This extrapolation may be performed by setting $|\Gamma_k| < 1$ for $k > p$ and finding the corresponding autocorrelation sequence using the inverse Levinson-Durbin recursion.

In addition to the Levinson-Durbin recursion, we derived a number of other "Levinson-like" recursions. For example, with the view that the Levinson-Durbin recursion is a mapping from a set of autocorrelations $r_x(k)$ to a set of filter coefficients $a_p(k)$ and a set of reflection coefficients Γ_j, we saw that it was possible to derive recursions to find the filter coefficients from the reflection coefficients (step-up recursion), the reflection coefficients from the filter coefficients (step-down recursion), and the autocorrelation sequence from the reflection coefficients and the modeling error ϵ_p (inverse Levinson-Durbin recursion). Two additional recursions that were developed for solving equations of the form given in Eq. (5.161) are the split Levinson recursion and the Schur recursion. The split Levinson recursion reduces the computational requirements of the Levinson-Durbin recursion by approximately a factor of two by introducing the set of singular vectors $\mathbf{s}_j$. The Schur recursion, on the other hand, achieves some modest computational savings by generating only the reflection coefficients from the autocorrelation sequence. The main advantage of the Schur recursion over the Levinson-Durbin and split Levinson recursions, however, is that it is suitable for parallel processing in a multiprocessor environment. Finally, we derived the general Levinson recursion for solving a general set of Toeplitz equations of the form

$$\mathbf{R}_p \mathbf{a}_p = \mathbf{b} \tag{5.162}$$

where $\mathbf{b}$ is an arbitrary vector. As with the Levinson-Durbin recursion, the general Levinson recursion requires $O(p^2)$ operations.

In addition to the recursions developed in this chapter, there are other *fast* algorithms that may be of interest. With respect to the problem of signal modeling, one particular algorithm of interest is the *fast covariance algorithm* for solving the covariance normal equations given in Eq. (4.127). As with the Levinson and the Levinson-Durbin recursions, the fast covariance algorithm requires on the order of p^2 operations. Although the approach that is used to derive the fast covariance algorithm is similar in style to that used for the Levinson recursion, it is considerably more involved. For example, the fast covariance algorithm requires a time-update recursion in addition to the *order-update recursion* that is found in the Levinson recursion. This time-update recursion relates the solution to the covariance normal equations over the interval $[L, U]$ to the solution that is obtained over the intervals $[L+1, U]$ and $[L, U-1]$. The details of this recursion may be found in [18, 19]. Another recursion of interest is the Trench algorithm, which generalizes the Levinson recursion by allowing the Toeplitz matrix to be non-symmetric [10, 25]. The Trench algorithm therefore may be used to solve the Padé equations and the modified Yule-Walker equations. In its most general form, the Trench algorithm gives both the inverse matrix along with the solution $\mathbf{x}_p$ to Eq. (5.162). Finally, it should be noted that, in addition to the "Levinson-like" recursions, there are other *fast algorithms* for solving linear equations that have other types of structured matrices such as Hankel and Vandermond matrices [8, 15].

References

1. Y. Bistritz, H. Lev-Ari, and T. Kailath, "Complexity reduced lattice filters for digital speech processing," *Proc. 1987 Int. Conf. on Acoust., Speech, Sig. Proc.*, pp. 21–24, April 1987.
2. Y. Bistritz, H. Lev-Ari, and T. Kailath, "Immittance-type three-term Schur and Levinson recursions for quasi-Toeplitz and complex Hermitian matrices," *SIAM Journal on Matrix Analysis and Applications*, vol. 12, no. 3, pp. 497–520, July 1991.
3. R. V. Churchill and J. W. Brown, *Complex Variables and Applications*, McGraw-Hill, New York, 1990.
4. J. R. Deller, J. G. Proakis, and J. H. L. Hansen, *Discrete-time Processing of Speech Signals*, MacMillan, New York, 1993.
5. P. Delsarte and Y. V. Genin, "The split Levinson algorithm," *IEEE Trans. Acoust., Speech, Sig. Proc.*, vol. ASSP-34, no. 3, pp. 470–478, June 1986.
6. P. Delsarte and Y. V. Genin, "On the splitting of classical algorithms in linear prediction theory," *IEEE Trans. Acoust., Speech, Sig. Proc.*, vol. ASSP-35, no. 5, pp. 645–653, May 1987.
7. J. Durbin, "The fitting of time series models," *Rev. Internat. Statist. Inst.*, vol. 23, pp. 233–244, 1960.
8. I. Gohberg, T. Kailath, and I. Koltracht, "Efficient solution of linear systems of equations with recursive structure," *Lin. Alg. Appl.*, vol. 80, no. 81, 1986.
9. I. Gohberg, ed., *I. Schur Methods in Operator Theory and Signal Processing*, vol. 18, Birkhäuser, Boston, 1986.
10. G. H. Golub and C. F. Van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, 1989.
11. M. H. Hayes and M. A. Clements, "An efficient algorithm for computing Pisarenko's harmonic decomposition using Levinson's recursion," *IEEE Trans. Acoust., Speech, Sig. Proc.*, vol. ASSP-34, no. 3, pp. 485–491, June 1986.
12. H. Krishna and S. D. Morgera, "The Levinson recurrence and fast algorithms for solving Toeplitz systems of linear equations," *IEEE Trans. Acoust., Speech, Sig. Proc.*, vol. ASSP-35, no. 6, pp. 839–848, June 1987.
13. S. Y. Kung and Y. H. Hu, "A highly concurrent algorithm and pipelined architecture for solving Toeplitz systems," *IEEE Trans. Acoust., Speech, Sig. Proc.*, vol. ASSP-31, pp. 66–76, Feb. 1983.
14. S. Lang and J. McClellan, "A simple proof of stability for all-pole linear prediction models," *Proc. IEEE*, vol. 67, pp. 860–861, May 1979.
15. H. Lev-Ari and T. Kailath, "Triangular factorization of structured Hermitian matrices," in *Schur Methods in Operator Theory and Signal Processing*, I. Gohberg, Ed., vol. 18, Boston, Birkhäuser, 1986.
16. N. Levinson, "The Wiener rms error criterion in filter design and prediction," *J. Math. Phys.*, vol. 25, pp. 261–278, 1947.
17. J. Makhoul, "On the eigenvalues of symmetric Toeplitz matrices," *IEEE Trans. Acoust., Speech, Sig. Proc.*, vol. ASSP-29, no. 4, pp. 868–872, Aug. 1981.
18. J. H. McClellan, "Parametric Signal Modeling," Chapter 1 in *Advanced Topics in Signal Processing*, Prentice-Hall, NJ, 1988.
19. M. Morf, B. W. Dickenson, T. Kailath, and A. C. G. Vieira, "Efficient solution of covariance equations for linear prediction," *IEEE Trans. Acoust., Speech, Sig. Proc.*, vol. ASSP-25, pp. 429–433, Oct. 1977.
20. L. Pakula and S. Kay, "Simple proofs of the minimum phase property of the prediction error filter," *IEEE Trans. Acoust., Speech, Sig. Proc.*, vol. ASSP-31, no. 2, pp. 501, April 1983.
21. R. A. Roberts and C. T. Mullis, *Digital Signal Processing*, Addison Wesley, Reading, MA. 1987.
22. I. Schur, "On power series which are bounded in the interior of the unit circle," *J. Reine Angew. Math.*, vol. 147, pp. 205–232, 1917, vol. 148, pp. 122–125, 1918. (Translation of Parts I and II may be found in I. Gohberg, pp. 31–59 and 61–88).

23. F. K. Soong and B.-H. Juang, "Line spectrum pair (LSP) and speech compression," *Proc. 1984 Int. Conf. on Acoust., Speech, Sig. Proc.*, pp. 1.10.1–1.10.4, March 1984.
24. S. Treitel and T. J. Ulrych, "A new proof of the minimum-phase property of the unit prediction error operator," *IEEE Trans. Acoust., Speech, Sig. Proc.*, vol. ASSP-27, no. 1, pp. 99–100, February 1979.
25. W. F. Trench, "An algorithm for the inversion of finite Toeplitz matrices," *J. SIAM*, vol. 12, no. 3, pp. 512–522, 1964.
26. R. A. Wiggins and E. A. Robinson, "Recursive solution to the multichannel filtering problem," *J. Geophys. Research*, vol. 70, no. 8, pp. 1885–1891, April 1965.
27. A. Vieira and T. Kailath, "On another approach to the Schur-Cohn criterion," *IEEE Trans. Circ. Syst*, CAS-24, no. 4, pp. 218–220, 1977.

5.6 PROBLEMS

5.1. Given the autocorrelation sequence

$$r_x(0) = 1, \quad r_x(1) = 0.8, \quad r_x(2) = 0.5, \quad r_x(3) = 0.1$$

find the reflection coefficients, Γ_j, the model parameters, $a_j(k)$, and the modeling errors, ϵ_j, for $j = 1, 2, 3$.

5.2. Let $A_{p-1}(z)$ be a polynomial of order $(p-1)$ of the form

$$A_{p-1}(z) = 1 + \sum_{k=1}^{p-1} a_{p-1}(k) z^{-k} \tag{P5.2-1}$$

and let $\Gamma_1, \Gamma_2, \ldots, \Gamma_{p-1}$ be the reflection coefficients that are generated by the Levinson-Durbin recursion. A pth-order polynomial is formed via the Levinson update equation as follows

$$A_p(z) = A_{p-1}(z) + \Gamma_p z^{-p} A_{p-1}^*(1/z^*)$$

(a) If $|\Gamma_j| < 1$ for $j = 1, \ldots, p-1$ and if $|\Gamma_p| = 1$, what can be said about the location of the zeros of $A_p(z)$?

(b) Suppose $A_p(z)$ may be factored as

$$A_p(z) = \prod_{k=1}^{p} (1 - \alpha_k z^{-1})$$

i.e., the zeros of $A_p(z)$ are at $\alpha_1, \alpha_2, \ldots, \alpha_p$. If $\widetilde{A}_p(z)$ is a polynomial with reflection coefficients $\widetilde{\Gamma}_j$ where

$$\widetilde{\Gamma}_j = \Gamma_j \quad \text{for} \quad j = 1, 2, \ldots, p-1$$
$$\widetilde{\Gamma}_p = 1/\Gamma_p^*$$

How are the zeros of $\widetilde{A}_p(z)$ related to those of $A_p(z)$?

(c) If we consider $A_p(z)$ in (P5.2-1) to be a continuous function of the reflection coefficient Γ_p, using your results derived in parts (a) and (b), describe how the zeros move in the z-plane as Γ_p is varied in a continuous fashion from some number, say ϵ, to its reciprocal, $1/\epsilon$.

5.3. Let $a_p(k)$ be the filter coefficients corresponding to the reflection coefficients Γ_k, for $k = 1, 2, \ldots p$.

(a) Prove that if the reflection coefficients are modulated by $(-1)^k$

$$\hat{\Gamma}_k = (-1)^k \Gamma_k$$

then the new set of filter coefficients $\hat{a}_p(k)$ are

$$\hat{a}_p(k) = (-1)^k a_p(k)$$

(b) Can you generalize the result in part (a) to the case in which

$$\hat{\Gamma}_k = \alpha^k \Gamma_k$$

where α is a complex number with $|\alpha| = 1$? What about if $|\alpha| < 1$?

5.4. For the reflection coefficient sequence

$$\Gamma_k = \alpha^k \quad ; \quad k = 1, 2, \ldots$$

with $|\alpha| < 1$, prove that the zeros of the polynomials $A_p(k)$ lie on a circle of radius α for every $p \geq 1$.

5.5. Without factoring any polynomials, determine whether or not a linear shift-invariant filter with system function

$$H(z) = \frac{1 + 0.8z^{-1} - 0.9z^{-2} + 0.3z^{-3}}{1 - 0.9z^{-1} + 0.8z^{-2} - 0.5z^{-3}}$$

is minimum phase, i.e., all the poles and zeros are inside the unit circle.

5.6. Consider the signal

$$x(n) = \delta(n) + b\delta(n - 1)$$

Suppose that we observe $x(n)$ for $n = 0, 1, \ldots, N$.

(a) Using the autocorrelation method, find the 2nd-order all-pole model for $x(n)$.

(b) Suppose we want to find a p-pole model for $x(n)$. Let Γ_j denote the *jth* reflection coefficient of the lattice filter implementation of the all-pole model. Find a recursion for the reflection coefficients that expresses Γ_j in terms of Γ_{j-1} and Γ_{j-2}.

5.7. Suppose we have a data sequence whose z-transform is of the form:

$$X(z) = \frac{G}{1 + \sum_{k=1}^{p} a_p(k) z^{-k}}$$

although the value of p is unknown. The coefficients of the model are computed using Levinson's recursion. How can the value of p be determined by looking at the sequence of reflection coefficients Γ_j for $j = 1, 2, \ldots$?

5.8. Let $r_x(k)$ be a complex autocorrelation sequence given by

$$\mathbf{r}_x = [2, \ 0.5(1 + j), \ 0.5j]^T$$

Use the Levinson-Durbin recursion to solve the autocorrelation normal equations for a second-order all-pole model.

5.9. Determine whether the following statements are *True* or *False*.

(a) If $r_x(k)$ is an autocorrelation sequence with $r_x(k) = 0$ for $|k| > p$ then $\Gamma_k = 0$ for $|k| > p$.

(b) Given an autocorrelation sequence, $r_x(k)$ for $k = 0, \ldots, p$, if the $(p+1) \times (p+1)$ Toeplitz matrix

$$\mathbf{R}_p = \text{Toep}\{r_x(0), r_x(1), \ldots, r_p(p)\}$$

is positive definite, then

$$\mathbf{r}_x = [r_x(0), \ldots, r_x(p), 0, 0, \ldots]^T$$

will *always* be a valid autocorrelation sequence, i.e., extending $r_x(k)$ with zeros is a valid autocorrelation extension.

(c) If $r_x(k)$ is periodic then Γ_j will be periodic with the same period.

5.10. In our discussions of the Levinson-Durbin recursion, we demonstrated the equivalence between the following three sets of parameters:

- $r_x(0), r_x(1), \ldots, r_x(p)$
- $a_p(1), a_p(2), \ldots, a_p(p), b(0)$
- $\Gamma_1, \Gamma_2, \ldots, \Gamma_p, \epsilon_p$

For each of the following signal transformations, determine which of these parameters change and, if possible, describe how.

(a) The signal $x(n)$ is scaled by a constant C,

$$x'(n) = Cx(n)$$

(b) The signal $x(n)$ is modulated by $(-1)^n$,

$$x'(n) = (-1)^n x(n)$$

5.11. The autocorrelation of a signal consisting of a random phase sinusoid in noise is

$$r_x(k) = P\cos(k\omega_0) + \sigma_w^2\delta(k)$$

where ω_0 is the frequency of the sinusoid, P is the power, and σ_w^2 is the variance of the noise. Suppose that we fit an AR(2) model to the data.

(a) Find the coefficients $\mathbf{a}_2 = [1, a_2(1), a_2(2)]^T$ of the AR(2) model as a function of ω_0, σ_w^2, and P.

(b) Find the reflection coefficients, Γ_1 and Γ_2, corresponding to the AR(2) model.

(c) What are the limiting values of the AR(2) parameters and the reflection coefficients as $\sigma_w^2 \to 0$?

5.12. Given that $r_x(0) = 1$ and that the first three reflection coefficients are $\Gamma_1 = 0.5$, $\Gamma_2 = 0.5$, and $\Gamma_3 = 0.25$,

(a) Find the corresponding autocorrelation sequence $r_x(1), r_x(2), r_x(3)$.

(b) Find the autocorrelation sequence $r_x(1), r_x(2), r_x(3)$ for the case in which the reflection coefficient Γ_3 is a free variable, i.e., solve for the autocorrelation values as a function of Γ_3.

(c) Repeat part (b) when both Γ_2 and Γ_3 are free parameters.

5.13. Using the autocorrelation method, an all-pole model of the form

$$H(z) = \frac{b(0)}{1 + a(1)z^{-1} + a(2)z^{-2} + a(3)z^{-3}}$$

has been found for a signal $x(n)$. The constant in the numerator has been chosen so that the autocorrelation of the signal matches the autocorrelation of the model, i.e.,

$$r_x(k) = r_h(k)$$

where $r_h(k)$ is the autocorrelation of $h(n)$. In addition, you know the following about the signal and the model:

1. $r_x(0) = 4$
2. $\Gamma_3 = 0.5$
3. $\Gamma_1 > 0$ and $\Gamma_2 > 0$
4. $x(0) = 1$
5. $\epsilon_3 = 11/16$, where ϵ_3 is the third-order modeling error.
6. $\det(\mathbf{R}_2) = 27$, where $\mathbf{R}_2 = \text{Toep}\{r_x(0), r_x(1), r_x(2)\}$.

Find the values for the model parameters

$$a(1), \quad a(2), \quad a(3), \quad \text{and} \quad b(0)$$

5.14. The first seven values of the unit sample response, $h(n)$, of a 3rd-order linear shift-invariant filter

$$H(z) = A\frac{1 + b(1)z^{-1} + b(2)z^{-2} + b(3)z^{-3}}{1 + a(1)z^{-1} + a(2)z^{-2} + a(3)z^{-3}}$$

are given by

$$\mathbf{h} = \left[1, \ 1/4, \ 1/2, \ 1, \ 0, \ 0, \ 7/8\right]^T$$

Determine whether or not the filter is stable. If more information is needed, state what is required and make the necessary assumptions.

5.15. The extendibility problem in power spectrum estimation concerns the issue of whether or not a finite length sequence,

$$r_x(1), r_x(2), r_x(3), \ldots, r_x(p)$$

may be extended (extrapolated) into a legitimate autocorrelation sequence so that

$$P_x(e^{j\omega}) = \sum_{k=-\infty}^{\infty} r_x(k)e^{-jk\omega} \qquad \text{(P5.15-1)}$$

is a valid power spectrum. In other words, is it possible to find values of $r_x(k)$ for $|k| > p$ such that $P_x(e^{j\omega})$ in (P5.15-1) is a non-negative real function of ω?

(a) Develop a procedure that uses Levinson's recursion to test the extendibility of a sequence.

(b) Use your procedure developed in (a) to determine the constraints on a and b that are necessary and sufficient in order for the sequence

$$r_x(0) = 1, \qquad r_x(1) = a, \qquad r_x(2) = b$$

to be an extendible sequence.

(c) Assuming that the sequence in (b) is extendible, find two different legitimate extensions.

5.16. Which of the following autocorrelation sequences are extendible? For those that are extendible, find an extension and determine whether or not the extension is unique.

(a) $\mathbf{r}_x = [1.0,\ 0.6,\ 0.6]^T$

(b) $\mathbf{r}_x = [1.0,\ 0.6,\ -0.6]^T$

(c) $\mathbf{r}_x = [1.0,\ 0.0,\ 1.0]^T$

5.17. Let $\mathbf{R}_3$ be the symmetric Toeplitz matrix formed from the autocorrelation sequence $r_x(0)$, $r_x(1)$, $r_x(2)$, and $r_x(3)$. If the reflection coefficients that result from applying the Levinson-Durbin recursion to $\mathbf{R}_3$ are

$$\Gamma_1 = \tfrac{1}{2} \qquad \Gamma_2 = \tfrac{1}{3} \qquad \Gamma_3 = \tfrac{1}{4}$$

and if $r_x(0) = 1$, find the determinant of $\mathbf{R}_3$.

5.18. Let $r_x(k) = \sigma_x^2 \delta(k) + 1$. Find the reflection coefficients Γ_k for all $k \geq 0$ and find the all-pole models $A_p(k)$ for all $p \geq 1$.

5.19. A pth-order all-pole model for a signal $x(n)$ is parameterized by the $p+1$ parameters ϵ_p and $a_p(k)$. Since the reflection coefficients Γ_k for a stable model are bounded by one in magnitude, they are automatically scaled. In speech processing, therefore, there has been an interest in coding a speech waveform in terms of its reflection coefficient sequence. The relationship between a reflection coefficient sequence and the spectrum, however, is not easily discernable. Consider, for example, the following three reflection coefficient sequences

1. $\Gamma_k = \dfrac{1}{k+1}$
2. $\Gamma_k = -\dfrac{1}{k+1}$
3. $\Gamma_k = \dfrac{(-1)^k}{k+1}$

Although the only difference between these reflection coefficient sequences is in terms of the sign of Γ_k, the power spectra are quite different. For each of these sequences, find the corresponding autocorrelation sequence, $r_x(k)$, and power spectrum, $P_x(e^{j\omega})$, in closed form.

5.20. Let $x(n)$ be a random process with autocorrelation sequence

$$r_x(k) = (0.2)^{|k|}$$

(a) Find the reflection coefficients Γ_1 and Γ_2 for a second-order predictor and draw the lattice filter network.

(b) Suppose that uncorrelated white noise with a variance of $\sigma_w^2 = 0.1$ is added to $x(n)$,

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

How do the reflection coefficients change?

(c) Can you make any general statements about the effect on the reflection coefficients when white noise is added to a process?

5.21. The reflection coefficients corresponding to a third-order all-pole model are

$$\Gamma_1 = 0.25 \qquad \Gamma_2 = 0.50 \qquad \Gamma_3 = 0.25$$

and the modeling error is given by

$$\epsilon_3 = (15/16)^2$$

(a) Find the direct form filter coefficients, $a_3(k)$, for this third-order model.

(b) Find the autocorrelation values $r_x(0)$, $r_x(1)$, $r_x(2)$, and $r_x(3)$ that led to this model.

(c) If a fourth-order model were found for $x(n)$, what value or values for $r_x(4)$ result in the minimum model error, ϵ_4?

(d) Repeat part (c) and find the value or values for $r_x(4)$ that produce the maximum error, ϵ_4.

5.22. The reflection coefficients for a two-pole model of a signal $x(n)$ are $\Gamma_1 = 0.25$ and $\Gamma_2 = 0.25$ and the "modeling error" is $\epsilon_2 = 9$.

(a) If $r_x(3) = 1$ find the modeling error, ϵ_3, for a three-pole model.

(b) If the signal values, $x(n)$, are multiplied by $1/2$, i.e., $y(n) = 0.5x(n)$, find the reflection coefficients and the modeling error for a two-pole model of $y(n)$.

5.23. You are given the following sequence of autocorrelation values

$$\mathbf{r}_x = \begin{bmatrix} 10, & -1, & 0.1, & -1 \end{bmatrix}^T$$

(a) Use the Schur recursion to find the reflection coefficient sequence $\Gamma_1, \Gamma_2, \Gamma_3$.

(b) What is the final generator matrix, $\mathbf{G}_3$, equal to?

(c) Find the modeling error, ϵ_3.

5.24. Let $r_x(0), r_x(1), \ldots, r_x(p)$ be a set of autocorrelation values and let $\mathbf{R}_p$ be the corresponding $(p+1) \times (p+1)$ autocorrelation matrix. Show that

$$1 - |\Gamma_p|^2 = \frac{\det \mathbf{R}_p \det \mathbf{R}_{p-2}}{\left[\det \mathbf{R}_{p-1}\right]^2}$$

where Γ_p is the pth reflection coefficient.

5.25. If $|\Gamma_j| < 1$, derive a bound for the coefficients δ_j in the split Levinson recursion.

5.26. Let $h_{n_0}(n)$ be the FIR least squares inverse filter of length N with delay n_0 for a sequence $g(n)$, i.e.,

$$h_{n_0}(n) * g(n) \approx \delta(n - n_0)$$

The coefficients $h_{n_0}(n)$ are the solution to the Toeplitz equations (see p. 174 in Chapter 4)

$$\mathbf{R}_g \mathbf{h}_{n_0} = \mathbf{g}_{n_0} \qquad \text{(P5.28-1)}$$

which may be solved efficiently using the Levinson recursion. Since the value for the delay n_0 that produces the smallest least squares error is typically unknown, to find the optimum value for n_0 these equations must be solved for each value of n_0, beginning with $n_0 = 0$. Instead of using the Levinson recursion to solve these equations repeatedly, it is possible to

take advantage of the relationship between $\mathbf{g}_{n_0}$ and $\mathbf{g}_{n_0+1}$,

$$\mathbf{g}_{n_0} = \begin{bmatrix} g^*(n_0) \\ g^*(n_0-1) \\ \vdots \\ g^*(0) \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad ; \cdot \quad \mathbf{g}_{n_0+1} = \begin{bmatrix} g^*(n_0+1) \\ g^*(n_0) \\ \vdots \\ g^*(1) \\ g^*(0) \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

to derive a recursion for $h_{n_0}(n)$. In this problem we derive this recursion which is known as the *Simpson sideways recursion* [26].

(a) The solution to Eq. (P5.28-1) for $n_0 = 0$ may be found using the Levinson-Durbin recursion. Show how to generate the solution for $n_0 = 1$ from the solution for $n_0 = 0$ in less than $4N$ multiplications and divisions where N is the length of the inverse filter $\mathbf{h}_{n_0}$. Note that any information generated in the Levinson-Durbin recursion (for $n_0 = 0$) can be used to construct the new solution.

(b) Generalize the result of part (a) to obtain a recursion that will successively construct the solution for all $n_0 > 0$. Again your method should have less than $4N$ multiplications and divisions at each step.

(c) Write an expression for the error $\mathcal{E}_{n_0}$ at the n_0th step of the recursion in terms of the coefficients $g(n)$ and the coefficients of the least squares inverse filter $h_{n_0}(n)$.

(d) Write a MATLAB program that implements the Simpson sideways recursion.

(e) How can this recursion be used to find the inverse of a Toeplitz matrix?

Computer Exercises

C5.1. Modify the m-file `rtog.m` to find the Cholesky (LDU) decomposition of a Hermitian Toeplitz matrix $\mathbf{R}_x$. Compare the efficiency of your m-file with the MATLAB Cholesky decomposition program `chol.m`.

C5.2. The inverse Levinson-Durbin recursion is a mapping from the sequence of reflection coefficients Γ_j to the autocorrelation sequence $r_x(k)$. Beginning with $r_x(0)$, the autocorrelations $r_x(k)$ for $k > 0$ are determined recursively. In this problem, we derive and implement another approach for the recovery of the autocorrelation sequence.

(a) Beginning with the normal equations

$$\mathbf{R}_p\mathbf{a}_p = \epsilon_p\mathbf{u}_1$$

rewrite these equations in the form

$$\mathbf{A}_p\mathbf{r}_p = \epsilon_p\mathbf{u}_1$$

where $\mathbf{A}_p$ is a matrix containing the pth-order model coefficients $a_p(k)$ and $\mathbf{r}_p$ is the

vector containing the autocorrelation values that are to be determined (assume that the data is complex). What is the structure of the matrix $\mathbf{A}_p$ and how may it be generated?

(b) Write a MATLAB program that will set up these equations and solve for the autocorrelation sequence.

(c) Compare the complexity of this solution to the inverse Levinson-Durbin recursion.

C5.3. The Schur recursion is a mapping from the autocorrelation sequence $r_x(k)$ to a set of reflection coefficients.

(a) Write a MATLAB program that implements the Schur recursion. Using the `flops` command, compare the efficiency of the Schur recursion to the Levinson-Durbin recursion to find the reflection coefficients Γ_j from an autocorrelation sequence $r_x(k)$.

(b) Derive the *inverse Schur recursion* that produces the autocorrelation sequence $r_x(k)$ from the reflection coefficients Γ_j, and write a MATLAB program to implement this recursion. Compare the efficiency of the inverse Schur recursion to the inverse Levinson-Durbin recursion to find the autocorrelation sequence from the reflection coefficients.

C5.4. In the derivation of the split Levinson recursion, we defined a new set of coefficients, δ_j. As we will see in Chapter 6, these coefficients are used in the implementation of a split lattice filter.

(a) Show that these two sets of coefficients are equivalent in the sense that one set may be derived from the other,

$$\{\Gamma_1, \Gamma_2, \ldots, \Gamma_p\} \longleftrightarrow \{\delta_1, \delta_2, \ldots, \delta_p\}$$

(b) Write a MATLAB program `gtod.m` that will convert the reflection coefficients Γ_j into the split Levinson coefficients δ_j.

(c) Write a MATLAB program `dtog.m` that will convert the split Levinson coefficients δ_j into a set of reflection coefficients Γ_j.

(d) Use these m-files to study the relationship between the coefficients Γ_j and δ_j. For example, what does the constraint $|\Gamma_j| < 1$ imply about δ_j? What happens if $\Gamma_p = \pm 1$? Is it possible to determine whether or not $|\Gamma_j| < 1$ by simply looking at δ_j?

C5.5. The *line spectral pair* (LSP) coefficients were introduced in the 1980's as an alternative to the filter coefficients $a_p(k)$ and the reflection coefficients Γ_j in the representation of an all-pole model for a signal $x(n)$. Given the prediction error filter $A_p(z)$, the LSP representation is as follows. With $S_j(z)$ and $S_j^\star(z)$ the singular predictor polynomials that were introduced in the derivation of the split Levinson recursion, note that $A_p(z)$ may be represented in terms of $S_{p+1}(z)$ and $S_{p+1}^\star(z)$ as follows,

$$A_p(z) = \frac{1}{2}\left[S_{p+1}(z) + S_{p+1}^\star(z)\right]$$

Since the roots of $S_{p+1}(z)$ and $S_{p+1}^\star(z)$ lie on the unit circle, the angles of these zeros uniquely define the singular predictor polynomials and, thus, the prediction error filter $A_p(z)$.

(a) Generate several different singular predictor polynomials, $S_{p+1}(z)$ and $S_{p+1}^\star(z)$, and look at the locations of the roots of these polynomials. What relationship do you observe between the locations of the roots of $S_{p+1}(z)$ and the roots of $S_{p+1}^\star(z)$?

(b) Each zero of $A_p(z)$ maps into one zero in each of the polynomials $S_{p+1}(z)$ and $S^{\star}_{p+1}(z)$. Investigate what happens when a zero of $S_{p+1}(z)$ is close to a zero of $S^{\star}_{p+1}(z)$. Does this imply anything about the spectrum $A_p(e^{j\omega})$?

(c) Let θ_k denote the angles of the roots of the polynomials $S_{p+1}(z)$ and $S^{\star}_{p+1}(z)$ that lie within the interval $[0, \pi]$. Assume that the angles are ordered so that $\theta_{k+1} \geq \theta_k$, and let $\Delta\theta_k = \theta_{k+1} - \theta_k$. Write a MATLAB program `lsp.m` that will produce the prediction error filter $A_p(z)$ from the angles $\Delta\theta_k$.

(d) Generate a number of different AR(12) processes by filtering unit variance white noise with a 12th-order all-pole filter. Place the poles of the filter close to the unit circle, at approximately the same radius, and at angles that are approximately harmonically related to each other. Compare the accuracy in the model for $x(n)$ when quantizing the model coefficients $a_p(k)$ to 16 bits and quantizing the LSP coefficients $\Delta\theta_k$ to 16 bits.

LATTICE FILTERS 6

6.1 INTRODUCTION

In Chapter 5 we discovered how the reflection coefficients from the Levinson-Durbin recursion may be used in a lattice filter implementation of the *inverse filter* $A_p(z)$. This structure has a number of interesting and important properties including modularity, low sensitivity to parameter quantization effects, and a simple test for ensuring that $A_p(z)$ is minimum phase. As a result, lattice filters have found their way into many different and important signal processing applications and have been used both for signal analysis and synthesis. In this chapter, we look again at the lattice filter structure, derive some alternative lattice filter forms, and explore how these filters may be used for signal modeling.

This chapter begins with a derivation of the FIR lattice filter structure in Section 6.2 where we explore the relationship between the lattice filter and the problems of forward and backward linear prediction. In Section 6.3 we then develop the split lattice filter which is based on the singular predictor polynomials and the split Levinson recursion. All-pole lattice filters, allpass lattice filters, and pole-zero lattice filters are then considered in Section 6.4. In Section 6.5 we then turn our attention to the use of lattice filters for all-pole signal modeling. We derive three approaches that are based on the sequential estimation of the reflection coefficients. These methods are the forward covariance method, the backward covariance method, and Burg's method. We also present a nonsequential method known as the modified covariance method. Finally, in Section 6.6 we look at how the lattice methods of signal modeling may be used to model stochastic signals.

6.2 THE FIR LATTICE FILTER

To derive the FIR lattice filter structure, we will proceed exactly as we did in Section 5.2.2, beginning with the problem of all-pole signal modeling. Therefore, let $x(n)$ be a signal that is to be modeled as the unit sample response of an all-pole filter of the form

$$H(z) = \frac{b(0)}{A_p(z)}$$

As we saw in Section 4.4.3 of Chapter 4, with Prony's method the coefficients of $A_p(z)$ are found by minimizing the squared error,[1]

$$\mathcal{E}_p = \sum_{n=0}^{\infty} |e_p(n)|^2$$

where

$$e_p(n) = x(n) * a_p(n) = x(n) + \sum_{k=1}^{p} a_p(k)x(n-k)$$

with $a_p(0) = 1$. Note that by minimizing $\mathcal{E}_p$ we are solving for the coefficients $a_p(k)$ that minimize the difference between $x(n)$ and what may be considered to be an estimate or *prediction* of $x(n)$,

$$\hat{x}(n) = -\sum_{k=1}^{p} a_p(k)x(n-k)$$

This estimate is formed by taking a linear combination of the previous p values of $x(n)$ and is referred to as the forward predictor of $x(n)$ and

$$e_p(n) = x(n) - \hat{x}(n)$$

is referred to as the pth-order *forward prediction error*. In order to distinguish between the *forward prediction error* and the *backward prediction error*, which will be introduced shortly, we will use the following notation,

$$\boxed{e_p^+(n) = x(n) + \sum_{k=1}^{p} a_p(k)x(n-k)} \tag{6.1}$$

In addition, we will use $\mathcal{E}_p^+$ to denote the sum of the squares of $e_p^+(n)$,

$$\boxed{\mathcal{E}_p^+ = \sum_{n=0}^{\infty} |e_p^+(n)|^2} \tag{6.2}$$

Equation (6.1) allows us to write the pth-order forward prediction error in the z-domain as follows

$$E_p^+(z) = A_p(z)X(z) \tag{6.3}$$

where

$$A_p(z) = 1 + \sum_{k=1}^{p} a_p(k)z^{-k} \tag{6.4}$$

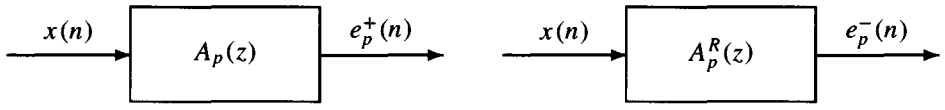
Therefore, $e_p^+(n)$ may be generated by filtering $x(n)$ with the FIR filter $A_p(z)$ which is referred to as the *forward prediction error filter*. This relationship between $e_p^+(n)$ and $x(n)$ is illustrated in Fig. 6.1*a*.

As we saw in Section 4.4.3, the coefficients of the forward prediction error filter are the solution to the normal equations

$$\mathbf{R}_p\mathbf{a}_p = \epsilon_p\mathbf{u}_1 \tag{6.5}$$

where $\mathbf{R}_p$ is a Hermitian Toeplitz matrix of autocorrelations. As we have seen in Chapter 5,

[1]Note that we have added a subscript p to $e(n)$ to indicate the order of the model.



(a) (b)

Figure 6.1 *(a) The forward prediction error, $e_p^+(n)$, expressed as the output of the forward prediction error filter, $A_p(z)$ and (b) the backward prediction error, $e_p^-(n)$, expressed as the output of the backward prediction error filter, $A_p^R(z)$.*

these equations may be solved recursively using the Levinson-Durbin recursion. Thus, the coefficients of the $(j+1)$st order prediction error filter, $a_{j+1}(i)$, are related to the coefficients of the jth order prediction error filter through the model-order update equation given in Eq. (5.16) which is repeated below for convenience,

$$a_{j+1}(i) = a_j(i) + \Gamma_{j+1} a_j^*(j - i + 1) \tag{6.6}$$

Expressing Eq. (6.6) in the z-domain, we may relate the prediction error filter $A_{j+1}(z)$ to $A_j(z)$ as follows

$$A_{j+1}(z) = A_j(z) + \Gamma_{j+1}\left[z^{-(j+1)} A_j^*(1/z^*)\right] \tag{6.7}$$

Multiplying both sides of Eq. (6.7) by $X(z)$ and using Eq. (6.3) we obtain the following relationship between $E_{j+1}^+(z)$ and $E_j^+(z)$,

$$E_{j+1}^+(z) = E_j^+(z) + z^{-1}\Gamma_{j+1} E_j^-(z) \tag{6.8}$$

where we have defined $E_j^-(z)$ as follows

$$E_j^-(z) = z^{-j} X(z) A_j^*(1/z^*) \tag{6.9}$$

If, as in Chapter 5, we introduce the notation $A_j^R(z) = z^{-j} A_j^*(1/z^*)$, then $E_j^-(z)$ may be written as follows

$$\boxed{E_j^-(z) = A_j^R(z) X(z)} \tag{6.10}$$

Taking the inverse z-transform of both sides of Eq. (6.8), we obtain the time-domain recursion

$$\boxed{e_{j+1}^+(n) = e_j^+(n) + \Gamma_{j+1} e_j^-(n-1)} \tag{6.11}$$

that relates $e_{j+1}^+(n)$ to $e_j^+(n)$ and $e_j^-(n)$.

A signal processing interpretation of the signal $e_j^-(n)$ may be derived from Eq. (6.9) by taking the inverse z-transform as follows,

$$e_j^-(n) = x(n-j) + \sum_{k=1}^{j} a_j^*(k) x(n - j + k) \tag{6.12}$$

As we did with the forward prediction error, $e_j^-(n)$ may be expressed as the difference between $x(n-j)$ and what may be considered to be the estimate or *prediction* of $x(n-j)$ that is formed by taking a linear combination of the j signal values $x(n), x(n-1), \ldots, x(n-j+1)$, i.e.,

$$e_j^-(n) = x(n-j) - \hat{x}(n-j)$$

where

$$\hat{x}(n-j) = -\sum_{k=1}^{j} a_j^*(k)x(n-j+k)$$

What is interesting about this interpretation is that if we minimize the sum of the squares of $e_j^-(n)$,

$$\mathcal{E}_j^- = \sum_{n=0}^{\infty} |e_j^-(n)|^2$$

then we find that the coefficients that minimize $\mathcal{E}_j^-$ are the same as those that minimize $\mathcal{E}_j^+$ and therefore are found by solving the normal equations given in Eq. (6.5). Thus, $e_j^-(n)$ is referred to as the jth-order *backward prediction error* and $A_j^R(z)$ is known as the *backward prediction error filter*. The relationship between $x(n)$ and $e_j^-(n)$ is illustrated in Fig. 6.1*b*.

Equation (6.11) provides a recursion for the $(j+1)$st-order forward prediction error in terms of the jth-order forward and backward prediction errors. A similar recursion may be derived for the backward prediction error as follows. Taking the complex conjugates of both sides of Eq. (6.6) and substituting $j-i+1$ for i we have

$$a_{j+1}^*(j-i+1) = a_j^*(j-i+1) + \Gamma_{j+1}^* a_j(i) \tag{6.13}$$

Expressing Eq. (6.13) in the z-domain we find that

$$z^{-(j+1)}A_{j+1}^*(1/z^*) = z^{-(j+1)}A_j^*(1/z^*) + \Gamma_{j+1}^* A_j(z) \tag{6.14}$$

Multiplying both sides of Eq. (6.14) by $X(z)$ and using the definitions for $E_j^+(z)$ and $E_j^-(z)$ in Eqs. (6.3) and (6.9), respectively, yields

$$E_{j+1}^-(z) = z^{-1}E_j^-(z) + \Gamma_{j+1}^* E_j^+(z) \tag{6.15}$$

Finally, taking the inverse z-transform we obtain the desired recursion

$$\boxed{e_{j+1}^-(n) = e_j^-(n-1) + \Gamma_{j+1}^* e_j^+(n)} \tag{6.16}$$

Equations (6.11) and (6.16) represent a pair of coupled difference equations that correspond to the two-port network shown in Fig. 6.2*a*. With a cascade of two-port networks having reflection coefficients Γ_j, we have the pth-order FIR lattice filter as shown in Fig. 6.2*b*. Note that since

$$e_0^+(n) = e_0^-(n) = x(n) \tag{6.17}$$

then the two inputs to the first stage of the lattice are the same.

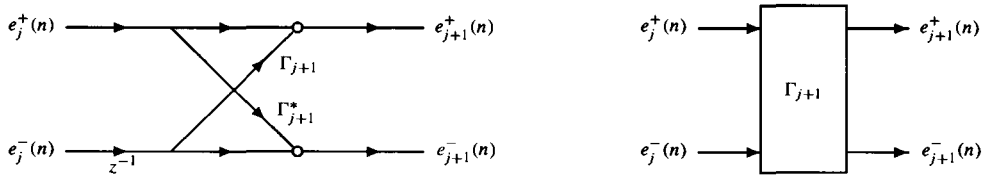
It is interesting to note that with a forward prediction error filter of the form

$$A_p(z) = \prod_{i=1}^{p}\left(1 - \alpha_i z^{-1}\right) \tag{6.18}$$

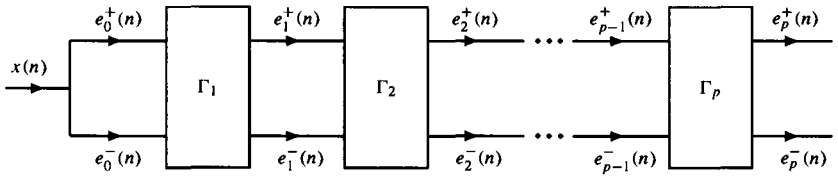
the system function of the backward prediction error filter is

$$A_p^R(z) = z^{-p}A_p^*(1/z^*) = \prod_{i=1}^{p}\left(z^{-1} - \alpha_i^*\right) \tag{6.19}$$

Therefore, the zeros of $A_p(z)$ are at the conjugate reciprocal locations of those of $A_p^R(z)$, i.e., if there is a zero at $z = \alpha_i$ in $A_p(z)$ then there is a zero at $z = 1/\alpha_i^*$ in $A_p^R(z)$. Consequently,



(a) Single stage of an FIR lattice filter.



(b) A pth-order FIR lattice filter.

Figure 6.2 *The FIR lattice filter.*

since the forward prediction error filter that is obtained from the autocorrelation method is minimum phase (assuming $\mathbf{R}_p > 0$) then the backward prediction error filter is ***maximum phase***. Another interesting relationship to note is that since

$$E_p^+(z) = A_p(z)X(z)$$

and

$$E_p^-(z) = A_p^R(z)X(z)$$

then

$$E_p^-(z) = \frac{A_p^R(z)}{A_p(z)}E_p^+(z) = \prod_{i=1}^{p}\left[\frac{z^{-1} - \alpha_i^*}{1 - \alpha_i z^{-1}}\right]E_p^+(z) = H_{ap}(z)E_p^+(z) \tag{6.20}$$

where

$$H_{ap}(z) = \prod_{i=1}^{p}\frac{z^{-1} - \alpha_i^*}{1 - \alpha_i z^{-1}} \tag{6.21}$$

is an allpass filter. Thus, as illustrated in Fig. 6.3, the backward prediction error may be generated by filtering the forward prediction error with an allpass filter.[2] Note that when $A_p(z)$ is minimum phase, all of the poles of the allpass filter are inside the unit circle and all of the zeros are outside. Therefore, $H_{ap}(z)$ is a stable and causal filter.

There are a number of advantages of a lattice filter over a direct form filter that often make it a popular structure to use in signal processing applications. The first is the modularity of the filter. It is this modularity that allows one to increase or decrease the order of a lattice filter in a linear prediction or all-pole signal modeling application without having to recompute the reflection coefficients. By contrast, when the prediction error filter is implemented in direct form and if the filter order is increased or decreased then all of the filter coefficients, in general, will change. Lattice filters also have the advantage of being easy to determine whether or not the filter is minimum phase (all of the roots inside the unit circle). Specifically, the filter will be minimum phase if and only if the reflection

[2] An allpass lattice filter for computing $e_p^-(n)$ from $e_p^+(n)$ will be derived in Section 6.4.1.

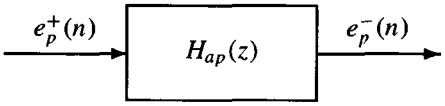


Figure 6.3 *Allpass filter for generating the backward prediction error $e_p^-(n)$ from the forward prediction error $e_p^+(n)$.*

coefficients are bounded by one in magnitude. In the case of IIR lattice filters, which will be discussed in the following section, this allows one to easily ensure that a filter is stable by simply imposing the constraint that $|\Gamma_j| < 1$. This is particularly useful in an adaptive filtering environment in which the reflection coefficients are changing as a function of time (See Chapter 9). This minimum phase condition thus allows one to easily update the filter coefficients in time in such a way that, for each value of n, the roots of the filter remain inside the unit circle. Finally, compared to other filter structures, the lattice filter tends to be less sensitive to parameter quantization effects [13]. Therefore, a lattice filter is frequently used when it is important to minimize the parameter quantization effects or when a signal is to be *coded* in terms of the filter coefficients of the model that provides the best approximation to the signal. There are other forms of FIR lattice filter, including the one-multiplier and the normalized lattice filters [7]. In the following section, we derive the split lattice filter.

6.3 SPLIT LATTICE FILTER*

In Section 5.4 of Chapter 5, the singular predictor polynomials were introduced as a way of imposing symmetry in the prediction error filters $A_j(z)$, thereby allowing for an increase in the efficiency of the Levinson-Durbin recursion. These polynomials were defined to be

$$S_j(z) = A_{j-1}(z) + z^{-1}A_{j-1}^R(z)$$

and they were shown to satisfy the 3-term recurrence

$$S_{j+1}(z) = (1 + z^{-1})S_j(z) - \delta_j z^{-1} S_{j-1}(z)$$

We now show how this recurrence leads to a *split lattice filter* structure [4]. As in Section 5.4 where we derived the split Levinson algorithm, here we restrict ourselves to the case of real signals.

Paralleling our development of the FIR lattice filter in the previous section, note that if we were to filter $x(n)$ with $S_j(z)$, then the filter's response

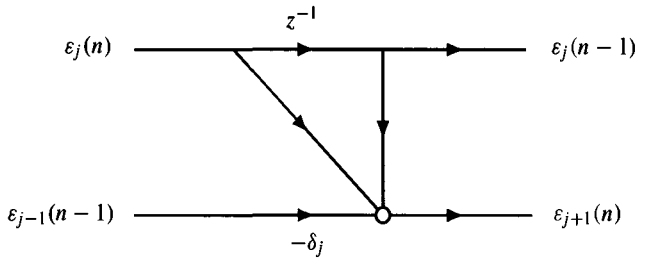
$$\varepsilon_j(n) = s_j(n) * x(n)$$

will be the forward prediction error that would result if, at the jth stage of the Levinson-Durbin recursion, $\Gamma_j = 1$. Therefore, we will refer to $\varepsilon_j(n)$ as the jth-order *singular forward prediction error*. Using the 3-term recurrence for $S_j(z)$, it follows that $\varepsilon_j(n)$ satisfies the difference equation

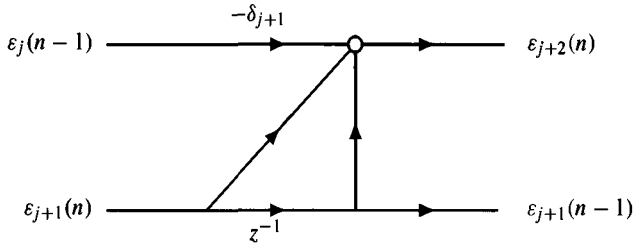
$$\varepsilon_{j+1}(n) = \varepsilon_j(n) + \varepsilon_j(n-1) - \delta_j \varepsilon_{j-1}(n-1)$$

This relationship, shown in block diagram form in Fig. 6.4*a*, represents one stage of a split lattice filter. The next stage, which takes the two inputs $\varepsilon_{j+1}(n)$ and $\varepsilon_j(n-1)$ and produces the outputs $\varepsilon_{j+2}(n)$ and $\varepsilon_{j+1}(n-1)$, is defined by the equation

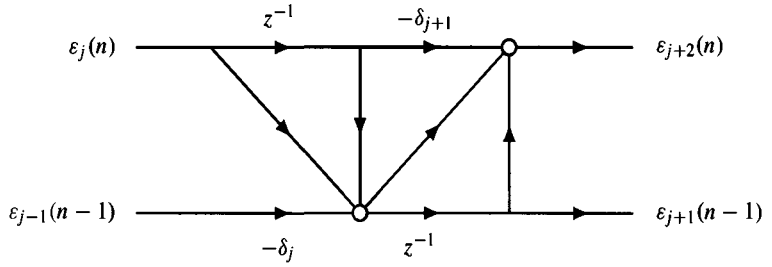
$$\varepsilon_{j+2}(n) = \varepsilon_{j+1}(n) + \varepsilon_{j+1}(n-1) - \delta_{j+1}\varepsilon_j(n-1)$$



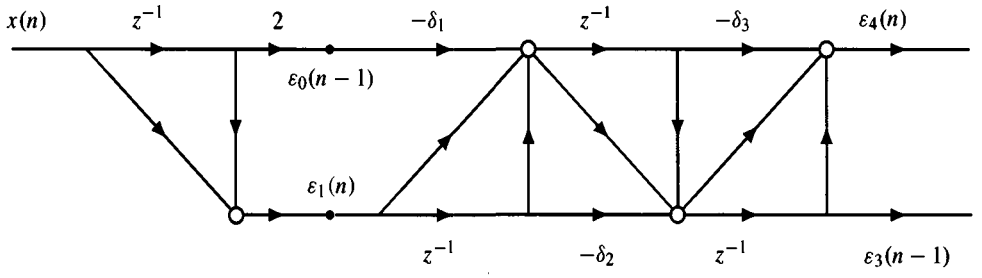
(a) A network corresponding to the 3-term recurrence for the singular predictor polynomials.



(b) A network that produces $\varepsilon_{j+2}(n)$ from $\varepsilon_{j+1}(n)$ and $\varepsilon_j(n-1)$.



(c) A second-order split lattice module that is formed by cascading the networks in (a) and (b).



(d) A 3rd-order split lattice filter that produces the singular prediction errors $\varepsilon_4(n)$ and $\varepsilon_3(n-1)$.

Figure 6.4 *A split lattice filter.*

and is shown in block diagram form in Fig. 6.4*b*. Note that this network is the mirror image of the previous one. Cascading these two sections together produces the second-order split lattice module shown in Fig. 6.4*c*. What is needed to complete the split lattice filter is an initial section to transform $x(n)$ into $\varepsilon_0(n)$ and $\varepsilon_1(n)$, and a final section to convert the singular prediction errors $\varepsilon_{p+1}(n)$ and $\varepsilon_p(n)$ into the forward prediction error $e_p^+(n)$. The initial conditions in the split Levinson recursion are $S_0(z) = 2$ and $S_1(z) = 1 + z^{-1}$ (see p. 273), which require that we form the signals

$$\begin{aligned}\varepsilon_0(n) &= 2x(n)\\ \varepsilon_1(n) &= x(n) + x(n-1)\end{aligned}$$

These signals may be generated as illustrated in Fig. 6.4*d* which shows a 3rd-order split lattice filter for producing the singular prediction errors $\varepsilon_4(n)$ and $\varepsilon_3(n-1)$. Note that a multiplication may be saved if we combine the factor of 2 with δ_1.

Finally, to convert the singular prediction errors $\varepsilon_{p+1}(n)$ and $\varepsilon_p(n)$ at the output of the split lattice filter into the forward prediction error $e_p^+(n)$, recall that $A_p(z)$ is related to the singular predictor polynomials as follows

$$A_p(z) = z^{-1}A_p(z) + S_{p+1}(z) - z^{-1}(1 + \Gamma_p)S_p(z)$$

Therefore, the pth-order forward prediction error may be generated from the singular prediction errors $\varepsilon_p(n)$ and $\varepsilon_{p+1}(n)$ as follows,

$$e_p^+(n) = e_p^+(n-1) + \varepsilon_{p+1}(n) - (1 + \Gamma_p)\varepsilon_p(n-1) \tag{6.22}$$

The following example illustrates the procedure.

Example 6.3.1 *Split Lattice Filter*

Consider the third-order all-pole filter

$$H(z) = \frac{1}{1 - \frac{1}{3}z^{-1} - \frac{1}{3}z^{-2} + \frac{2}{3}z^{-3}}$$

In order to implement this filter using a split lattice filter structure, it is necessary that we derive the coefficients δ_j from the coefficients $a(k)$. This may be done in a straightforward manner as follows. First, we convert the coefficients $a(k)$ into reflection coefficients Γ_j. Using the m-file `atog.m` we find

$$\mathbf{\Gamma} = [-1/4,\ -1/5,\ 2/3]^T$$

Second, using the relationship between Γ_j and δ_j given in Eq. (5.149), i.e.,

$$\delta_j = (1 - \Gamma_j)(1 + \Gamma_{j-1}) \quad ; \quad \Gamma_0 = 0$$

we have

$$\begin{aligned}\delta_1 &= 1 - \Gamma_1 = 5/4\\ \delta_2 &= (1 - \Gamma_2)(1 + \Gamma_1) = 9/10\\ \delta_3 &= (1 - \Gamma_3)(1 + \Gamma_2) = 4/15\end{aligned}$$

Finally, with $\Gamma_3 = \frac{2}{3}$, using the termination specified in Eq. (6.22) we have the split lattice filter shown in Fig. 6.5.

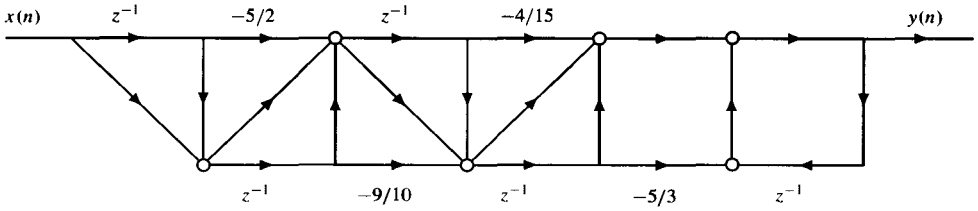


Figure 6.5 *A third-order split lattice filter.*

6.4 IIR LATTICE FILTERS

In Section 6.2 we saw how the reflection coefficients Γ_j could be used to implement an FIR filter $A_p(z)$ using a lattice filter structure. In this section we develop a lattice filter structure for IIR filters. We begin with a derivation of the "standard" two-multiplier all-pole lattice filter for implementing the system function $1/A_p(z)$. At the same time we show how this all-pole lattice may also be used as an allpass filter $H_{ap}(z) = A_p^R(z)/A_p(z)$. We then look at some other structures including the Kelly-Lochbaum lattice filter, the normalized all-pole lattice, and the one-multiplier all-pole lattice. As with the FIR lattice filter, these structures enjoy the same advantages of modularity, simple tests for stability, and decreased sensitivity to parameter quantization effects. Finally, we will look at how the all-pole lattice filter may be modified to implement a general pole-zero filter of the form $H(z) = B_q(z)/A_p(z)$.

6.4.1 All-pole Filter

A pth-order FIR lattice filter is shown in Fig. 6.2*b*. The input to the filter is the zeroth-order forward prediction error $e_0^+(n)$ and the outputs are the pth-order forward and backward prediction errors, $e_p^+(n)$ and $e_p^-(n)$, respectively. The system function of this filter is

$$A_p(z) = \frac{E_p^+(z)}{E_0^+(z)} = 1 + \sum_{k=1}^{p} a_p(k)z^{-k}$$

The all-pole filter

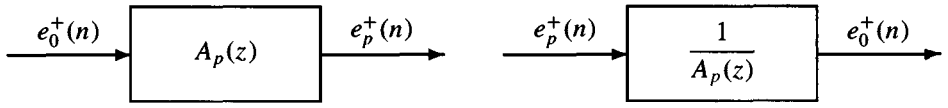
$$\frac{1}{A_p(z)} = \frac{E_0^+(z)}{E_p^+(z)} = \frac{1}{1 + \sum_{k=1}^{p} a_p(k)z^{-k}}$$

on the other hand, would produce a response of $e_0^+(n)$ to the input $e_p^+(n)$. Therefore, as illustrated in Fig. 6.6, whereas the FIR lattice filter builds up the prediction errors $e_j^+(n)$ and $e_j^-(n)$ from $e_0^+(n)$, the all-pole lattice filter produces the lower order prediction errors, $e_j^+(n)$ and $e_j^-(n)$, from $e_p^+(n)$. Thus, in order to implement a pth-order all-pole lattice filter, we must determine how the lower-order prediction errors $e_j^+(n)$ and $e_j^-(n)$ may be generated from $e_p^+(n)$. Note that if we solve Eq. (6.11) for $e_j^+(n)$ we have, along with Eq. (6.16), the following pair of coupled difference equations

$$e_j^+(n) = e_{j+1}^+(n) - \Gamma_{j+1}e_j^-(n-1) \tag{6.23}$$

$$e_{j+1}^-(n) = e_j^-(n-1) + \Gamma_{j+1}^* e_j^+(n) \tag{6.24}$$

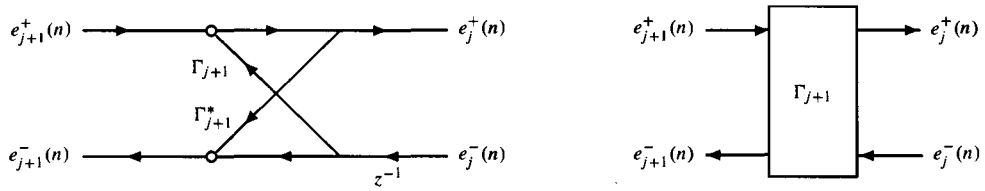
These two equations define the two-port network shown in Fig. 6.7*a* and represent a single stage of an all-pole lattice filter. With a cascade of p such sections we have the pth-order



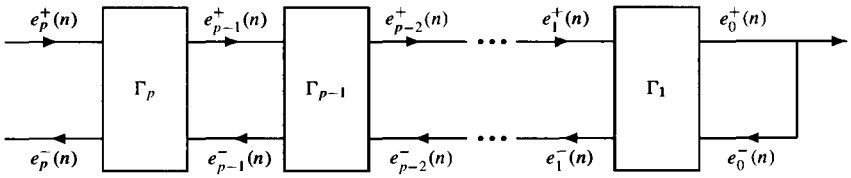
(a) A pth-order all-zero filter for generating $e_p^+(n)$ from $e_0^+(n)$.

(b) A pth-order all-pole filter for generating $e_0^+(n)$ from $e_p^+(n)$.

Figure 6.6 *Generating the pth-order forward prediction error $e_p^+(n)$ from $e_0^+(n)$ and vice versa.*



(a) Single stage of an all-pole lattice filter.



(b) A pth-order all-pole lattice filter and a pth-order allpass filter that is formed by cascading p modules of the form given in (a).

Figure 6.7 *A lattice filter for implementing all-pole and allpass filters.*

all-pole filter shown in Fig. 6.7*b*. Recall that since $e_0^+(n) = e_0^-(n) = x(n)$ then the lattice filter is terminated by feeding $e_0^+(n)$ back into the lower input of the last module. While the system function relating $e_0^+(n)$ to $e_p^+(n)$ is the all-pole filter

$$H(z) = \frac{E_0^+(z)}{E_p^+(z)} = \frac{1}{A_p(z)} \tag{6.25}$$

note that, as we saw in Section 6.2, the system function relating $e_p^+(n)$ to $e_p^-(n)$ is the allpass filter

$$H_{ap}(z) = \frac{E_p^-(z)}{E_p^+(z)} = z^{-p}\frac{A_p^*(1/z^*)}{A_p(z)} = \frac{A_p^R(z)}{A_p(z)} \tag{6.26}$$

Therefore, this structure may also be used to implement allpass systems.

Example 6.4.1 ***All-pole Lattice Filter***

Let us implement the third-order all-pole filter

$$H(z) = \frac{0.328}{1 - 0.8z^{-1} + 0.64z^{-2} - 0.512z^{-3}}$$

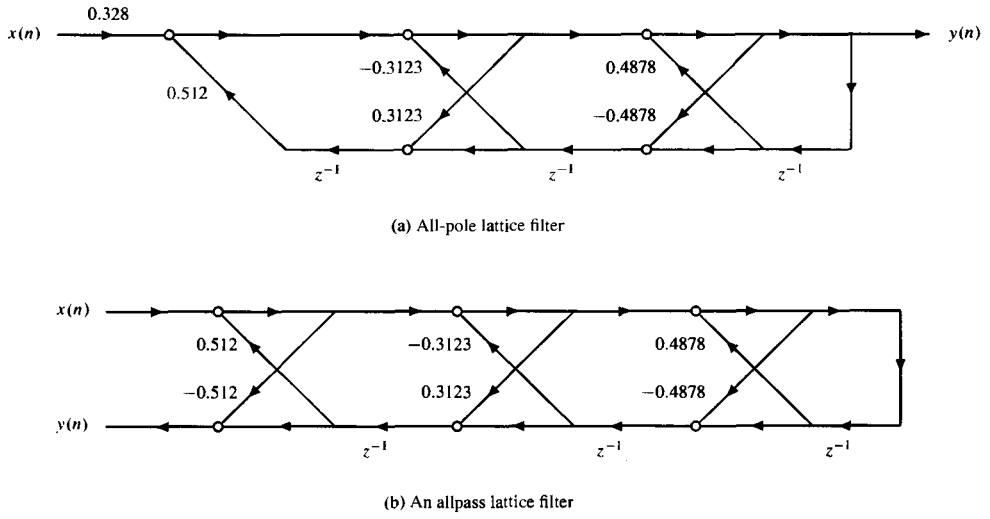


Figure 6.8 *All-pole and allpass lattice filter implementations.*

using a lattice filter structure. With the step-down recursion we find that the reflection coefficients corresponding to the filter coefficients $a_p(k)$ are

$$\Gamma = [-0.4878,\ 0.3123,\ -0.512]^T$$

Thus, the all-pole lattice implementation of $H(z)$ is as illustrated in Fig. 6.8*a*. This structure may also be used to implement the allpass filter

$$H_{ap}(z) = \frac{-0.512 + 0.64z^{-1} - 0.8z^{-2} + z^{-3}}{1 - 0.8z^{-1} + 0.64z^{-2} - 0.512z^{-3}}$$

as illustrated in Fig. 6.8*b*.

6.4.2 Other All-pole Lattice Structures*

Beginning with the all-pole lattice filter in Fig. 6.7, we may develop some additional all-pole lattice filter structures. One such structure may be derived from Eqs. (6.23) and (6.24) as follows. Note that Eq. (6.23) expresses the forward prediction error $e_j^+(n)$ in terms of the two lattice filter inputs $e_{j+1}^+(n)$ and $e_j^-(n-1)$ whereas Eq. (6.24) expresses the backward prediction error $e_{j+1}^-(n)$ in terms of the output $e_j^+(n)$ and the input $e_j^-(n-1)$. However, we may also express $e_{j+1}^-(n)$ in terms of the two inputs $e_{j+1}^+(n)$ and $e_j^-(n-1)$ by substituting Eq. (6.23) for $e_j^+(n)$ into Eq. (6.24) as follows

$$\begin{aligned} e_{j+1}^-(n) &= e_j^-(n-1) + \Gamma_{j+1}^* e_j^+(n) \\ &= e_j^-(n-1) + \Gamma_{j+1}^* \left[e_{j+1}^+(n) - \Gamma_{j+1} e_j^-(n-1) \right] \\ &= \left[1 - |\Gamma_{j+1}|^2 \right] e_j^-(n-1) + \Gamma_{j+1}^* e_{j+1}^+(n) \end{aligned} \tag{6.27}$$

Equation (6.27) along with (6.23) define the two-port network shown in Fig. 6.9. Note that, unlike the all-pole lattice in Fig. 6.7, this implementation requires three multiplies per stage.

Although not a particularly useful structure, through a sequence of flowgraph manipulations described below we may derive some interesting and important all-pole lattice structures. The first is the Kelly-Lochbaum model that has been used in speech modeling as well as models for the propagation of waves through a layered media [11,12]. Next, we derive the normalized all-pole lattice filter which, although requiring four multiplications per section, has an advantage in terms of computational accuracy using fixed-point arithmetic [10]. Finally, we derive the one-multiplier lattice which is the most efficient structure when measured in terms of the number of multiplications per section.

Beginning with the three-multiplier lattice module in Fig. 6.9, suppose we scale the input to the upper branch by a constant $1/\alpha_{j+1}$ and scale the outputs of the branch point by the inverse of this scale factor, α_{j+1}. This produces the equivalent two-port network shown in Fig. 6.10*a*. Next, if we divide the two inputs to the adder in the lower branch by α_{j+1} and multiply the output of the adder by α_{j+1} then we have the equivalent network shown in Fig. 6.10*b*. Now, recall that both of the outputs $e_j^+(n)$ and $e_{j+1}^-(n)$ are formed by taking a linear combination of the two inputs $e_{j+1}^+(n)$ and $e_j^-(n)$. Therefore, if we scale the two inputs by α_{j+1} and rescale the outputs by $1/\alpha_{j+1}$ then the network will remain the same. The net effect, as shown in Fig. 6.10*c*, is to bring the two scale factors on the left of the two-port network over to the right side. Forming a cascade of these modules, moving all of the scale factors to the end of the network, results in the equivalent all-pole lattice filter shown in Fig. 6.10*d* where

$$\alpha = \prod_{j=1}^{p} \alpha_j$$

Since $e_0^+(n) = e_0^-(n)$ then the network may be terminated by feeding the output of the final stage of the lattice back into the input. With the scale factors of α and $1/\alpha$ canceling each other at the input to the last module, we are then left with only a scale factor of $1/\alpha$ at the output that is applied to $e_0^+(n)$. This leads to the structure shown in Fig. 6.11 consisting of a cascade of *modified all-pole lattice modules*. Having dropped the final scale factor $1/\alpha$ from the network, the system function between the input $x(n)$ and the output $y(n)$ is

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\prod_{j=1}^{p} \alpha_j}{A_p(z)} \tag{6.28}$$

and the system function relating $x(n)$ to $w(n)$ is the allpass filter

$$H(z) = \frac{W(z)}{X(z)} = \frac{A_p^R(z)}{A_p(z)} \tag{6.29}$$

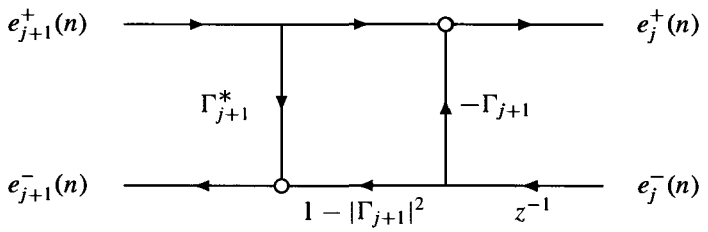
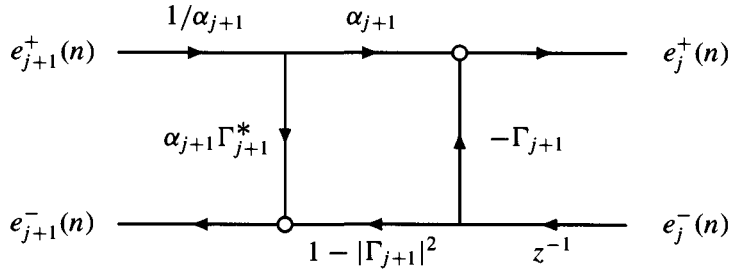
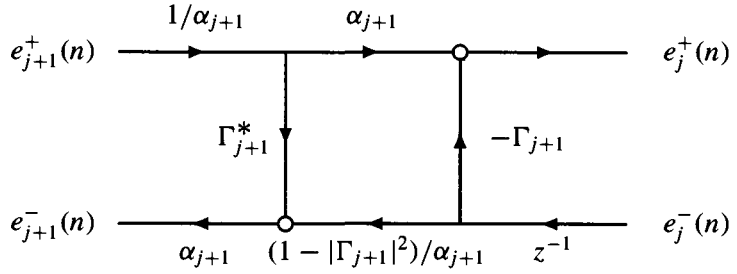


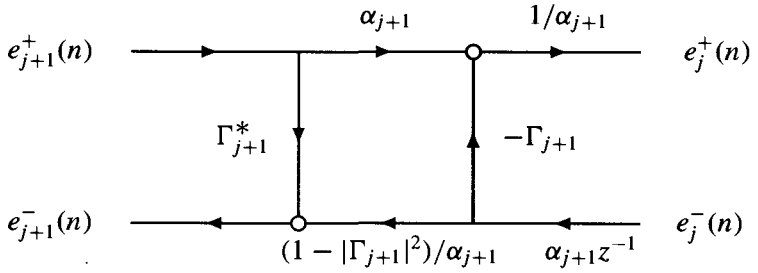
Figure 6.9 *A three multiplier all-pole lattice filter module.*



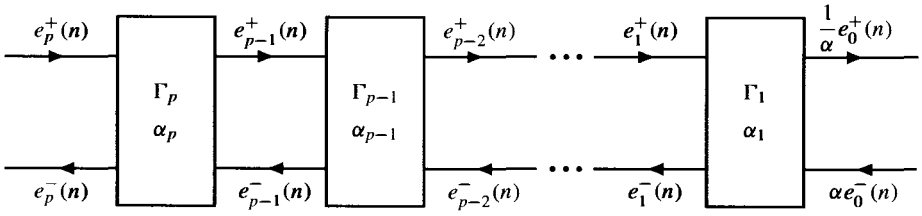
(a) Scaling the input in the upper branch of the all-pole lattice module by $1/\alpha_{j+1}$ and rescaling the outputs of the first branch point by α_{j+1}.



(b) Scaling the inputs to the adder in the lower branch by $1/\alpha_{j+1}$ and rescaling the output by α_{j+1}.

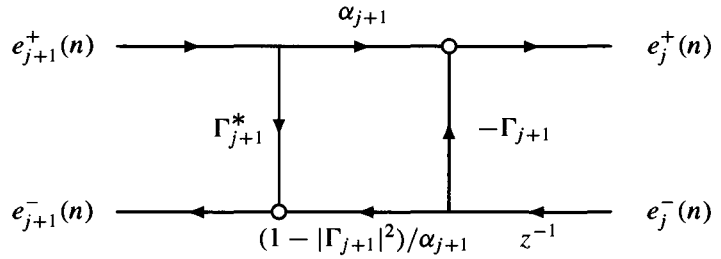


(c) Moving the scale factors α_{j+1} and $1/\alpha_{j+1}$ on the left to the right.

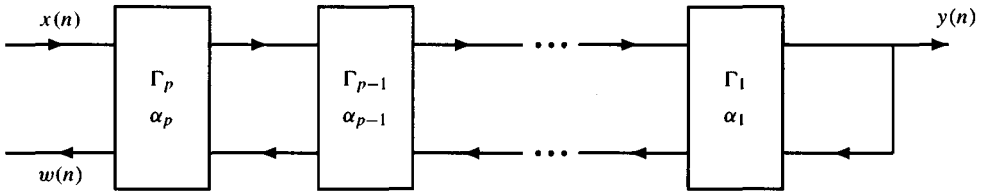


(d) A cascade of lattice filter modules of the form given in (c) with all of the scale factors moved to the final stage.

Figure 6.10 *A sequence of lattice filter flowgraph manipulations that lead to equivalent all-pole lattice filter structures.*



(a) The modified all-pole lattice module.



(b) A cascade of modified all-pole lattice modules.

Figure 6.11 *A modified all-pole lattice filter built-up as a cascade of modified all-pole lattice modules shown in (a).*

Given the all-pole module in Fig. 6.11, we may now derive three important all-pole lattice filter structures. In each case, we assume that the reflection coefficients are real-valued. The first is based on the factorization $(1 - \Gamma^2_{j+1}) = (1 + \Gamma_{j+1})(1 - \Gamma_{j+1})$ and is obtained by setting

$$\alpha_{j+1} = 1 + \Gamma_{j+1} \tag{6.30}$$

This results in the lattice filter module shown in Fig. 6.12*a* which is known as the Kelly-Lochbaum model. This structure has been used as a model for speech production in which the vocal tract is approximated as an interconnection of cylindrical acoustic tubes of equal length and differing cross-sectional areas where the reflection coefficients Γ_j correspond to the amount of acoustic energy that is reflected at the interface between two adjacent tubes having a differing acoustic impedance [3,6]. The difference equations for this structure are

$$\begin{aligned} e^+_j(n) &= [1 + \Gamma_{j+1}]e^+_{j+1}(n) - \Gamma_{j+1}e^-_j(n-1) \\ e^-_{j+1}(n) &= [1 - \Gamma_{j+1}]e^-_j(n-1) + \Gamma_{j+1}e^+_{j+1}(n) \end{aligned} \tag{6.31}$$

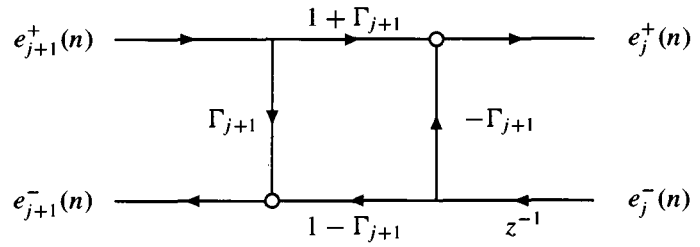
and the system function is

$$H(z) = \frac{\prod_{j=1}^{p}(1 + \Gamma_j)}{A_p(z)} \tag{6.32}$$

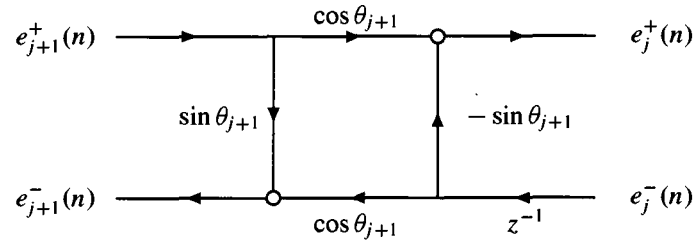
This structure requires four multiplications, two additions, and one delay for each stage.

The next all-pole lattice structure is derived by introducing a new set of variables, θ_{j+1}, where θ_{j+1} is related to the reflection coefficients Γ_{j+1} as follows

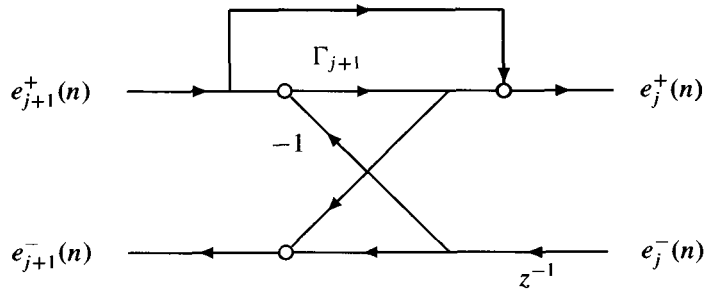
$$\theta_{j+1} = \sin^{-1}\Gamma_{j+1} \tag{6.33}$$



(a) Kelly-Lochbaum lattice filter.



(b) Normalized lattice filter.



(c) One-multiplier lattice filter.

Figure 6.12 *Three all-pole lattice filter structures with real-valued filter coefficients. Each of these lattice modules is equivalent to within a scale factor.*

Assuming that the filter is stable so that $|\Gamma_{j+1}| < 1$, this transformation is well-defined. With

$$\Gamma_{j+1} = \sin\theta_{j+1} \tag{6.34}$$

and

$$1 - \Gamma^2_{j+1} = \cos^2\theta_{j+1} \tag{6.35}$$

let us define the scale factor α_{j+1} by

$$\alpha_{j+1} = \cos\theta_{j+1} \tag{6.36}$$

This leads to the *normalized all-pole lattice filter* shown in Fig. 6.12*b*. The difference equations for this network are

$$\begin{aligned} e_j^+(n) &= [\cos\theta_{j+1}]e_{j+1}^+(n) - [\sin\theta_{j+1}]e_j^-(n-1) \\ e_{j+1}^-(n) &= [\cos\theta_{j+1}]e_j^-(n-1) + [\sin\theta_{j+1}]e_{j+1}^+(n) \end{aligned} \tag{6.37}$$

and the system function is

$$H(z) = \frac{\prod_{j=1}^{p}\cos\theta_j}{A_p(z)} \tag{6.38}$$

In this implementation, there are four multiplications, two additions, and one delay for each stage as in the Kelly-Lochbaum structure. However, what makes this structure interesting is the fact that it requires only one complex multiplication. Specifically, note that if we form the complex signal $e_{i+1}^+(n) + je_i^-(n-1)$ from the two input signals, and the complex signal $e_i^+(n) + je_{i+1}^-(n)$ from the two output signals, then these two signals are related as follows[3]

$$e_i^+(n) + je_{i+1}^-(n) = e^{j\theta_{i+1}}\left[e_{i+1}^+(n) + je_i^-(n-1)\right] \tag{6.39}$$

In particular, note that Eq. (6.37) corresponds to the real and imaginary parts of both sides of this equation. This relationship also demonstrates that the complex signals are *normalized* in the sense that their magnitudes do not change from one section to the next,

$$\left|e_i^+(n) + je_{i+1}^-(n)\right| = \left|e_{i+1}^+(n) + je_i^-(n-1)\right| \tag{6.40}$$

Thus, this structure is referred to as the normalized lattice. One of the advantages of the normalized lattice filter is its robustness to round-off errors when using fixed-point arithmetic [10].

The last structure is the one-multiplier lattice which, as its name implies, requires only one real multiplication per module [5]. This structure may be derived by re-writing the Kelly-Lochbaum equations as follows,

$$e_j^+(n) = \Gamma_{j+1}\left[e_{j+1}^+(n) - e_j^-(n-1)\right] + e_{j+1}^+(n) \tag{6.41}$$

$$e_{j+1}^-(n) = \Gamma_{j+1}\left[e_{j+1}^+(n) - e_j^-(n-1)\right] + e_j^-(n-1) \tag{6.42}$$

Since the reflection coefficient Γ_{j+1} multiplies the same signal in both equations, forming the sum $e_{j+1}^+(n) - e_j^-(n-1)$ prior to multiplying by Γ_{j+1} leads to a structure requiring only one multiplication per stage. The one-multiplier lattice structure is shown in Fig. 6.12*c*. In addition to one multiply, this structure requires three additions and one delay.

6.4.3 Lattice Filters having Poles and Zeros

In the previous section, several different all-pole lattice structures were derived from the basic all-pole lattice given in Fig. 6.7. From this structure we may also develop a lattice filter that realizes a general rational system function of the form

$$H(z) = \frac{B_q(z)}{A_p(z)} = \frac{b_q(0) + b_q(1)z^{-1} + \cdots + b_q(q)z^{-q}}{1 + a_p(1)z^{-1} + \cdots + a_p(p)z^{-p}} \tag{6.43}$$

[3]Note that we have changed the subscript indices on the forward and backward prediction errors from j to i in order to avoid confusion with the imaginary number $j = \sqrt{-1}$.

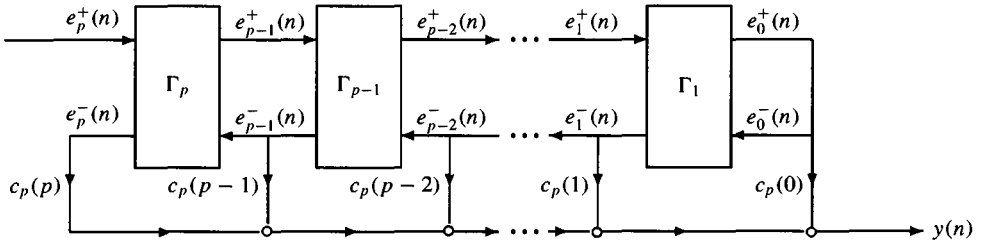


Figure 6.13 *A lattice filter having p poles and p zeros.*

where $q \leq p$. To see how this may be done, note that $H(z)$ may be implemented as a cascade of an all-pole filter $1/A_p(z)$ with an all-zero filter $B_q(z)$. The pair of difference equations corresponding to this cascade are

$$w(n) = x(n) - \sum_{k=1}^{p} a_p(k)w(n-k)$$

where $w(n)$ is the output of the all-pole filter $1/A_p(z)$ and

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

is the output of the FIR filter $B_q(z)$. Since the zeros of $H(z)$ are introduced by taking a linear combination of delayed outputs of the all-pole filter, with the all-pole lattice filter in Fig. 6.7, a rational system function may similarly be realized with a linear combination of the signals $e_0^+(n-k)$ for $k = 0, 1, \ldots, q$. However, a more efficient way is to take a linear combination of the backward prediction errors $e_j^-(n)$

$$y(n) = \sum_{j=0}^{q} c_q(j)e_j^-(n) \tag{6.44}$$

as shown in Fig. 6.13 (in this figure we assume that $q = p$). To show that this filter does, in fact, have a rational system function, we begin by expressing Eq. (6.44) in the z-domain as follows

$$Y(z) = \sum_{j=0}^{q} c_q(j)E_j^-(z) \tag{6.45}$$

With

$$E_j^-(z) = A_j^R(z)E_0^+(z) = \frac{A_j^R(z)}{A_p(z)}E_p^+(z)$$

it follows that

$$Y(z) = \sum_{j=0}^{q} c_q(j)\frac{A_j^R(z)}{A_p(z)}E_p^+(z)$$

Therefore, the system function relating the input $E_p^+(z)$ to the output $Y(z)$ is

$$H(z) = \frac{\sum_{j=0}^{q} c_q(j)A_j^R(z)}{A_p(z)} \tag{6.46}$$

which has p poles and q zeros. Note that the zeros of $H(z)$ are the roots of the polynomial

$$B_q(z) = \sum_{j=0}^{q} c_q(j) A_j^R(z) \tag{6.47}$$

which is a function of not only the tap weights $c_q(j)$ but also the reflection coefficients Γ_j.

To see how the coefficients $b_q(j)$ are related to the coefficients $c_q(j)$, we substitute the following expression for $A_j^R(z)$,

$$A_j^R(z) = z^{-j} A_j^*(1/z^*) = \sum_{m=0}^{j} a_j^*(m) z^{m-j}$$

into Eq. (6.47) as follows

$$B_q(z) = \sum_{j=0}^{q} c_q(j) \sum_{m=0}^{j} a_j^*(m) z^{m-j} \tag{6.48}$$

With the substitution $k = j - m$, Eq. (6.48) becomes

$$B_q(z) = \sum_{j=0}^{q} \sum_{k=0}^{j} c_q(j) a_j^*(j-k) z^{-k}$$

Interchanging the order of the summations and making the appropriate changes in the indexing we have

$$B_q(z) = \sum_{k=0}^{q} \left[\sum_{j=k}^{q} c_q(j) a_j^*(j-k) \right] z^{-k} = \sum_{k=0}^{q} b_q(k) z^{-k} \tag{6.49}$$

Finally, equating powers of z on both sides of Eq. (6.50) gives the desired expression

$$\boxed{b_q(k) = \sum_{j=k}^{q} c_q(j) a_j^*(j-k)} \tag{6.50}$$

This equation shows how the coefficients $b_q(k)$ may be found from the coefficients $c_q(k)$ and Γ_j. Specifically, from the reflection coefficients Γ_j of the all-pole filter we may use the step-up recursion to find the all-pole coefficients for model orders $j = 0, 1, \ldots, q$. With $a_j(k)$ along with the coefficients $c_q(k)$, Eq. (6.50) may then be used to determine the coefficients $b_q(k)$. Equation (6.50) may also be used to find the coefficients $c_q(k)$ required to implement a given rational system function with a numerator $B_q(z)$. To do this, Eq. (6.50) is rewritten as follows

$$b_q(k) = c_q(k) + \sum_{j=k+1}^{q} c_q(j) a_j^*(j-k) \tag{6.51}$$

(here we have used the fact that $a_j(0) = 1$). Thus, the coefficients $c_q(k)$ may be computed recursively as follows

$$\boxed{c_q(k) = b_q(k) - \sum_{j=k+1}^{q} c_q(j) a_j^*(j-k)} \tag{6.52}$$

for $k = q-1, q-2, \ldots, 0$. This recursion is initialized by setting

$$c_q(q) = b_q(q) \tag{6.53}$$

The following example illustrates the procedure.

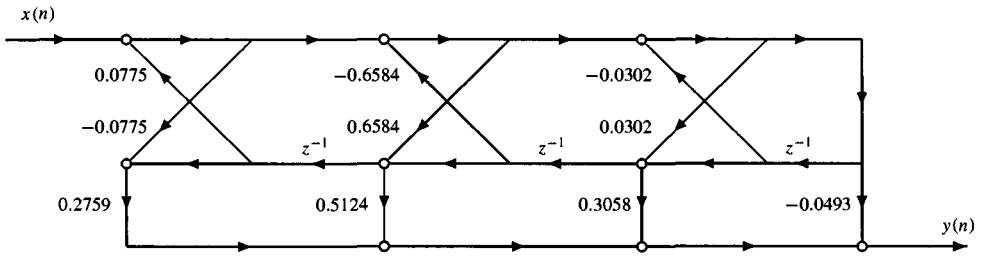


Figure 6.14 *Lattice filter realization of a 4th-order low-pass elliptic filter.*

Example 6.4.2 ***Lattice Filter Implementation of a Low-Pass Elliptic Filter***

A third-order low-pass elliptic filter with a cutoff frequency of $\omega = 0.5\pi$ has a system function given by

$$H(z) = \frac{0.2759 + 0.5121z^{-1} + 0.5121z^{-2} + 0.2759z^{-3}}{1 - 0.0010z^{-1} + 0.6546z^{-2} - 0.0775z^{-3}}$$

To implement this filter using a lattice filter structure, we first transform the denominator coefficients into reflection coefficients. Using the step-down recursion we find

$$\boldsymbol{\Gamma} = [0.0302,\ 0.6584,\ -0.0775]^T$$

with the second-order coefficients given by

$$\mathbf{a}_2 = [1,\ 0.0501,\ 0.6584]^T$$

and the first-order coefficients equal to

$$\mathbf{a}_1 = [1,\ 0.0302]^T$$

Next, we find the coefficients $c_3(k)$ that correspond to the given numerator using the recursion in Eq.(6.52). Beginning with

$$c_3(3) = b_3(3) = 0.2759$$

we then find

$$\begin{aligned} c_3(2) &= b_3(2) - c_3(3)a_3(1) = 0.5124 \\ c_3(1) &= b_3(1) - c_3(2)a_2(1) - c_3(3)a_3(2) = 0.3058 \\ c_3(0) &= b_3(0) - c_3(1)a_1(1) - c_3(2)a_2(2) - c_3(3)a_3(3) = -0.0493 \end{aligned}$$

This leads to the lattice filter implementation illustrated in Fig. 6.14.

6.5 LATTICE METHODS FOR ALL-POLE SIGNAL MODELING

In Chapter 4 we developed several different approaches for finding an all-pole model for a signal $x(n)$. In each case, we first defined an error that was to be minimized and then solved the minimization problem for the optimum set of transversal filter coefficients $a_p(k)$. Since the lattice filter provides an alternate parameterization of the all-pole filter, i.e., in terms of its reflection coefficients, we may also consider formulating the all-pole signal modeling

problem as one of finding the reflection coefficients that minimize some error. In this section we look at several such *lattice methods* for signal modeling including the *forward covariance method*, the *backward covariance method*, *Burg's method*, and the *modified covariance method*. Except for the modified covariance method, each of these methods is a *sequential* optimization procedure. That is to say, given an error that is to be minimized, such as the sum of the squares of the forward prediction error, the first reflection coefficient, Γ_1, is found that minimizes this error. Then, with Γ_1 fixed, the optimum value of Γ_2 is determined. Thus, in general, with $\Gamma_1, \ldots, \Gamma_{j-1}$ held constant, the value of Γ_j is found that minimizes some error. We begin, in the following section, with the forward covariance method.

6.5.1 The Forward Covariance Method

Let $x(n)$ for $n = 0, 1, \ldots, N$ be a signal, either real or complex, that is to be modeled as the unit sample response of an all-pole filter of order p. For an all-pole lattice filter, it is necessary to determine the reflection coefficients which, for reasons soon to become apparent, we will denote by Γ_j^+. In the *forward covariance method* the reflection coefficients are computed *sequentially* with the jth reflection coefficient, Γ_j^+, determined by minimizing the *covariance-type* error

$$\mathcal{E}_j^+ = \sum_{n=j}^{N} \left|e_j^+(n)\right|^2 \tag{6.54}$$

Note that since $\mathcal{E}_j^+$ is a function only of the signal values $x(n)$ over the interval $[0, N]$, as with the covariance method described in Section 4.6.2, it is not necessary to window the signal or make any assumptions about the values of $x(n)$ outside the given interval.

To begin the sequential minimization of $\mathcal{E}_j^+$ using the forward covariance method, we must first find the reflection coefficient Γ_1^+ that minimizes the first-order error

$$\mathcal{E}_1^+ = \sum_{n=1}^{N} \left|e_1^+(n)\right|^2 \tag{6.55}$$

This minimization may be performed by setting the derivative of $\mathcal{E}_1^+$ with respect to $(\Gamma_1^+)^*$ equal to zero as follows

$$\frac{\partial}{\partial(\Gamma_1^+)^*}\mathcal{E}_1^+ = \sum_{n=1}^{N} e_1^+(n)\frac{\partial}{\partial(\Gamma_1^+)^*}\left[e_1^+(n)\right]^* = 0 \tag{6.56}$$

Since

$$e_1^+(n) = e_0^+(n) + \Gamma_1^+ e_0^-(n-1) \tag{6.57}$$

then

$$\frac{\partial\left[e_1^+(n)\right]^*}{\partial(\Gamma_1^+)^*} = \left[e_0^-(n-1)\right]^*$$

and Eq. (6.56) becomes

$$\sum_{n=1}^{N} e_1^+(n)\left[e_0^-(n-1)\right]^* = 0 \tag{6.58}$$

Note that if we consider $e_1^+(n)$ and $e_0^-(n-1)$ for $n = 1, 2, \ldots, N$ to be N-dimensional vectors,

$$\begin{aligned} \mathbf{e}_1^+ &= \left[e_1^+(1),\ e_1^+(2), \cdots,\ e_1^+(N)\right]^T \\ \mathbf{e}_0^- &= \left[e_0^-(0),\ e_0^-(1), \cdots,\ e_0^-(N-1)\right]^T \end{aligned} \tag{6.59}$$

then Eq. (6.58) states that $\mathcal{E}_1^+$ will be minimized when $\mathbf{e}_1^+$ and $\mathbf{e}_0^-$ are *orthogonal*,

$$\langle \mathbf{e}_1^+,\ \mathbf{e}_0^- \rangle = 0$$

Substituting Eq. (6.57) into Eq. (6.58) and solving for Γ_1^+ leads to the following expression for Γ_1^+,

$$\Gamma_1^+ = -\frac{\displaystyle\sum_{n=1}^{N} e_0^+(n)\left[e_0^-(n-1)\right]^*}{\displaystyle\sum_{n=1}^{N} \left|e_0^-(n-1)\right|^2} \tag{6.60}$$

which, in terms of the vectors $\mathbf{e}_0^+$ and $\mathbf{e}_0^-$, may be expressed as

$$\Gamma_1^+ = -\frac{\langle \mathbf{e}_0^+,\ \mathbf{e}_0^- \rangle}{\|\mathbf{e}_0^-\|^2}$$

Continuing with this process, sequentially minimizing $\mathcal{E}_j^+$ while holding $\Gamma_1^+, \ldots, \Gamma_{j-1}^+$ fixed, the value of Γ_j^+ is found by setting the derivative of $\mathcal{E}_j^+$ with respect to $(\Gamma_j^+)^*$ equal to zero,

$$\frac{\partial}{\partial(\Gamma_j^+)^*}\mathcal{E}_j^+ = \sum_{n=j}^{N} e_j^+(n)\frac{\partial}{\partial(\Gamma_j^+)^*}\left[e_j^+(n)\right]^* = 0 \tag{6.61}$$

Using the forward prediction error update equation

$$e_j^+(n) = e_{j-1}^+(n) + \Gamma_j^+ e_{j-1}^-(n-1) \tag{6.62}$$

it follows that the partial derivative of $\left[e_j^+(n)\right]^*$ with respect to $(\Gamma_j^+)^*$ is $\left[e_{j-1}^-(n-1)\right]^*$. Therefore, Eq. (6.61) becomes

$$\boxed{\sum_{n=j}^{N} e_j^+(n)\left[e_{j-1}^-(n-1)\right]^* = 0} \tag{6.63}$$

Equation (6.63) states that the vectors $\mathbf{e}_j^+$ and $\mathbf{e}_{j-1}^-$ are orthogonal,

$$\boxed{\langle \mathbf{e}_j^+,\ \mathbf{e}_{j-1}^- \rangle = 0} \tag{6.64}$$

where

$$\begin{aligned} \mathbf{e}_j^+ &= \left[e_j^+(j),\ e_j^+(j+1), \ldots,\ e_j^+(N)\right]^T \\ \mathbf{e}_{j-1}^- &= \left[e_{j-1}^-(j-1),\ e_{j-1}^-(j), \ldots,\ e_{j-1}^-(N-1)\right]^T \end{aligned} \tag{6.65}$$

are vectors of length $N - j + 1$. Substituting Eq. (6.62) for $e_j^+(n)$ into Eq. (6.63) and solving for Γ_j^+ we find that the jth reflection coefficient is

The Forward Covariance Method

```
function [gamma,err] = fcov(x,p)
%
x = x(:);
N=length(x);
eplus  = x(2:N);
eminus = x(1:N-1);
N=N-1;
for j=1:p;
    gamma(j) = -eminus'*eplus/(eminus'*eminus);
    temp1    =  eplus  + gamma(j)*eminus;
    temp2    =  eminus + conj(gamma(j))*eplus;
    err(j)   =  temp1'*temp1;
    eplus    =  temp1(2:N);
    eminus   =  temp2(1:N-1);
    N=N-1;
    end;
```

Figure 6.15 *A* MATLAB *program for finding the reflection coefficients for a pth-order all-pole model of a signal $x(n)$ using the forward covariance method.*

$$\Gamma_j^+ = -\frac{\sum_{n=j}^{N} e_{j-1}^+(n)\left[e_{j-1}^-(n-1)\right]^*}{\sum_{n=j}^{N}\left|e_{j-1}^-(n-1)\right|^2} = -\frac{\langle \mathbf{e}_{j-1}^+, \mathbf{e}_{j-1}^- \rangle}{\|\mathbf{e}_{j-1}^-\|^2} \tag{6.66}$$

From a computational point of view, the forward covariance algorithm works as follows. Given the first $j-1$ reflection coefficients, $\Gamma_{j-1}^+ = \left[\Gamma_1^+, \Gamma_2^+, \ldots, \Gamma_{j-1}^+\right]^T$, and given the forward and backward prediction errors $e_{j-1}^+(n)$ and $e_{j-1}^-(n)$, the jth reflection coefficient is found by evaluating Eq. (6.66). Then, using the lattice filter, the $(j-1)$st-order forward and backward prediction errors are updated to form the jth-order errors $e_j^+(n)$ and $e_j^-(n)$ and the process is repeated. A MATLAB program for finding a pth-order all-pole model for a complex signal $x(n)$ using the forward covariance method is given in Fig. 6.15.

Example 6.5.1 *Forward Covariance Method*

Given the signal $x(n) = \beta^n u(n)$ for $n = 0, \ldots, N$, let us find the pth-order all-pole model for $x(n)$ using the forward covariance method. We begin by initializing the forward and backward prediction errors as follows

$$e_0^+(n) = e_0^-(n) = x(n) = \beta^n \quad ; \quad n = 0, 1, \ldots, N$$

Next, we evaluate the norm of $e_0^-(n-1)$,

$$\|\mathbf{e}_0^-\|^2 = \sum_{n=1}^{N}\left[e_0^-(n-1)\right]^2 = \sum_{n=0}^{N-1}\beta^{2n} = \frac{1-\beta^{2N}}{1-\beta^2}$$

and the inner product between $e_0^+(n)$ and $e_0^-(n-1)$,

$$\langle \mathbf{e}_0^+, \mathbf{e}_0^- \rangle = \sum_{n=1}^{N} e_0^+(n) e_0^-(n-1) = \beta \sum_{n=0}^{N-1} \beta^{2n} = \beta \frac{1-\beta^{2N}}{1-\beta^2}$$

Then, using Eq. (6.60), we find for the first reflection coefficient,

$$\Gamma_1^+ = -\frac{\langle \mathbf{e}_0^+, \mathbf{e}_0^- \rangle}{\|\mathbf{e}_0^-\|^2} = -\beta$$

Updating the forward prediction error using Eq. (6.62) we have

$$e_1^+(n) = e_0^+(n) + \Gamma_1^+ e_0^-(n-1) = \beta^n u(n) - \beta(\beta)^{n-1} u(n-1) = \delta(n)$$

Therefore, the first-order modeling error is zero

$$\mathcal{E}_1^+ = \sum_{n=1}^{N} [e_1^+(n)]^2 = 0$$

The first-order backward prediction error, on the other hand, is not equal to zero. In fact, using Eq. (6.16) we see that

$$e_1^-(n) = e_0^-(n-1) + \Gamma_1^+ e_0^+(n) = \beta^{n-1} u(n-1) - \beta^{n+1} u(n)$$

For the second reflection coefficient, since $e_1^+(n) = 0$ for $n > 0$ then

$$\Gamma_2^+ = -\frac{\langle \mathbf{e}_1^+, \mathbf{e}_1^- \rangle}{\|\mathbf{e}_1^-\|^2} = 0$$

With the second reflection coefficient equal to zero it follows that $e_2^+(n) = e_1^+(n) = \delta(n)$ and that $\Gamma_3^+ = 0$. Continuing, we see that $\Gamma_j^+ = 0$ for all $j > 1$,

$$\boldsymbol{\Gamma}^+ = [-\beta, \ 0, \ 0, \ldots]^T$$

Lest we be misled by the results of the first example, as another example let us consider the signal $x(n)$ shown in Fig. 6.16*a* which is the unit sample response of the third-order filter

$$H(z) = \frac{1}{A_3(z)} = \frac{1}{1 - 0.12z^{-1} - 0.456z^{-2} + 0.6z^{-3}}$$

The lattice filter coefficients for this filter are

$$\boldsymbol{\Gamma} = [0.6, \ -0.6, \ 0.6]^T$$

Using the MATLAB program for the forward covariance method given in Fig. 6.15 with $p = 3$ and $N = 60$ we find that the reflection coefficients are

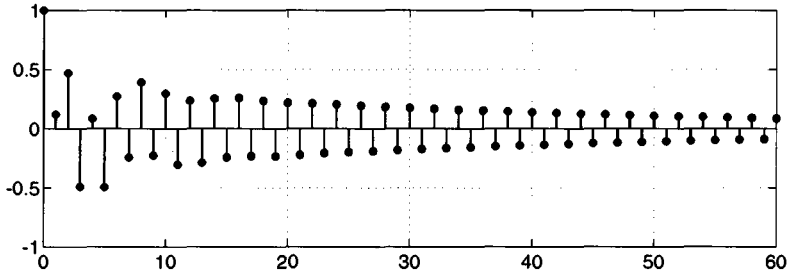
$$\boldsymbol{\Gamma}^+ = [0.5836, \ -0.4962, \ 0.5603]^T$$

and the sequence of squared errors is

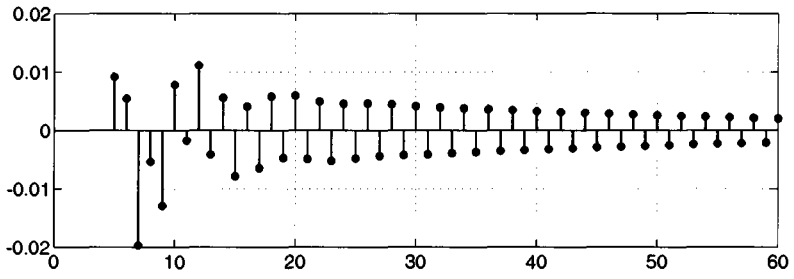
$$\underline{\mathcal{E}}^+ = [1.4179, \ 0.4128, \ 0.0117]^T$$

The system function corresponding to these reflection coefficients is

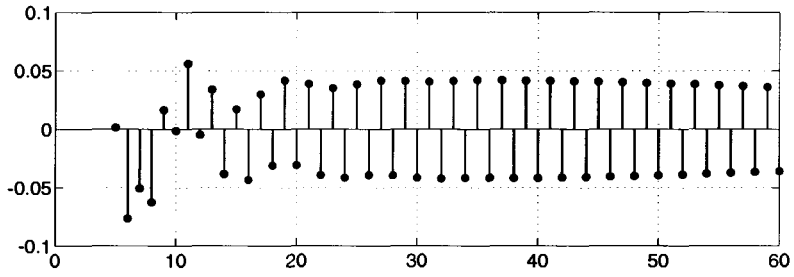
$$\hat{H}_3(z) = \frac{1}{\hat{A}_3(z)} = \frac{1}{1 + 0.0160z^{-1} - 0.3315z^{-2} + 0.5603z^{-3}}$$



(a) A third-order all-pole signal that is to modeled.



(b) The forward covariance error, $e_5^+(n)$, using a model order of $p = 5$.



(c) The model error, $e(n) = x(n) - \widehat{x}(n)$.

Figure 6.16 *Using the forward covariance method to model a signal that is the unit sample response of a third-order all-pole filter.*

Note that although $x(n)$ is the unit sample response of an all-pole filter, the forward covariance method is unable to correctly model $x(n)$. If we increase the model order to $p = 5$, the next two reflection coefficients are

$$\Gamma_4^+ = 0.0483 \quad ; \quad \Gamma_5^+ = 0.1067$$

and the corresponding modeling errors are

$$\mathcal{E}_4^+ = 0.0087 \quad ; \quad \mathcal{E}_5^+ = 0.0016$$

Thus, extra poles may be used to further decrease the forward prediction error. Shown in Fig. 6.16*b* is the forward covariance error $e_5^+(n)$ over the interval $[5, N]$, the interval over

which $e_5^+(n)$ is minimized. Finally, in Fig. 6.16*c* is the error, $e(n) = x(n) - \hat{x}(n)$. If we compute the squared model error we find

$$\mathcal{E}_{LS} = \sum_{n=5}^{N} [x(n) - \hat{x}(n)]^2 = 0.0889$$

The fact that $\mathcal{E}_{LS}$ and $\mathcal{E}_5^+$ are *different* should not be surprising since they are defined differently. It is important to keep in mind that in the forward covariance method $\mathcal{E}_p^+$ is the sum of the squares of the forward prediction error,

$$e_j^+(n) = a_j(n) * x(n)$$

whereas $\mathcal{E}_{LS}$ is the sum of the squares of the modeling error

$$e(n) = x(n) - \hat{x}(n)$$

Finally, note that since $x(n)$ is the output of a third-order all-pole filter, if we were to use the covariance method then $x(n)$ would be modeled exactly and the third-order modeling error $\mathcal{E}_3^C$ would be zero.

As illustrated in the previous example, the model that is produced with the forward covariance method is, in general, different from that obtained using the covariance method. Therefore, it is not necessarily true that the forward covariance method will produce an exact model for a signal $x(n)$, even if it is the output of an all-pole filter. In addition, as we observed when modeling the signal $x(n) = \beta^n$, the forward covariance method does not guarantee a stable model.

6.5.2 The Backward Covariance Method

As we saw in the previous section, using the forward covariance method the reflection coefficients of the lattice filter are found by sequentially minimizing the sum of the squares of the forward prediction error. Since a lattice filter generates both forward and backward prediction errors, we may just as easily minimize the sum of the squares of the backward prediction error. These two approaches are not entirely independent of each other, however, since minimizing the backward prediction error is the same as *time reversing* $x(n)$ and minimizing the forward prediction error. One may argue, in fact, that since $x(n)$ and $x^*(N-n)$ are statistically equivalent in the sense that they have identical deterministic autocorrelation sequences, there is no inherent reason why one error should be preferred over the other. Therefore, in this section we will look at the *backward covariance method* which performs a sequential minimization of the sum of the squares of the backward prediction error,

$$\mathcal{E}_j^- = \sum_{n=j}^{N} |e_j^-(n)|^2 \tag{6.67}$$

In order to distinguish this solution from that obtained using the forward covariance algorithm, we will denote the reflection coefficients in the backward covariance method by Γ_j^-. Proceeding as we did for the forward covariance method, setting the derivative of $\mathcal{E}_j^-$ with respect to $(\Gamma_j^-)^*$ equal to zero, we find that the reflection coefficient Γ_j^- that minimizes the backward prediction error $\mathcal{E}_j^-$ is given by

$$\Gamma_j^- = -\frac{\sum_{n=j}^{N} e_{j-1}^+(n)\left[e_{j-1}^-(n-1)\right]^*}{\sum_{n=j}^{N}\left|e_{j-1}^+(n)\right|^2} = -\frac{\langle \mathbf{e}_{j-1}^+,\ \mathbf{e}_{j-1}^-\rangle}{\|\mathbf{e}_{j-1}^+\|^2} \tag{6.68}$$

From a computational point of view, the backward covariance method works in the same way as the forward covariance method. Given the first $j-1$ reflection coefficients, $\boldsymbol{\Gamma}^- = \left[\Gamma_1^-,\ \Gamma_2^-,\ \ldots,\ \Gamma_{j-1}^-\right]^T$, and given the forward and backward prediction errors $e_{j-1}^+(n)$ and $e_{j-1}^-(n)$, the jth reflection coefficient that minimizes $\mathcal{E}_j^-$ is computed using Eq. (6.68). Then, using the lattice filter the $(j-1)$st-order forward and backward prediction errors are updated to form the jth-order errors $e_j^+(n)$ and $e_j^-(n)$ and the process is repeated. A MATLAB program for the backward covariance method is easily derived by modifying `fcov.m` in Fig. 6.15. Changes are required only in the lines that compute the reflection coefficient Γ_j^- and the error $\mathcal{E}_j^-$.

Example 6.5.2 *Backward Covariance Method*

Let us look again at the modeling problem considered in Example 6.5.1, this time using the backward covariance method. As before, we begin by initializing the forward and backward errors

$$e_0^+(n) = e_0^-(n) = \beta^n u(n) \quad ; \quad n = 0, 1, \ldots, N$$

Evaluating the norm

$$\|\mathbf{e}_0^+\|^2 = \sum_{n=1}^{N}\left[e_0^+(n)\right]^2 = \sum_{n=1}^{N}\beta^{2n} = \beta^2\frac{1-\beta^{2N}}{1-\beta^2}$$

and the inner product

$$\langle \mathbf{e}_0^+,\ \mathbf{e}_0^-\rangle = \sum_{n=1}^{N} e_0^+(n)e_0^-(n-1) = \beta\frac{1-\beta^{2N}}{1-\beta^2}$$

we find that the first reflection coefficient is

$$\Gamma_1^- = -\frac{1}{\beta}$$

Updating the backward prediction error

$$e_1^-(n) = e_0^-(n-1) + \Gamma_1^- e_0^+(n) = \beta^{n-1}u(n-1) - \frac{1}{\beta}\beta^n u(n) = -\frac{1}{\beta}\delta(n)$$

it follows that the first-order modeling error is zero

$$\mathcal{E}_1^- = \sum_{n=1}^{N}\left[e_1^-(n)\right]^2 = 0$$

The first-order forward prediction error, however, is nonzero

$$e_1^+(n) = e_0^+(n) + \Gamma_1^- e_0^-(n-1) = \beta^n u(n) - \beta^{-1}\beta^{n-1}u(n-1)$$

Nevertheless, since $e_1^-(n) = 0$ for $n > 0$ then $\langle \mathbf{e}_1^-, \mathbf{e}_1^+ \rangle = 0$ and the second reflection coefficient is equal to zero, $\Gamma_2^- = 0$. In fact, as with the forward covariance method, all succeeding reflection coefficients will also be equal to zero,

$$\boldsymbol{\Gamma}^- = [-1/\beta,\ 0,\ 0, \ldots]^T$$

As another example, let us again consider the signal $x(n)$ from Example 6.5.1 that is the unit sample response of the third-order filter

$$H(z) = \frac{1}{1 - 0.12z^{-1} - 0.456z^{-2} + 0.6z^{-3}}$$

which has lattice filter coefficients

$$\boldsymbol{\Gamma} = [0.6,\ -0.6,\ 0.6]$$

Using the backward covariance method with $p = 3$ and $N = 60$, the reflection coefficient sequence is

$$\boldsymbol{\Gamma}^- = [0.8011,\ -1.1787,\ 0.6796]^T$$

which corresponds to a filter having a system function

$$\hat{H}_3(z) = \frac{1}{1 - 0.9441z^{-1} - 1.2760z^{-2} + 0.6796z^{-3}}$$

Furthermore, the sequence of squared errors is

$$\underline{\mathcal{E}}^- = [3.6556,\ 1.9460,\ 0.9147]^T$$

Note that, unlike the solution to the forward covariance method, the model is unstable. If we increase the model order to $p = 5$, the next two reflection coefficients are

$$\Gamma_4^- = -1.1255 \quad ; \quad \Gamma_5^- = -0.4135$$

and the corresponding errors are

$$\mathcal{E}_4^- = 0.5189 \quad ; \quad \mathcal{E}_5^- = 0.1100$$

As illustrated in the previous example, the reflection coefficients in the backward covariance method are, in general, different from those obtained using the forward covariance method. As with the forward covariance method, it is not necessarily true that the backward covariance method will produce an exact model for a signal $x(n)$, even if it is the unit sample response of an all-pole filter. In addition, the backward covariance method may lead to an unstable model.

6.5.3 Variations

One of the properties that we have discovered about the forward and the backward covariance methods is that the reflection coefficients are not guaranteed to be less than one in magnitude. As a result, it is possible for either of these methods to produce an unstable model. Since ensuring a stable model may be necessary in applications such as speech synthesis or signal extrapolation, there is an interest in modeling techniques that are guaranteed to produce a stable model. There are several ways that we may *combine* the models generated by the

forward and backward covariance methods to produce a stable model. One such approach proposed by Itakura [5] is to model a signal using reflection coefficients that are given by

$$\Gamma_j^I = -\frac{\sum_{n=j}^{N} e_{j-1}^+(n)\left[e_{j-1}^-(n-1)\right]^*}{\sqrt{\sum_{n=j}^{N}\left|e_{j-1}^+(n)\right|^2 \sum_{n=j}^{N}\left|e_{j-1}^-(n-1)\right|^2}} = -\frac{\langle \mathbf{e}_{j-1}^+,\ \mathbf{e}_{j-1}^- \rangle}{\|\mathbf{e}_{j-1}^+\|\ \|\mathbf{e}_{j-1}^-\|} \tag{6.69}$$

If we compare Γ_j^I with the reflection coefficients used in the forward and backward covariance methods, we see that $|\Gamma_j^I|$ is the *geometric mean* of $|\Gamma_j^+|$ and $|\Gamma_j^-|$,

$$|\Gamma_j^I| = \sqrt{|\Gamma_j^+|\ |\Gamma_j^-|} \tag{6.70}$$

We may show that $|\Gamma_j^I| \leq 1$ by applying the *Cauchy-Schwartz inequality* to the vectors $\mathbf{e}_{j-1}^-$ and $\mathbf{e}_{j-1}^+$ as follows,

$$|\langle \mathbf{e}_{j-1}^+,\ \mathbf{e}_{j-1}^- \rangle| \leq \|\mathbf{e}_{j-1}^+\|\ \|\mathbf{e}_{j-1}^-\| \tag{6.71}$$

where equality holds if and only if $\mathbf{e}_{j-1}^+ = \alpha\ \mathbf{e}_{j-1}^-$ for some constant α. Therefore, it follows that $|\Gamma_j^I| \leq 1$.

There are other ways to ensure that the reflection coefficients are bounded by one in magnitude [8]. For example, it follows from the properties of the geometric mean that

$$\min\left\{|\Gamma_j^+|,\ |\Gamma_j^-|\right\} \leq |\Gamma_j^I| \leq \max\left\{|\Gamma_j^+|,\ |\Gamma_j^-|\right\}$$

Since $|\Gamma_j^I| \leq 1$, it follows that if $|\Gamma_j^+|$ is greater than one in magnitude then $|\Gamma_j^-|$ will be less than one in magnitude, and vice versa. Therefore, if we set

$$|\Gamma_j^{\min}| = \min\left\{|\Gamma_j^+|,\ |\Gamma_j^-|\right\}$$

then $|\Gamma_j^{\min}|$ is guaranteed to be bounded by one in magnitude.

Although both the geometric mean and the minimum method produce reflection coefficients that are bounded by one in magnitude and thus represent a stable model, neither approach corresponds to a solution that results from the minimization of an error. In the next section we look at Burg's method which uses a sequential approach to find the reflection coefficients and does so in a way that guarantees that the reflection coefficients are bounded by one in magnitude and, therefore, guarantees that the model is stable.

6.5.4 Burg's Method

In Chapter 4 we saw that the covariance method for all-pole signal modeling, while more accurate than the autocorrelation method since it does not apply a window to the data, has the property that it does not always lead to a stable model. In addition, as we have seen in the previous two sections, sequentially minimizing either the forward or the backward prediction error using a "covariance-like" error may also lead to an unstable model. In the 1960s, Burg developed a method for spectrum estimation known as the maximum entropy method [1]. As a part of this method, which involves finding an all-pole model for the data, he proposed that the reflection coefficients be computed sequentially by minimizing the *sum* of the forward and backward prediction errors [2],

$$\mathcal{E}_j^B = \mathcal{E}_j^+ + \mathcal{E}_j^- = \sum_{n=j}^{N} |e_j^+(n)|^2 + \sum_{n=j}^{N} |e_j^-(n)|^2 \tag{6.72}$$

This *blending* of the forward and backward prediction errors places an equal emphasis on $\mathcal{E}_j^+$ and $\mathcal{E}_j^-$, and minimizing this error may be justified based on the statistical equivalence of $x(n)$ and $x^*(N-n)$ as pointed out in Section 6.5.2. As we will soon see, what makes the Burg error interesting and attractive is the fact that sequentially minimizing $\mathcal{E}_j^B$ guarantees that the reflection coefficients will be bounded by one in magnitude and thus, the model will be stable.

As we have done many times before, we may find the value of the reflection coefficient, Γ_j^B, that minimizes $\mathcal{E}_j^B$ by setting the derivative of $\mathcal{E}_j^B$ with respect to $(\Gamma_j^B)^*$ equal to zero as follows

$$\frac{\partial}{\partial(\Gamma_j^B)^*}\mathcal{E}_j^B = \frac{\partial}{\partial(\Gamma_j^B)^*}\sum_{n=j}^{N}\left\{|e_j^+(n)|^2 + |e_j^-(n)|^2\right\}$$

$$= \sum_{n=j}^{N}\left\{e_j^+(n)[e_{j-1}^-(n-1)]^* + [e_j^-(n)]^* e_{j-1}^+(n)\right\} = 0 \tag{6.73}$$

Substituting the error update equations for $e_j^+(n)$ and $[e_j^-(n)]^*$, which are similar to those given for $e_{j+1}^+(n)$ and $e_{j+1}^-(n)$ in Eqs. (6.11) and (6.16), and solving for Γ_j^B we find that the value of Γ_j^B that minimizes $\mathcal{E}_j^B$ is

$$\boxed{\Gamma_j^B = -\frac{2\sum_{n=j}^{N} e_{j-1}^+(n)[e_{j-1}^-(n-1)]^*}{\sum_{n=j}^{N}\left\{|e_{j-1}^+(n)|^2 + |e_{j-1}^-(n-1)|^2\right\}} = -\frac{2\langle \mathbf{e}_{j-1}^+, \mathbf{e}_{j-1}^-\rangle}{\|\mathbf{e}_{j-1}^+\|^2 + \|\mathbf{e}_{j-1}^-\|^2}} \tag{6.74}$$

To show that $|\Gamma_j^B| \leq 1$ we use the inequality given in Eq. (2.26) on p. 24 which states that, for any two vectors $\mathbf{a}$ and $\mathbf{b}$,

$$2|\langle \mathbf{a}, \mathbf{b}\rangle| \leq \|\mathbf{a}\|^2 + \|\mathbf{b}\|^2 \tag{6.75}$$

with equality holding if and only if $\mathbf{a} = \pm\mathbf{b}$. With $\mathbf{a} = \mathbf{e}_{j-1}^-$ and $\mathbf{b} = \mathbf{e}_{j-1}^+$ it follows immediately that $|\Gamma_j^B| \leq 1$ with equality if and only if $e_{j-1}^+(n) = e_{j-1}^-(n-1)$.

In the process of deriving the Levinson-Durbin recursion we saw that the modeling errors ϵ_j and ϵ_{j-1} for the autocorrelation method are related by

$$\epsilon_j = \epsilon_{j-1}[1 - |\Gamma_j|^2] \tag{6.76}$$

A similar relationship exists between the Burg errors $\mathcal{E}_j^B$ and $\mathcal{E}_{j-1}^B$. This relationship may be derived by incorporating the lattice filter update equations into the Burg error given in Eq. (6.72) and simplifying.[4] Specifically, we have

$$\mathcal{E}_j^B = \sum_{n=j}^{N}\left\{|e_j^+(n)|^2 + |e_j^-(n)|^2\right\}$$

$$= \sum_{n=j}^{N}\left|e_{j-1}^+(n) + \Gamma_j^B e_{j-1}^-(n-1)\right|^2 + \sum_{n=j}^{N}\left|e_{j-1}^-(n-1) + (\Gamma_j^B)^* e_{j-1}^+(n)\right|^2 \tag{6.77}$$

[4] Although not difficult, this derivation is a bit messy due to the plethora of complex conjugates and inner products.

Expanding the squares and combining together common terms we may write Eq. (6.77) as follows

$$\mathcal{E}_j^B = \left[1 + |\Gamma_j^B|^2\right] \sum_{n=j}^{N} \left\{ |e_{j-1}^+(n)|^2 + |e_{j-1}^-(n-1)|^2 \right\}$$
$$+ 2(\Gamma_j^B)^* \sum_{n=j}^{N} e_{j-1}^+(n)[e_{j-1}^-(n-1)]^* + 2\Gamma_j^B \sum_{n=j}^{N} e_{j-1}^-(n-1)[e_{j-1}^+(n)]^* \qquad (6.78)$$

Now, note from Eq. (6.74) that the inner product between $e_{j-1}^+(n)$ and $e_{j-1}^-(n-1)$ may be written as

$$2\sum_{n=j}^{N} e_{j-1}^+(n)[e_{j-1}^-(n-1)]^* = -\Gamma_j^B \sum_{n=j}^{N} \left\{ |e_{j-1}^+(n)|^2 + |e_{j-1}^-(n-1)|^2 \right\}$$

Substituting this expression and its conjugate into Eq. (6.78) and simplifying we have

$$\mathcal{E}_j^B = \left[1 - |\Gamma_j^B|^2\right] \sum_{n=j}^{N} \left\{ |e_{j-1}^+(n)|^2 + |e_{j-1}^-(n-1)|^2 \right\} \qquad (6.79)$$

Finally, since

$$\sum_{n=j}^{N} \left\{ |e_{j-1}^+(n)|^2 + |e_{j-1}^-(n-1)|^2 \right\} = \sum_{n=j-1}^{N} \left\{ |e_{j-1}^+(n)|^2 + |e_{j-1}^-(n)|^2 \right\}$$
$$- |e_{j-1}^+(j-1)|^2 - |e_{j-1}^-(N)|^2$$
$$= \mathcal{E}_{j-1}^B - |e_{j-1}^+(j-1)|^2 - |e_{j-1}^-(N)|^2 \qquad (6.80)$$

we have the following recursion for the Burg error

$$\boxed{\mathcal{E}_j^B = \left\{ \mathcal{E}_{j-1}^B - |e_{j-1}^+(j-1)|^2 - |e_{j-1}^-(N)|^2 \right\} \left[1 - |\Gamma_j^B|^2\right]} \qquad (6.81)$$

which is initialized with

$$\mathcal{E}_0^B = \sum_{n=0}^{N} \left\{ |e_0^+(n)|^2 + |e_0^-(n)|^2 \right\} = 2\sum_{n=0}^{N} |x(n)|^2 \qquad (6.82)$$

From a computational point of view, Burg's method works in the same way as both the forward and the backward covariance methods. Specifically, given the first $j-1$ reflection coefficients, $\mathbf{\Gamma}_{j-1}^B = [\Gamma_1^B, \ldots, \Gamma_{j-1}^B]^T$, and given the forward and backward prediction errors $e_{j-1}^+(n)$ and $e_{j-1}^-(n)$, the jth reflection coefficient is computed using Eq. (6.74). Then, using the lattice filter, the $(j-1)$st-order forward and backward prediction errors are updated to form the jth-order errors $e_j^+(n)$ and $e_j^-(n)$. A MATLAB program that implements Burg's method, sometimes referred to as the Burg recursion, is given in Fig. 6.17.

The primary computational requirements in the Burg recursion come from the evaluation of the inner product and the norms in Eq. (6.74) that are necessary to evaluate Γ_j^B. It is possible, however, to derive a recursion for updating the denominator of Eq. (6.74), thereby reducing the amount of computation. This recursion may be derived by noting that the denominator, which we will denote by D_j, is equal to the term on the left hand side of

The Burg Algorithm

```
function [gamma,err] = burg(x,p)
%
x=x(:);
N=length(x);
eplus  = x(2:N);
eminus = x(1:N-1);
N=N-1;
for j=1:p;
    gamma(j) = -2*eminus'*eplus/(eplus'*eplus+eminus'*eminus);
    temp1    = eplus  + gamma(j)*eminus;
    temp2    = eminus + conj(gamma(j))*eplus;
    err(j)   = temp1'*temp1+temp2'*temp2;
    eplus    = temp1(2:N);
    eminus   = temp2(1:N-1);
    N=N-1;
    end;
```

Figure 6.17 *A* MATLAB *program for finding the reflection coefficients corresponding to a pth-order model for a signal $x(n)$ using Burg's method.*

Eq. (6.80) and, therefore,

$$D_j = \mathcal{E}_{j-1}^B - |e_{j-1}^+(j-1)|^2 - |e_{j-1}^-(N)|^2 \tag{6.83}$$

This, however, is equivalent to the term in braces in Eq. (6.81). Thus, substituting Eq. (6.83) into Eq. (6.81) we have

$$\mathcal{E}_j^B = D_j\left[1 - |\Gamma_j^B|^2\right] \tag{6.84}$$

Since

$$D_{j+1} = \mathcal{E}_j^B - |e_j^+(j)|^2 - |e_j^-(N)|^2 \tag{6.85}$$

incorporating Eq. (6.84) into Eq. (6.85) we have the desired recursion,

$$\boxed{D_{j+1} = D_j\left[1 - |\Gamma_j^B|^2\right] - |e_j^+(j)|^2 - |e_j^-(N)|^2} \tag{6.86}$$

The equations for the Burg recursion are summarized in Table 6.1. From this table we may evaluate the computational requirements of the Burg algorithm. First, note that in order to initialize the recursion we need N multiplications and $N-1$ additions (assuming the multiplication by 2 can be done with a data shift). Since the evaluation of the error $\mathcal{E}_j$ at each stage of the recursion is not necessary, at the jth step of the recursion there are $3(N-j)+7$ multiplications, $3(N-j)+5$ additions, and one division. Finally, since step $2(a)$ is the only calculation that needs to be performed during the last stage of the recursion, $(j=p)$, then the total number of multiplications is

$$N + \sum_{j=1}^{p-1}\left[3(N-j)+7\right] + N - p + 1 = N(3p-1) - \frac{3}{2}p(p-1) + 6(p-1) \tag{6.87}$$

Table 6.1 *The Burg Recursion*

1. Initialize the recursion
   (a) $e_0^+(n) = e_0^-(n) = x(n)$
   (b) $D_1 = 2\sum_{n=1}^{N}\{|x(n)|^2 - |x(n-1)|^2\}$
2. For $j = 1$ to p
   a) $\Gamma_j^B = -\frac{2}{D_j}\sum_{n=j}^{N} e_{j-1}^+(n)\left[e_{j-1}^-(n-1)\right]^*$
   b) For $n = j$ to N
   $$e_j^+(n) = e_{j-1}^+(n) + \Gamma_j^B e_{j-1}^-(n-1)$$
   $$e_j^-(n) = e_{j-1}^-(n-1) + (\Gamma_j^B)^* e_{j-1}^+(n)$$
   c) $D_{j+1} = D_j\left(1 - |\Gamma_j^B|^2\right) - |e_j^+(j)|^2 - |e_j^-(N)|^2$
   d) $\mathcal{E}_j^B = D_j\left[1 - |\Gamma_j^B|^2\right]$

and a similar number of additions. Therefore, assuming that $N \gg p$, the Burg recursion requires on the order of $3Np$ multiplications and additions. The autocorrelation method, on the other hand, requires only $p^2 + 2p$ multiplications and p^2 additions to solve the autocorrelation normal equations using the Levinson-Durbin recursion. In addition to this, however, is the requirement for computing the $p + 1$ autocorrelations $r_x(k)$ for $k = 0, 1, \ldots, p$. For a sequence of length N this requires approximately an additional $N(p + 1)$ multiplications and additions. Both approaches, therefore, require on the order of Np multiplies and adds.

Example 6.5.3 ***The Burg Recursion***

Let us compute the reflection coefficients for the exponential signal

$$x(n) = \beta^n u(n) \quad ; \quad n = 0, 1, \ldots, N$$

using Burg's method. Using the norms and inner products derived in Examples 6.5.1 and 6.5.2 we find for the first-order Burg model

$$\Gamma_1^B = -2\frac{\langle \mathbf{e}_0^+, \mathbf{e}_0^- \rangle}{\|\mathbf{e}_0^+\|^2 + \|\mathbf{e}_0^-\|^2} = -\frac{2\beta}{1 + \beta^2}$$

Note that $|\Gamma_1^B| \leq 1$ for any value of β. Updating the forward and backward errors,

$$\begin{aligned} e_1^+(n) &= e_0^+(n) + \Gamma_1^B e_0^-(n-1) &&= \beta^n u(n) + \Gamma_1^B \beta^{n-1} u(n-1) \\ e_1^-(n) &= e_0^-(n-1) + \Gamma_1^B e_0^+(n) &&= \beta^{n-1} u(n-1) + \Gamma_1^B \beta^n u(n) \end{aligned}$$

Computing the error norms and inner products for $e_1^+(n)$ and $e_1^-(n)$ we have, after a lot of algebra,

$$\|\mathbf{e}_1^+\|^2 = \sum_{n=2}^{N}[e_1^+(n)]^2 = \beta^4(1-\beta^2)\frac{1-\beta^{2(N-1)}}{(1+\beta^2)^2}$$

$$\|\mathbf{e}_1^-\|^2 = \sum_{n=2}^{N}[e_1^-(n-1)]^2 = (1-\beta^2)\frac{1-\beta^{2(N-1)}}{(1+\beta^2)^2}$$

$$\langle \mathbf{e}_1^-, \mathbf{e}_1^+ \rangle = \sum_{n=2}^{N} e_1^+(n)e_1^-(n-1) = -\beta^2(1-\beta^2)\frac{1-\beta^{2(N-1)}}{(1+\beta^2)^2}$$

Thus, the second reflection coefficient is equal to

$$\Gamma_2^B = -2\frac{\langle \mathbf{e}_1^+, \mathbf{e}_1^- \rangle}{\|\mathbf{e}_1^+\|^2 + \|\mathbf{e}_1^-\|^2} = 2\frac{\beta^2(1-\beta^2)}{\beta^4(1-\beta^2)+(1-\beta^2)} = \frac{2\beta^2}{1+\beta^4}$$

Let us now evaluate the Burg error for this model using the recursion given in Eq. (6.81). Assuming that $N \gg 1$ and $|\beta| < 1$, for the zeroth-order error we have

$$\mathcal{E}_0^B = 2\sum_{n=0}^{N} x^2(n) = 2\frac{1-\beta^{2(N+1)}}{1-\beta^2} \approx \frac{2}{1-\beta^2}$$

For the first-order error,

$$\begin{aligned}\mathcal{E}_1^B &= \left\{\mathcal{E}_0^B - [e_0^+(0)]^2 - [e_0^-(N)]^2\right\}\left[1-|\Gamma_1^B|^2\right] \\ &= \left\{\mathcal{E}_0^B - 1 - \beta^{2N}\right\}\left[1-|\Gamma_1^B|^2\right] \\ &\approx \frac{1-\beta^2}{1+\beta^2}\end{aligned}$$

Finally, for the second-order error we find, after some algebra,

$$\begin{aligned}\mathcal{E}_2^B &= \left\{\mathcal{E}_1^B - [e_1^+(1)]^2 - [e_1^-(N)]^2\right\}\left[1-|\Gamma_2^B|^2\right] \\ &\approx \frac{(1-\beta^2)^3}{1+\beta^4}\end{aligned}$$

As a final example, we again consider the signal in Example 6.5.1 which is the unit sample response of the second-order system

$$H(z) = \frac{1}{1-0.12z^{-1}-0.456z^{-2}+0.6z^{-3}}$$

which has lattice filter coefficients

$$\Gamma = [0.6,\ -0.6,\ 0.6]^T$$

Using the MATLAB program for the Burg algorithm we find, with $p = 3$ and $N = 60$, the reflection coefficient sequence

$$\Gamma^B = [0.6753,\ -0.6653,\ 0.8920]^T$$

which corresponds to a filter having a system function

$$\hat{H}_3(z) = \frac{1}{1-0.3675z^{-1}-0.4637z^{-2}+0.8920z^{-3}}$$

Furthermore, the sequence of squared errors is

$$\underline{\mathcal{E}}^B = [3.4371,\ 1.5627,\ 0.3134]^T$$

Again, if we increase the model order to $p = 5$, the next two reflection coefficients are

$$\Gamma_4^B = -0.5324 \quad ; \quad \Gamma_5^B = 0.0390$$

and the corresponding errors are

$$\mathcal{E}_4^B = 0.2027 \quad ; \quad \mathcal{E}_5^B = 0.2022$$

An interesting property of the Burg algorithm is that Γ_j^B is the *harmonic mean* of the reflection coefficients Γ_j^+ and Γ_j^-. Specifically, note that if we are given the forward and backward prediction errors $e_j^+(n)$ and $e_j^-(n)$, and if we were to compute the jth reflection coefficient using the forward covariance method, the backward covariance method, and Burg's method, then these reflection coefficients would be related to each other as follows[5]

$$\frac{2}{\Gamma_j^B} = \frac{1}{\Gamma_j^+} + \frac{1}{\Gamma_j^-} \tag{6.88}$$

It is important to be careful in using this equation, however, because it does not imply that we may use the forward and backward covariance methods to compute Γ_j^+ and Γ_j^- and then use Eq. (6.88) to find Γ_j^B. This relationship between the reflection coefficients is only valid when Γ_j^+ and Γ_j^- are computed using the forward and backward prediction errors that are produced with the Burg algorithm, i.e., using the lattice filter with reflection coefficients Γ_j^B.

6.5.5 Modified Covariance Method

In the previous section we derived the Burg recursion, which finds the reflection coefficients for an all-pole model by sequentially minimizing the sum of the squares of the forward and backward prediction errors. In this section we look at the *modified covariance method* or *forward-backward algorithm* for all-pole signal modeling. As with the Burg algorithm, the modified covariance method minimizes the sum of the squares of the forward and backward prediction errors,

$$\mathcal{E}_p^M = \mathcal{E}_p^+ + \mathcal{E}_p^-$$

The difference, however, between the two approaches is that, in the modified covariance method, the minimization is not performed sequentially. In other words, for a pth-order model, the modified covariance method finds the set of reflection coefficients or, equivalently, the set of transversal filter coefficients $a_p(k)$, that minimize $\mathcal{E}_p^M$.

To find the filter coefficients that minimize $\mathcal{E}_p^M$ we set the derivative of $\mathcal{E}_p^M$ with respect to $a_p^*(l)$ equal to zero for $l = 1, 2, \ldots, p$ as we did in the covariance method. Since

$$e_p^+(n) = x(n) + \sum_{k=1}^{p} a_p(k)x(n-k) \tag{6.89}$$

[5]This relationship follows directly from the definitions given in Eqs. (6.74), (6.66) and (6.68).

and

$$e_p^-(n) = x(n-p) + \sum_{k=1}^{p} a_p^*(k)x(n-p+k) \tag{6.90}$$

then

$$\frac{\partial \mathcal{E}_p^M}{\partial a_p^*(l)} = \sum_{n=p}^{N}\left[e_p^+(n)\frac{\partial\left[e_p^+(n)\right]^*}{\partial a_p^*(l)} + \left[e_p^-(n)\right]^*\frac{\partial e_p^-(n)}{a_p^*(l)}\right]$$

$$= \sum_{n=p}^{N}\left[e_p^+(n)x^*(n-l) + \left[e_p^-(n)\right]^* x(n-p+l)\right] = 0 \tag{6.91}$$

Substituting Eqs. (6.89) and (6.90) into Eq. (6.91) and simplifying we find that the normal equations for the modified covariance method are given by

$$\boxed{\sum_{k=1}^{p}\left[r_x(l,k) + r_x(p-k, p-l)\right]a_p(k) = -\left[r_x(l,0) + r_x(p, p-l)\right]; \quad l = 1, \ldots, p} \tag{6.92}$$

where

$$r_x(l,k) = \sum_{n=p}^{N} x(n-k)x^*(n-l) \tag{6.93}$$

Recall that if we were to use the covariance method and minimize $\mathcal{E}_p^+$, then the normal equations would be

$$\sum_{k=1}^{p} r_x(l,k)a_p(k) = -r_x(l,0) \tag{6.94}$$

On the other hand, if we were to minimize $\mathcal{E}_p^-$, then the normal equations would be

$$\sum_{k=1}^{p} r_x(p-k, p-l)a_p(k) = -r_x(p, p-l) \tag{6.95}$$

Therefore, the normal equations for the modified covariance method is a combination of these two sets of equations.

For the modified covariance error,

$$\mathcal{E}_p^M = \sum_{n=p}^{N}\left\{\left|e_p^+(n)\right|^2 + \left|e_p^-(n)\right|^2\right\}$$

we may use the orthogonality condition in Eq. (6.91) to express $\mathcal{E}_p^M$ as follows

$$\mathcal{E}_p^M = \sum_{n=p}^{N}\left[e_p^+(n)x^*(n) + \left[e_p^-(n)\right]^* x(n-p)\right]$$

Substituting the expressions given in Eqs. (6.89) and (6.90) for $e_p^+(n)$ and $e_p^-(n)$ and simplifying, we have

The Modified Covariance Method

```
function [a,err] = mcov(x,p)
%
x   = x(:);
N   = length(x);
if p>=length(x), error('Model order too large'), end
X   = toeplitz(x(p+1:N),flipud(x(1:p+1)));
R   = X'*X;
R1  = R(2:p+1,2:p+1);
R2  = flipud(fliplr(R(1:p,1:p)));
b1  = R(2:p+1,1);
b2  = flipud(R(1:p,p+1));
a   = [1 ; -(R1+R2)\(b1+b2)];
err = R(1,:)*a+fliplr(R(p+1,:))*a;
end;
```

Figure 6.18 *A* MATLAB *program for finding a pth-order all-pole model for a signal $x(n)$ using the modified covariance method.*

$$\mathcal{E}_p^M = r_x(0,0) + r_x(p,p) + \sum_{k=1}^{p} a(k)\left[r_x(0,k) + r_x(p,p-k)\right] \tag{6.96}$$

A MATLAB program for finding a pth-order all-pole model for $x(n)$ is given in Fig. 6.18.

Example 6.5.4 ***The Modified Covariance Method***

Let us use the modified covariance method to find a second-order all-pole model for

$$x(n) = \beta^n u(n)$$

which is assumed to be known for $n = 0, 1, 2, \ldots, N$. We begin by evaluating the autocorrelations given in Eq. (6.93). With $p = 2$ we have

$$\begin{aligned} r_x(k,l) &= \sum_{n=2}^{N} \beta^{n-k}\beta^{n-l} = \beta^{-k-l}\sum_{n=2}^{N}\beta^{2n} \\ &= \beta^4 \frac{1-\beta^{2(N-1)}}{1-\beta^2}\beta^{-k-l} \equiv \lambda\beta^{4-k-l} \end{aligned}$$

where we have defined, for convenience,

$$\lambda = \frac{1-\beta^{2(N-1)}}{1-\beta^2}$$

The coefficients of the second-order model, $a(1)$ and $a(2)$, are the solution to the linear equations

$$\begin{bmatrix} r_x(1,1)+r_x(1,1) & r_x(1,2)+r_x(0,1) \\ r_x(2,1)+r_x(1,0) & r_x(2,2)+r_x(0,0) \end{bmatrix}\begin{bmatrix} a(1) \\ a(2) \end{bmatrix} = -\begin{bmatrix} r_x(1,0)+r_x(2,1) \\ r_x(2,0)+r_x(2,0) \end{bmatrix}$$

Inserting the computed values for $r_x(k,l)$ we have

$$\lambda \begin{bmatrix} 2\beta^2 & \beta+\beta^3 \\ \beta+\beta^3 & 1+\beta^4 \end{bmatrix} \begin{bmatrix} a(1) \\ a(2) \end{bmatrix} = -\lambda \begin{bmatrix} \beta+\beta^3 \\ 2\beta^2 \end{bmatrix}$$

Solving these equations for $a(1)$ and $a(2)$ we find that $a(1) = -(1+\beta^2)/\beta$ and $a(2) = 1$ which leads to an all-pole model with

$$A(z) = 1 - \frac{1+\beta^2}{\beta} z^{-1} + z^{-2}$$

Note that since the roots of $A(z)$ are at $z = \beta$ and $z = 1/\beta$ then the model is *unstable* for any value of β. We find an interesting result, however, when when we examine the error. Inserting the given values for the autocorrelations and the computed values for $a(k)$ into Eq. (6.96) we have

$$\begin{aligned} \mathcal{E}_2^M &= \lambda\left[\beta^4 + 1 + a(1)\left(\beta+\beta^3\right) + a(2)\left(2\beta^2\right)\right] \\ &= \lambda\left[\beta^4 + 1 - \frac{1+\beta^2}{\beta}\beta(1+\beta^2) + 2\beta^2\right] = 0 \end{aligned}$$

Thus, the sum of the squares of the forward and backward prediction errors is equal to zero!

6.6 STOCHASTIC MODELING

In Section 6.5 we considered several different approaches for all-pole modeling of deterministic signals. As discussed in Section 4.7 of Chapter 4, in some applications it is necessary to find models for stochastic processes. We may easily adapt the lattice methods developed in this chapter to stochastic processes by replacing the least squares errors with mean-square errors. For example, we may use the forward covariance method to model a stochastic process $x(n)$ as the output of an all-pole filter driven by white noise by sequentially minimizing the mean-square forward prediction error,

$$\xi_j^+ = E\left\{\left|e_j^+(n)\right|^2\right\}$$

As we did in the deterministic case, the value for Γ_j^+ that minimizes ξ_j^+ is found by setting the derivative of ξ_j^+ with respect to $(\Gamma_j^+)^*$ equal to zero and solving for Γ_j^+. The result, not too surprisingly, is

$$\boxed{\Gamma_j^+ = -\frac{E\left\{e_{j-1}^+(n)\left[e_{j-1}^-(n-1)\right]^*\right\}}{E\left\{\left|e_{j-1}^-(n-1)\right|^2\right\}}} \tag{6.97}$$

which is related to the reflection coefficients in the deterministic case by replacing the sums in Eq. (6.66) with expectations. Instead of ξ_j^+, if we minimize the mean-square backward prediction error,

$$\xi_j^- = E\left\{\left|e_j^-(n)\right|^2\right\}$$

then we find that the jth reflection coefficient is given by

$$\boxed{\Gamma_j^- = -\frac{E\left\{e_{j-1}^+(n)[e_{j-1}^-(n-1)]^*\right\}}{E\left\{|e_{j-1}^+(n)|^2\right\}}} \tag{6.98}$$

On the other hand, minimizing the mean-square Burg error

$$\xi_j^B = E\left\{|e_j^+(n)|^2 + |e_j^-(n)|^2\right\}$$

results in reflection coefficients given by

$$\boxed{\Gamma_j^B = -2\frac{E\left\{e_{j-1}^+(n)[e_{j-1}^-(n-1)]^*\right\}}{E\left\{|e_{j-1}^+(n)|^2\right\} + E\left\{|e_{j-1}^-(n-1)|^2\right\}}} \tag{6.99}$$

Of course, in any practical application these ensemble averages are unknown and it is necessary to estimate them from the given data. Replacing the expected values with estimates that are formed by taking time averages, however, takes us back to the deterministic techniques discussed in Section 6.5. For example, if we use the estimates

$$\hat{E}\left\{e_{j-1}^+(n)[e_{j-1}^-(n-1)]^*\right\} = \sum_{n=j}^{N} e_{j-1}^+(n)[e_{j-1}^-(n-1)]^*$$

and

$$\hat{E}\left\{|e_j^-(n)|^2\right\} = \sum_{n=j}^{N} |e_j^-(n)|^2$$

in Eq. (6.97), then we have the expression for Γ_j^+ given in Eq. (6.66).

Using the stochastic formulation of the lattice all-pole signal modeling techniques, we may establish some useful and interesting orthogonality relations. In the deterministic case these relations may be assumed to be approximately true. One orthogonality relation that we will find particularly useful in our discussions of adaptive lattice filters (see Section 9.2.8) is the condition that the backward prediction errors $e_i^-(n)$ and $e_j^-(n)$ are orthogonal when $i \neq j$,

$$E\left\{e_i^-(n)[e_j^-(n)]^*\right\} = 0 \quad ; \quad i \neq j$$

To establish this orthogonality relation, recall from Eq. (6.12) that the backward prediction errors $e_j^-(n)$ are generated by filtering $x(n)$ with the backward prediction error filter $A_j^R(z)$ as follows

$$e_j^-(n) = x(n) * a_j^R(n) = x(n-j) + \sum_{k=1}^{j} a_j^*(k)x(n-j+k)$$

Writing these equations in matrix form for $j = 0, 1, \ldots, p$ we have

$$\begin{bmatrix} e_0^-(n) \\ e_1^-(n) \\ e_2^-(n) \\ \vdots \\ e_p^-(n) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ a_1^*(1) & 1 & 0 & \cdots & 0 \\ a_2^*(2) & a_2^*(1) & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ a_p^*(p) & a_p^*(p-1) & a_p^*(p-2) & \cdots & 1 \end{bmatrix} \begin{bmatrix} x(n) \\ x(n-1) \\ x(n-2) \\ \vdots \\ x(n-p) \end{bmatrix} \tag{6.100}$$

which may be written in vector form as follows

$$\mathbf{e}^-(n) = \mathbf{A}_p^T \mathbf{x}(n)$$

where $\mathbf{A}_p$ is the matrix of predictor coefficients defined in Eq. (5.84) (see p. 250). Therefore, it follows that the correlation matrix for the backward prediction errors is

$$E\{[\mathbf{e}^-(n)]^* [\mathbf{e}^-(n)]^T\} = E\{\mathbf{A}_p^H \mathbf{x}^*(n)\mathbf{x}^T(n)\mathbf{A}_p\} = \mathbf{A}_p^H \mathbf{R}_p \mathbf{A}_p$$

where $\mathbf{R}_p$ is the autocorrelation matrix for $x(n)$. However, as we saw in Section 5.2.7, the matrix $\mathbf{A}_p$ diagonalizes the autocorrelation matrix $\mathbf{R}_p$. Therefore

$$E\{[\mathbf{e}^-(n)]^* [\mathbf{e}^-(n)]^T\} = \mathbf{D}_p = \text{diag}\{\epsilon_0, \epsilon_1, \ldots, \epsilon_p\}$$

and we have

$$\boxed{E\{e_i^-(n)[e_j^-(n)]^*\} = \begin{cases} \epsilon_j & ; \ i = j \\ 0 & ; \ i \neq j \end{cases}} \tag{6.101}$$

and the backward prediction errors are orthogonal. Thus, the lattice filter transforms the input sequence, $x(n), x(n-1), \ldots, x(n-p)$ into an orthogonal sequence, $e_0^-(n), e_1^-(n), \ldots, e_p^-(n)$. This orthogonalization results in a *decoupling* of the successive errors form from each other.

6.7 SUMMARY

In the first part of this chapter we derived a number of different lattice filter structures for all-zero, all-pole, and pole-zero filters. These lattice filters are important in signal processing applications due their modularity, robustness to finite precision effects, and ease in ensuring filter stability. Using the Levinson-Durbin recursion, we derived a two-multiplier lattice filter and showed how the forward and backward prediction errors, $e_j^+(n)$ and $e_j^-(n)$, respectively, are generated by this filter. Another FIR lattice filter structure, one that is based on the three-term recurrence for the singular predictor polynomials, $S_j(z)$, was then derived. This structure, known as the split lattice filter, is parameterized in terms of the coefficients δ_j produced by the split Levinson recursion. Next, we developed several different lattice filter structures for all-pole filters. Beginning with the two-multiplier all-pole lattice that follows directly from the FIR lattice filter, it was shown how this structure could be used to implement an allpass filter. Through a sequence of flowgraph manipulations, we then derived the Kelly-Lochbaum lattice filter, the normalized lattice filter, and the one-multiplier lattice filter. Finally, a lattice filter structure was presented for implementing a filter that

has both poles and zeros. This structure consists of an all-pole lattice along with a set of tap weights $c_q(k)$ that form a linear combination of the backward prediction errors $e_j^-(n)$ to produce the filter output.

The second part of the chapter was concerned with the application of lattice filters to signal modeling. Since a lattice filter is parameterized in terms of its reflection coefficients, Γ_j, or some set of parameters derived from these coefficients, we considered a number of different ways to optimally select the reflection coefficients. The first approach, known as the forward covariance method, involved the sequential minimization of the forward prediction error using a covariance-type error, i.e., one that does not require knowledge of the signal outside the interval $[0, N]$. The second, the backward covariance method, performed a sequential minimization of the backward prediction errors. Since neither the forward nor the backward covariance methods produce a model that is guaranteed to be stable, we then considered the sequential minimization of the Burg error, which is the *sum* of the forward and backward prediction errors. It was shown that the sequential minimization of the Burg error leads to a model that is guaranteed to be stable. Finally, we consided the modified covariance method, which drops the requirement that the minimization of the Burg error be done sequentially, and finds the global minimum. Although optimum in the sense of minimizing the Burg error, the model produced with this method is not guaranteed to be stable.

In the last section of this chapter, we looked briefly at how the lattice methods for modeling deterministic signals may be generalized to model stochastic processes. What we saw was that a stochastic signal model may be derived using the approaches used for deterministic signals by simply replacing the inner product for deterministic signals with an inner product that is appropriate for stochastic processes, i.e., the expected value of the product of two processes.

References

1. J. Burg, "Maximum entropy spectral analysis," *Proc. 37th Meeting of Soc. of Exploration Geophysicists*, Oklahoma City, OK., October 1967 (reprinted in *Modern Spectrum Estimation*, D.G. Childers, Ed., IEEE Press, New York).
2. J. Burg, "Maximum entropy spectral analysis," Ph.D. dissertation, Stanford University, Stanford, CA, May, 1975.
3. J. R. Deller Jr., J. G. Proakis, and J. H. L. Hansen, *Discrete-time Processing of Speech Signals*, MacMillan, New York, 1993.
4. P. Delsarte and Y. V. Genin, "On the splitting of classical algorithms in linear prediction theory," *IEEE Trans. Acoust., Speech, Sig. Proc.*, vol. ASSP-35, no. 5, pp. 645–653, May 1987.
5. F. Itakura and S. Saito, "Digital filtering techniques for speech analysis and synthesis," *7th Int. Conf. Acoustics*, Budapest, 1971, Paper 25-C-1.
6. J. L. Kelly and C. C. Lochbaum, "Speech synthesis," *Proc. 4th Int. Congress on Acoust.*, vol. G42, pp. 1–4, 1962.
7. J. Makhoul, "A class of all-zero lattice digital filters: Properties and applications," *IEEE Trans. Acoust., Speech, Sig. Proc.*, vol. ASSP-26, no. 4, pp. 304–314, Aug. 1978.
8. J. Makhoul, "Stable and efficient lattice methods for linear prediction," *IEEE Trans. Acoust., Speech, Sig. Proc.*, vol. ASSP-25, no. 5, pp. 423–428, October 1977.
9. J. D. Markel and A. H. Gray, Jr., *Linear Prediction of Speech*, Springer-Verlag, New York, 1976.
10. J. D. Markel and A. H. Gray, Jr., "Roundoff noise characteristics of a class of orthogonal polynomial structures," *IEEE Trans. Acoust., Speech, Sig. Proc.*, vol. ASSP-23, no. 5, pp. 473–486, October 1975.
11. S. J. Orfanidis, *Optimal Signal Processing: An Introduction*, 2nd Ed., Macmillan, New York, 1988.
12. E. A. Robinson and S. Treitel, *Geophysical Signal Analysis*, Prentice-Hall, Englewood Cliffs, N. J., 1980.

13. R. Viswanathan and J. Makhoul, "Quantization properties of transmission parameters in linear predictive systems," *IEEE Trans. Acoust., Speech, Sig. Proc.*, vol. ASSP-22, no. 3, pp. 309–321, June 1975.

6.8 PROBLEMS

6.1. Design a two-pole lattice filter that has poles at $re^{j\theta}$ and $re^{-j\theta}$ and draw a carefully labeled flowgraph of your filter.

6.2. Consider the all-pole filter

$$H(z) = \frac{1}{1 - 0.2z^{-1} + 0.9z^{-2} + 0.6z^{-3}}$$

Draw the flowgraph for a lattice filter implementation of $H(z)$ using

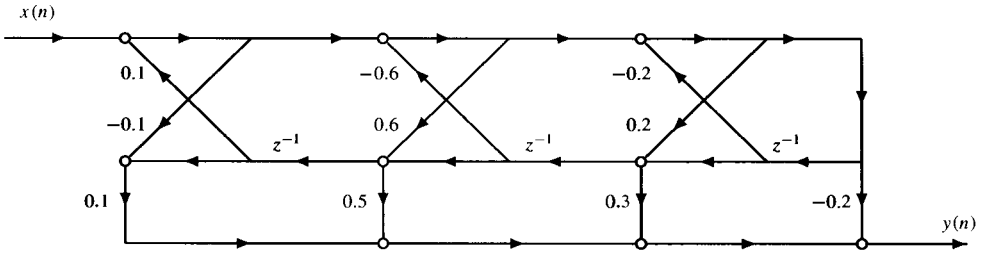
(a) A Kelly-Lochbaum lattice filter.

(b) A normalized lattice filter.

(c) A one-muliplier lattice filter.

For each structure, determine the number of multiplies, adds, and delays required to implement the filter and compare them to a direct-form realization of $H(z)$. Based solely on computational considerations, which structure is the most efficient?

6.3. Find the system function $H(z)$ for the lattice filter given in the figure below.



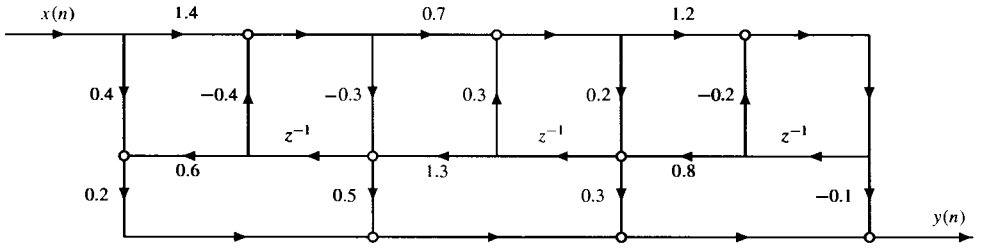
6.4. Sketch a lattice filter structure for each of the following system functions.

(a) $H(z) = \dfrac{2 - z^{-1}}{1 + 0.7z^{-1} + 0.49z^{-2}}$

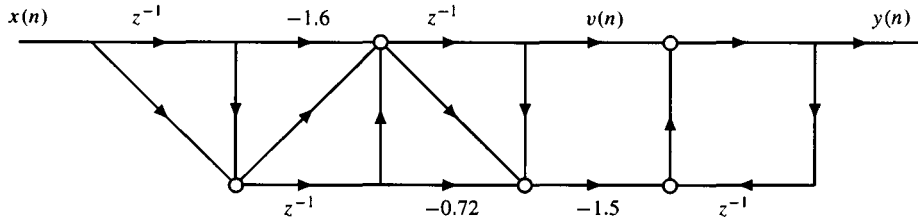
(b) $H(z) = \dfrac{1 + 1.3125z^{-1} + 0.75z^{-2}}{1 + 0.875z^{-1} + 0.75z^{-2}}$

(c) $H(z) = \dfrac{0.75 + 0.875z^{-1} + z^{-2}}{1 + 0.875z^{-1} + 0.75z^{-2}}$

6.5. Determine the system function of the lattice filter shown in the figure below.



6.6. Shown in the figure below is a split lattice filter.



(a) What is the order of the filter (number of poles and zeros)?

(b) Does this filter have minimum phase?

(c) Find the system function $H(z) = Y(z)/X(z)$.

(d) What is the transfer function $V(z)/X(z)$ between $x(n)$ and $v(n)$?

(e) How would you modify this structure to add a zero in $H(z)$ at $z = -1$?

6.7. *True* or *False*: Let $H(z)$ be the system function of a linear shift-invariant filter with coefficients $a_p(k)$ and $b_q(k)$, and let Γ_k and $c_q(k)$ be the coefficients in the lattice filter realization of $H(z)$. If $a_p(k)$ and $b_q(k)$ are modified as follows

$$\widetilde{a}_p(k) = (-1)^k a_p(k) \quad ; \quad \widetilde{b}_p(k) = (-1)^k b_p(k)$$

then the coefficients in the lattice filter are modified in the same way, i.e.,

$$\widetilde{\Gamma}_k = (-1)^k \Gamma_k \quad ; \quad \widetilde{c}_q(k) = (-1)^k c_q(k)$$

6.8. As shown in Fig. 6.2*b*, a lattice filter may be used to generate the forward and backward prediction errors, $e_p^+(n)$ and $e_p^-(n)$, respectively.

(a) What is the relationship between the magnitudes of the discrete-time Fourier transforms of $e_p^+(n)$ and $e_p^-(n)$?

(b) Is it possible to design a realizable filter (causal and stable) that will produce the response $e_p^-(n)$ to the input $e_p^+(n)$? If so, describe how and, if not, state why not.

(c) Is it possible to design a realizable filter that will produce the response $e_p^+(n)$ to the input $e_p^-(n)$? If so, describe how and, if not, state why not.

6.9. The all-pole lattice filter in Fig. 6.7*b* may be used to generate the all-pole approximation $\hat{x}(n)$ to a signal $x(n)$. In this problem, we investigate another use for this filter. Suppose $e_0^+(n)$ is initialized to one at time $n = 0$, and all of the remaining states are set equal to zero. Determine the output of the all-pole filter for $n = 1, 2, \ldots, N$. Hint: Consider the expression for γ_j given in Eq. (5.10) of Chapter 5.

6.10. Consider the following modification to the Burg error

$$\mathcal{E}_j^w = \sum_{n=j}^{N} w_j(n) \left\{ |e_j^+(n)|^2 + |e_j^-(n)|^2 \right\}$$

where $w_j(n)$ is a window that is applied to the forward and backward prediction errors.

(a) Derive an expression that defines the value for the reflection coefficient Γ_j^w that minimizes the modified Burg error $\mathcal{E}_j^w$.

(b) What conditions, if any, are necessary in order to guarantee that the reflection coefficients are bounded by one in magnitude?

6.11. Consider the sequence of reflection coefficients Γ_j^B, Γ_j^I, and $\Gamma_j^{\min}$ where Γ_j^I is the reflection coefficient proposed by Itakura and $\Gamma_j^{\min}$ is the reflection coefficient that is formed using the minimum method (see Section 6.5.3).

(a) Establish the following relationship between these reflection coefficients:

$$|\Gamma_j^{\min}| \leq |\Gamma_j^B| \leq |\Gamma_j^I|$$

(b) Are there any conditions under which all three reflection coefficients will be the same for all j?

(c) Let Γ_j^M be the set of reflection coefficients corresponding to the all-pole model that is derived from the modified covariance method. Is it possible to upper or lower bound these coefficients in terms of Γ_j^B, Γ_j^I, or $\Gamma_j^{\min}$?

6.12. In Section 6.6 it was shown that the backward prediction errors are orthogonal, i.e.,

$$E\left\{e_i^-(n)[e_j^-(n)]^*\right\} = \begin{cases} \epsilon_j & ; \quad i = j \\ 0 & ; \quad i \neq j \end{cases}$$

Establish the following orthogonality conditions:

(a) $E\left\{e_i^+(n)x^*(n-k)\right\} = 0 \quad ; \quad 1 \leq k \leq i$

(b) $E\left\{e_i^-(n)x^*(n-k)\right\} = 0 \quad ; \quad 0 \leq k \leq i-1$

(c) $E\left\{e_i^+(n)x^*(n)\right\} = E\left\{e_i^-(n)x^*(n-i)\right\} = \epsilon_i$

6.13. In this problem it will be shown that the prediction error filters, $A_j(z)$, are *orthogonal* on the unit circle. Specifically, let $P_x(e^{j\omega})$ be the power spectrum of a zero mean random process $x(n)$ and let $A_j(z)$ be the system function of the jth-order prediction error filter. Show that these polynomials satisfy the orthogonality property

$$\int_{-\pi}^{\pi} P_x(e^{j\omega})A_j(e^{j\omega})A_k^*(e^{j\omega})d\omega = \lambda_k \delta_{jk}$$

and find an expression for the constant λ_k. Polynomials that satisfy this orthogonality condition are called Szegö polynomials.

Computer Exercises

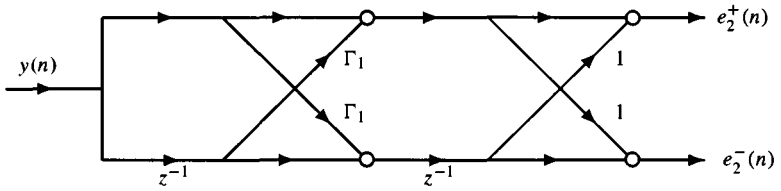
C6.1. One approach for estimating the frequency of a sinusoid from noisy samples is to use a constrained lattice filter. Specifically, consider the following signal

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

where

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

and $w(n)$ is white Gaussian noise with a variance of σ_w^2. Consider the constrained second-order lattice filter shown in the figure below where the second reflection coefficient has been set equal to one.



The outputs of the lattice filter, $e_2^+(n)$ and $e_2^-(n)$, are the second-order forward and backward prediction errors, respectively.

(a) Find the location of the zeros of the lattice filter as a function of the reflection coefficient Γ_1.

(b) Given $y(n)$ for $n = 0, 1, \ldots, N-1$, suppose that we would like to estimate the frequency of the sinusoid. One way to accomplish this is to find the value of the reflection coefficient Γ_1 in the constrained lattice filter that minimizes some error that depends upon $e_2^+(n)$ and $e_2^-(n)$, and to use this reflection coefficient to estimate the frequency. Derive an expression for the value of Γ_1 that minimizes the error

$$\mathcal{E}_2^B = \sum_{n=2}^{N-1} \{[e_2^+(n)]^2 + [e_2^-(n)]^2\}$$

Your expression should be expressed only in terms of the observations $y(n)$.

(c) Describe how you would use Γ_1 found in part (b) to estimate the frequency ω_0.

(d) Evaluate the performance of your constrained lattice filter in estimating the frequency of the sinusoid. Consider the effect of the data record length, signal to noise ratio, A^2/σ_w^2, and frequency, ω_0. Compare the performance of the constrained lattice filter to frequency estimates derived from the Burg algorithm, the modified covariance method, and the autocorrelation method.

(e) How may this frequency estimation technique be modified to estimate the frequency of a complex harmonic process, i.e., if

$$y(n) = x(n) + w(n) \qquad \text{for} \qquad n = 0, 1, \ldots, N-1$$

where

$$x(n) = Ae^{j(n\omega_0+\phi)}$$

and $w(n)$ is complex white Gaussian noise.

(f) How may the frequency estimation technique in part (e) be modified to estimate the frequencies of two complex-valued harmonic processes:

$$y(n) = A_1 e^{j(n\omega_1+\phi_1)} + A_2 e^{j(n\omega_2+\phi_2)} + w(n)$$

C6.2. In Section 6.4.3 we presented a recursion for converting the direct-form coefficients $b_q(k)$ into the lattice filter coefficients $c_q(k)$ and a recursion for converting the coefficients $c_q(k)$ into $b_q(k)$.

(a) Create an m-file `btoc.m` for converting the coefficients $b_q(k)$ into $c_q(k)$.

(b) Create an m-file `ctob.m` for converting the coefficients $c_q(k)$ into $b_q(k)$.

C6.3. The m-file `fcov.m` given in Fig. 6.15 finds a model for a signal $x(n)$ using the forward covariance method. Modify this program and create an m-file `bcov.m` that will find a model for $x(n)$ using the backwards covariance method.

C6.4. Consider the signal $x(n) = (n + 1)u(n)$ for $n = 0, 1, \ldots, N$.

(a) Find the first-order model and the minimum error using the covariance method, the forward and backward covariance methods, the Burg algorithm, and the modified covariance method. Comment on the differences in the models and compare $x(n)$ to the model, $\hat{x}(n)$.

(b) Repeat part (a) using a second-order model. What behavior do you observe for the forward and backward covariance methods?

(c) Examine what happens using the Burg algorithm for model orders $p > 2$. Can you suggest a way to estimate the model order by looking at the sequence of errors $\mathcal{E}_j^B$?

(d) Repeat parts (a) and (b) for the signal $x(n) = n(0.9)^n$. Add noise to $x(n)$ and discuss the sensitivity of your model to the noise.

C6.5. Generate a signal, $x(n)$, of the form

$$x(n) = \alpha^n + \beta^n$$

for $n = 0, 1, 2, \ldots, 15$.

(a) With $\alpha = 0.8$ and $\beta = 1.25$, find a second order all-pole model for $x(n)$ using the modified covariance method. What is the modeling error? What can you say about the stability of the model?

(b) Repeat part (a) using Burg's method, the covariance method, and the autocorrelation method. Discuss the differences that you observe between each of the modeling techniques.

(c) Repeat parts (a) and (b) with $\alpha = 0.75$ and $\beta = 2$.

C6.6. Write an m-file for finding the model for a signal using the method proposed by Itakura (see Section 6.5.3). Compare the effectiveness of Itakura's method to the forward covariance method and Burg's method on a number of different signals.

C6.7. Modify the m-file for the Burg algorithm given in Fig. 6.17 to incorporate the recursion for the denominator given in Eq. (6.86). How much savings is there in terms of the number of multiplies and adds? Do you notice any degradation in the algorithm resulting from numerical errors introduced by finite precision effects?

7 OPTIMUM FILTERS

7.1 INTRODUCTION

The estimation of one signal from another is one of the most important problems in signal processing and it embraces a wide range of interesting applications. In many of these applications the desired signal, whether is it speech, a radar signal, an EEG, or an image, is not available or observed directly. Instead, for a variety of reasons, the desired signal may be noisy and distorted. For example, the equipment used to measure the signal may be of limited resolution, the signal may be observed in the presence of noise or other interfering signals, or it may be distorted due to the propagation of the signal from the source to the receiver as in a digital communication system. In very simple and idealized environments, it may be possible to design a classical filter such as a lowpass, highpass, or bandpass filter, to *restore* the desired signal from the measured data. Rarely, however, will these filters be *optimum* in the sense of producing the *best* estimate of the signal. Therefore, in this chapter we consider the design of *optimum digital filters*, which include the digital Wiener filter and the discrete Kalman filter [2].

In the 1940s, driven by important applications in communication theory, Norbert Wiener pioneered research in the problem of designing a filter that would produce the optimum estimate of a signal from a noisy measurement or observation. The discrete form of the Wiener filtering problem, shown in Fig. 7.1, is to design a filter to recover a signal $d(n)$ from noisy observations

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

Assuming that both $d(n)$ and $v(n)$ are wide-sense stationary random processes, Wiener considered the problem of designing the filter that would produce the minimum mean-square error estimate of $d(n)$. Thus, with

$$\xi = E\{|e(n)|^2\}$$

where

$$e(n) = d(n) - \hat{d}(n)$$

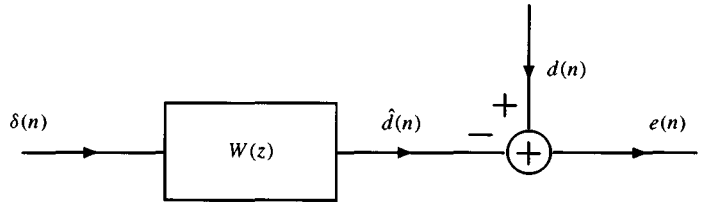


Figure 7.1 *Illustration of the general Wiener filtering problem. Given two wide-sense stationary processes, $x(n)$ and $d(n)$, that are statistically related to each other, the filter $W(z)$ is to produce the minimum mean-square error estimate, $\hat{d}(n)$, of $d(n)$.*

the problem is to find the filter that minimizes ξ. We begin this chapter by considering the general problem of Wiener filtering in which a linear shift-invariant filter, $W(z)$, is to be designed that will filter a given signal, $x(n)$, to produce the minimum mean square estimate, $\hat{d}(n)$, of another signal, $d(n)$. Depending upon how the signals $x(n)$ and $d(n)$ are related to each other, a number of different and important problems may be cast into a Wiener filtering framework. Some of the problems that will be considered in this chapter include:

1. **Filtering**. This is the classic problem considered by Wiener in which we are given $x(n) = d(n) + v(n)$ and the goal is to estimate $d(n)$ using a causal filter, i.e., to estimate $d(n)$ from the current and past values of $x(n)$.
2. **Smoothing**. This is the same as the *filtering* problem except that the filter is allowed to be noncausal. A Wiener smoothing filter, for example, may be designed to estimate $d(n)$ from $x(n) = d(n) + v(n)$ using all of the available data.
3. **Prediction**. If $d(n) = x(n + 1)$ and $W(z)$ is a causal filter, then the Wiener filter becomes a linear predictor. In this case, the filter is to produce a prediction (estimate) of $x(n + 1)$ in terms of a linear combination of previous values of $x(n)$.
4. **Deconvolution**. When $x(n) = d(n) * g(n) + v(n)$ with $g(n)$ being the unit sample response of a linear shift-invariant filter, the Wiener filter becomes a deconvolution filter.

First, we consider the design of FIR Wiener filters in Section 7.2. The main result here will be the derivation of the discrete form of the Wiener-Hopf equations which specify the filter coefficients of the optimum (minimum mse) filter. Solutions to the Wiener-Hopf equations are then given for the cases of filtering, smoothing, prediction, and noise cancellation. In Section 7.3 we then consider the design of IIR Wiener filters. First, in Section 7.3.1 we solve the noncausal Wiener filtering problem. Then, in Section 7.3.2, the problem of designing a causal Wiener filter is considered. Unlike the noncausal Wiener filter, solving for the optimum causal Wiener filter is a nonlinear problem that requires a spectral factorization of the power spectrum of the input process, $x(n)$. The design of a causal Wiener filter is illustrated with examples of Wiener filtering, Wiener prediction, and Wiener deconvolution. Finally, in Section 7.4 we consider recursive approaches to signal estimation and derive what is known as the discrete Kalman filter. Unlike the Wiener filter, which is a linear shift-invariant filter for estimating stationary processes, the Kalman filter is shift-varying and applicable to nonstationary as well as stationary processes.

7.2 THE FIR WIENER FILTER

In this section we consider the design of an FIR Wiener filter that produces the minimum mean-square estimate of a given process $d(n)$ by filtering a set of observations of a statistically related process $x(n)$. It is assumed that $x(n)$ and $d(n)$ are jointly wide-sense stationary with known autocorrelations, $r_x(k)$ and $r_d(k)$, and known cross-correlation $r_{dx}(k)$. Denoting the unit sample response of the Wiener filter by $w(n)$, and assuming a $(p-1)$st-order filter, the system function is

$$W(z) = \sum_{n=0}^{p-1} w(n) z^{-n}$$

With $x(n)$ the input to the filter, the output, which we denote by $\hat{d}(n)$, is the convolution of $w(n)$ with $x(n)$,

$$\hat{d}(n) = \sum_{l=0}^{p-1} w(l) x(n-l) \tag{7.1}$$

The Wiener filter design problem requires that we find the filter coefficients, $w(k)$, that minimize the mean-square error[1]

$$\xi = E\{|e(n)|^2\} = E\{|d(n) - \hat{d}(n)|^2\} \tag{7.2}$$

As discussed in Section 2.3.10 of Chapter 2, in order for a set of filter coefficients to minimize ξ it is necessary and sufficient that the derivative of ξ with respect to $w^*(k)$ be equal to zero for $k = 0, 1, \ldots, p-1$,

$$\frac{\partial \xi}{\partial w^*(k)} = \frac{\partial}{\partial w^*(k)} E\{e(n)e^*(n)\} = E\left\{e(n)\frac{\partial e^*(n)}{\partial w^*(k)}\right\} = 0 \tag{7.3}$$

With

$$e(n) = d(n) - \sum_{l=0}^{p-1} w(l) x(n-l) \tag{7.4}$$

it follows that

$$\frac{\partial e^*(n)}{\partial w^*(k)} = -x^*(n-k)$$

and Eq. (7.3) becomes

$$\boxed{E\{e(n)x^*(n-k)\} = 0 \quad ; \quad k = 0, 1, \ldots, p-1} \tag{7.5}$$

which is known as the *orthogonality principle* or the *projection theorem*.[2] Substituting Eq. (7.4) into Eq. (7.5) we have

$$E\{d(n)x^*(n-k)\} - \sum_{l=0}^{p-1} w(l) E\{x(n-l)x^*(n-k)\} = 0 \tag{7.6}$$

[1] Note that our wide-sense stationarity assumption implies that the mean-square error does not depend upon n.

[2] Compare this with the orthogonality principle in Chapter 4, p. 146.

Finally, since $x(n)$ and $d(n)$ are jointly WSS then $E\{x(n-l)x^*(n-k)\} = r_x(k-l)$ and $E\{d(n)x^*(n-k)\} = r_{dx}(k)$ and Eq. (7.6) becomes

$$\boxed{\sum_{l=0}^{p-1} w(l)r_x(k-l) = r_{dx}(k) \quad ; \quad k = 0, 1, \ldots, p-1} \tag{7.7}$$

which is a set of p linear equations in the p unknowns $w(k)$, $k = 0, 1, \ldots, p-1$. In matrix form, using the fact that the autocorrelation sequence is conjugate symmetric, $r_x(k) = r_x^*(-k)$, Eq. (7.7) becomes

$$\begin{bmatrix} r_x(0) & r_x^*(1) & \cdots & r_x^*(p-1) \\ r_x(1) & r_x(0) & \cdots & r_x^*(p-2) \\ r_x(2) & r_x(1) & \cdots & r_x^*(p-3) \\ \vdots & \vdots & & \vdots \\ r_x(p-1) & r_x(p-2) & \cdots & r_x(0) \end{bmatrix} \begin{bmatrix} w(0) \\ w(1) \\ w(2) \\ \vdots \\ w(p-1) \end{bmatrix} = \begin{bmatrix} r_{dx}(0) \\ r_{dx}(1) \\ r_{dx}(2) \\ \vdots \\ r_{dx}(p-1) \end{bmatrix} \tag{7.8}$$

which is the matrix form of the *Wiener-Hopf equations*. Equation (7.8) may be written more concisely as

$$\boxed{\mathbf{R}_x\mathbf{w} = \mathbf{r}_{dx}} \tag{7.9}$$

where $\mathbf{R}_x$ is a $p \times p$ Hermitian Toeplitz matrix of autocorrelations, $\mathbf{w}$ is the vector of filter coefficients, and $\mathbf{r}_{dx}$ is the vector of cross-correlations between the desired signal $d(n)$ and the observed signal $x(n)$.

The minimum mean-square error in the estimate of $d(n)$ may be evaluated from Eq. (7.2) as follows. With

$$\begin{aligned} \xi &= E\{|e(n)|^2\} = E\left\{e(n)\left[d(n) - \sum_{l=0}^{p-1} w(l)x(n-l)\right]^*\right\} \\ &= E\{e(n)d^*(n)\} - \sum_{l=0}^{p-1} w^*(l)E\{e(n)x^*(n-l)\} \end{aligned} \tag{7.10}$$

recall that if $w(k)$ is the solution to the Wiener-Hopf equations, then, it follows from Eq. (7.5) that $E\{e(n)x^*(n-k)\} = 0$. Therefore, the second term in Eq. (7.10) is equal to zero and

$$\xi_{\min} = E\{e(n)d^*(n)\} = E\left\{\left[d(n) - \sum_{l=0}^{p-1} w(l)x(n-l)\right]d^*(n)\right\}$$

Finally, taking expected values we have

$$\boxed{\xi_{\min} = r_d(0) - \sum_{l=0}^{p-1} w(l)r_{dx}^*(l)} \tag{7.11}$$

or, using vector notation,

$$\xi_{\min} = r_d(0) - \mathbf{r}_{dx}^H\mathbf{w} \tag{7.12}$$

Alternatively, since

$$\mathbf{w} = \mathbf{R}_x^{-1}\mathbf{r}_{dx}$$

Table 7.1 *The Wiener-Hopf Equations for the FIR Wiener Filter and the Minimum Mean-Square Error*

Wiener-Hopf equations	$\sum_{l=0}^{p-1} w(l) r_x(k-l) = r_{dx}(k) \quad ; \quad k = 0, 1, \ldots, p-1$
Correlations	$r_x(k) = E\{x(n)x^*(n-k)\}$ $r_{dx}(k) = E\{d(n)x^*(n-k)\}$
Minimum error	$\xi_{\min} = r_d(0) - \sum_{l=0}^{p-1} w(l) r_{dx}^*(l)$

the minimum error may also be written explicitly in terms of the autocorrelation matrix $\mathbf{R}_x$ and the cross-correlation vector $\mathbf{r}_{dx}$ as follows:

$$\boxed{\xi_{\min} = r_d(0) - \mathbf{r}_{dx}^H \mathbf{R}_x^{-1} \mathbf{r}_{dx}} \tag{7.13}$$

The FIR Wiener filtering equations are summarized in Table 7.1.

We now look at some Wiener filtering applications that illustrate how to formulate the Wiener filtering problem, set up the Wiener-Hopf equations (7.9), and solve for the filter coefficients.

7.2.1 Filtering

In the filtering problem, a signal $d(n)$ is to be estimated from a noise corrupted observation

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

Filtering, or noise reduction, is an extremely important and pervasive problem that is found in many applications such as the transmission of speech in a noisy environment and the reception of data across a noisy channel. It is also important in the detection and location of targets using sensor arrays, the restoration of old recordings, and the enhancement of images.

Using the results in the previous section, the optimum FIR Wiener filter may be easily derived. It will be assumed that the noise has zero mean and that it is uncorrelated with $d(n)$. Therefore, $E\{d(n)v^*(n-k)\} = 0$ and the cross-correlation between $d(n)$ and $x(n)$ becomes

$$\begin{aligned} r_{dx}(k) &= E\{d(n)x^*(n-k)\} \\ &= E\{d(n)d^*(n-k)\} + E\{d(n)v^*(n-k)\} \\ &= r_d(k) \end{aligned} \tag{7.14}$$

Next, since

$$\begin{aligned} r_x(k) &= E\{x(n+k)x^*(n)\} \\ &= E\{[d(n+k) + v(n+k)][d(n) + v(n)]^*\} \end{aligned} \tag{7.15}$$

with $v(n)$ and $d(n)$ uncorrelated processes it follows that

$$r_x(k) = r_d(k) + r_v(k)$$

Therefore, with $\mathbf{R}_d$ the autocorrelation matrix for $d(n)$, $\mathbf{R}_v$ the autocorrelation matrix for $v(n)$, and $\mathbf{r}_{dx} = \mathbf{r}_d = [r_d(0), \ldots, r_d(p-1)]^T$, the Wiener-Hopf equations become

$$[\mathbf{R}_d + \mathbf{R}_v]\mathbf{w} = \mathbf{r}_d \tag{7.16}$$

In order to simplify these equations any further, however, specific information about the statistics of the signal and noise are required.

Example 7.2.1 *Filtering*

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

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

with $0 < \alpha < 1$, and suppose that $d(n)$ is observed in the presence of uncorrelated white noise, $v(n)$, that has a variance of σ_v^2,

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

Let us design a first-order FIR Wiener filter to reduce the noise in $x(n)$. With

$$W(z) = w(0) + w(1)z^{-1}$$

the Wiener-Hopf equations are

$$\begin{bmatrix} r_x(0) & r_x(1) \\ r_x(1) & r_x(0) \end{bmatrix} \begin{bmatrix} w(0) \\ w(1) \end{bmatrix} = \begin{bmatrix} r_{dx}(0) \\ r_{dx}(1) \end{bmatrix}$$

Since $d(n)$ and $v(n)$ are assumed to be uncorrelated, then

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

and

$$r_x(k) = r_d(k) + r_v(k) = \alpha^{|k|} + \sigma_v^2 \delta(k)$$

Thus, the Wiener-Hopf equations become

$$\begin{bmatrix} 1+\sigma_v^2 & \alpha \\ \alpha & 1+\sigma_v^2 \end{bmatrix} \begin{bmatrix} w(0) \\ w(1) \end{bmatrix} = \begin{bmatrix} 1 \\ \alpha \end{bmatrix}$$

Solving for $w(0)$ and $w(1)$ we have

$$\begin{bmatrix} w(0) \\ w(1) \end{bmatrix} = \frac{1}{(1+\sigma_v^2)^2 - \alpha^2} \begin{bmatrix} 1+\sigma_v^2-\alpha^2 \\ \alpha\sigma_v^2 \end{bmatrix}$$

Therefore, the Wiener filter is

$$W(z) = \frac{1}{(1+\sigma_v^2)^2 - \alpha^2} \left[(1+\sigma_v^2-\alpha^2) + \alpha\sigma_v^2 z^{-1}\right]$$

which has a zero at $z = -\alpha\sigma_v^2/(1+\sigma_v^2-\alpha^2)$. As a specific example, let $\alpha = 0.8$ and $\sigma_v^2 = 1$. In this case the Wiener filter becomes

$$W(e^{j\omega}) = 0.4048 + 0.2381e^{-j\omega}$$

which is a lowpass filter that has a magnitude response as shown in Fig. 7.2*a*. The fact that $W(e^{j\omega})$ is a lowpass filter should not be surprising. Specifically, note that the power spectrum of $d(n)$ is

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

which, for $\alpha = 0.8$ becomes

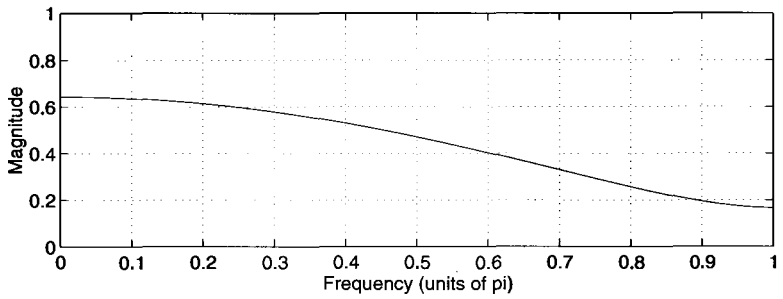
$$P_d(e^{j\omega}) = \frac{0.36}{1.64 - 1.6\cos\omega}$$

as shown in Fig. 7.2*b*. Therefore, since $P_d(e^{j\omega})$ decreases with increasing ω and since the power spectrum of the noise is constant for all ω, then the signal-to-noise ratio decreases with increasing ω. Thus, it follows that the filter should have a frequency response that has a magnitude that *decreases* with increasing ω. For the mean-square error, we have

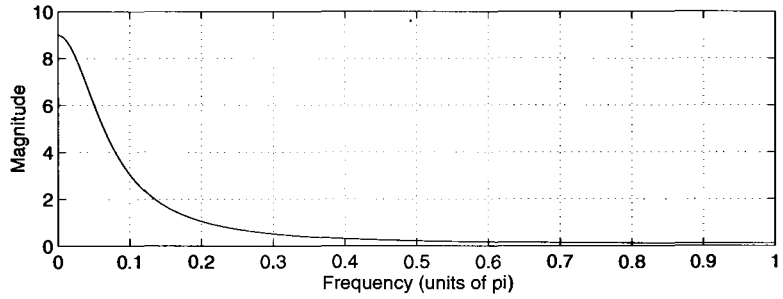
$$\xi_{\min} = E\{|e(n)|^2\} = r_d(0) - w(0)r_{dx}^*(0) - w(1)r_{dx}^*(1) = \sigma_v^2 \frac{1+\sigma_v^2-\alpha^2}{(1+\sigma_v^2)^2-\alpha^2}$$

which, for the case in which $\alpha = 0.8$ and $\sigma_v^2 = 1$, is $\xi_{\min} = 0.4048$.

Let us evaluate how much the signal-to-noise ratio is increased by using the Wiener filter. Prior to filtering, since $r_d(0) = \sigma_d^2 = 1$ and $\sigma_v^2 = 1$, then the power in $d(n)$ is equal to the power in $v(n)$, $E\{|d(n)|^2\} = E\{|v(n)|^2\} = 1$, and the signal to noise ratio (SNR)



(a) The magnitude of the frequency response of the Wiener filter.



(b) The power spectrum of an AR(1) process with an autocorrelation of $r_d(k) = 0.8^{|k|}$.

Figure 7.2 *FIR Wiener filter for filtering an AR(1) process in white noise.*

is 0 dB. After filtering, it follows from Eq. (3.90) on p. 101 that the power in the signal $d'(n) = w(n) * d(n)$ is

$$E\{|d'(n)|^2\} = \mathbf{w}^T\mathbf{R}_d\mathbf{w} = [w(0)\ w(1)]\begin{bmatrix} 1 & \alpha \\ \alpha & 1 \end{bmatrix}\begin{bmatrix} w(0) \\ w(1) \end{bmatrix} = 0.3748$$

and the noise power is

$$E\{|v'(n)|^2\} = \mathbf{w}^T\mathbf{R}_v\mathbf{w} = [w(0)\ w(1)]\begin{bmatrix} w(0) \\ w(1) \end{bmatrix} = 0.2206$$

Therefore, the SNR at the output of the Wiener filter is

$$\text{SNR} = 10\log_{10}\frac{0.3748}{0.2206} = 2.302\ \text{dB}$$

Thus, the Wiener filter increases the SNR by more than 2dB.

7.2.2 Linear Prediction

As discussed in previous chapters, linear prediction is an important problem in many signal processing applications. With noise-free observations, linear prediction is concerned with the estimation (prediction) of $x(n+1)$ in terms of a linear combination of the current and previous values of $x(n)$ as shown in Fig. 7.3. Thus, an FIR linear predictor of order $p-1$ has the form

$$\hat{x}(n+1) = \sum_{k=0}^{p-1} w(k)x(n-k)$$

where $w(k)$ for $k = 0, 1, \ldots, p-1$ are the coefficients of the prediction filter. The linear predictor may be cast into a Wiener filtering problem by setting $d(n) = x(n+1)$ in Fig. 7.1. To set up the Wiener-Hopf equations, all that is needed is to evaluate the cross-correlation between $d(n)$ and $x(n)$. Since

$$r_{dx}(k) = E\{d(n)x^*(n-k)\} = E\{x(n+1)x^*(n-k)\} = r_x(k+1)$$

then the Wiener-Hopf equations for the optimum linear predictor are

$$\begin{bmatrix} r_x(0) & r_x^*(1) & r_x^*(2) & \cdots & r_x^*(p-1) \\ r_x(1) & r_x(0) & r_x^*(1) & \cdots & r_x^*(p-2) \\ r_x(2) & r_x(1) & r_x(0) & \cdots & r_x^*(p-3) \\ \vdots & \vdots & \vdots & & \vdots \\ r_x(p-1) & r_x(p-2) & r_x(p-3) & \cdots & r_x(0) \end{bmatrix}\begin{bmatrix} w(0) \\ w(1) \\ w(2) \\ \vdots \\ w(p-1) \end{bmatrix} = \begin{bmatrix} r_x(1) \\ r_x(2) \\ r_x(3) \\ \vdots \\ r_x(p) \end{bmatrix} \tag{7.17}$$

and the mean-square error is

$$\xi_{\min} = r_x(0) - \sum_{k=0}^{p-1} w(k)r_x^*(k+1)$$

Comparing Eq. (7.17) with the Prony all-pole normal equations given in Eq. (4.79) we see that, except for the fact that $r_x(k)$ is a deterministic autocorrelation sequence in the Prony all-pole normal equations and $r_x(k)$ is a stochastic autocorrelation in Eq. (7.17), the two sets of equations are the same (the minus sign simply changes the sign of the coefficients).

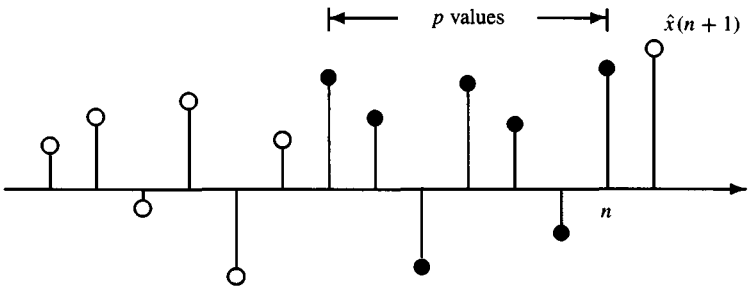


Figure 7.3 *Linear prediction is the problem of finding the minimum mean-square estimate of $x(n+1)$ using a linear combination of the p signal values from $x(n)$ to $x(n-p+1)$.*

Example 7.2.2 *Linear Prediction*

In this example we find the optimum linear predictor for an AR(1) process $x(n)$ that has an autocorrelation sequence given by

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

With a first-order predictor of the form

$$\hat{x}(n+1) = w(0)x(n) + w(1)x(n-1)$$

the Wiener-Hopf equations are

$$\begin{bmatrix} 1 & \alpha \\ \alpha & 1 \end{bmatrix} \begin{bmatrix} w(0) \\ w(1) \end{bmatrix} = \begin{bmatrix} \alpha \\ \alpha^2 \end{bmatrix}$$

Solving for the predictor coefficients we find

$$\begin{bmatrix} w(0) \\ w(1) \end{bmatrix} = \frac{1}{1-\alpha^2} \begin{bmatrix} 1 & -\alpha \\ -\alpha & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ \alpha^2 \end{bmatrix} = \begin{bmatrix} \alpha \\ 0 \end{bmatrix}$$

Therefore, the predictor for $x(n+1)$ is

$$\hat{x}(n+1) = \alpha x(n)$$

and the value of $x(n-1)$ is never used in the prediction of $x(n+1)$. This result may be explained intuitively as follows. Since $x(n)$ is an AR(1) process, it satisfies a first-order difference equation of the form

$$x(n) = \alpha x(n-1) + v(n)$$

or, equivalently,

$$x(n+1) = \alpha x(n) + v(n+1)$$

Since $v(n)$ is a white noise process, it cannot be predicted from previous values of either $x(n)$ or $v(n)$. Therefore, the best estimate of $x(n+1)$ in terms of $x(n)$ and $x(n-1)$ is obtained by replacing $v(n)$ with its expected value as follows

$$\hat{x}(n+1) = \alpha x(n) + E\{v(n+1)\}$$

Since $E\{v(n+1)\} = 0$ then $\hat{x}(n+1) = \alpha x(n)$ as before.

The mean-square linear prediction error is

$$\xi_{\min} = r_x(0) - w(0)r_x(1) - w(1)r_x(2) = 1 - \alpha^2$$

Note that as α increases, the correlation between successive samples of $x(n)$ increases, and the mean-square prediction error decreases. For an uncorrelated process, $\alpha = 0$ and

$$\xi_{\min} = 1$$

which is equal to the variance of $x(n)$, and the optimum predictor is

$$\hat{x}(n+1) = 0$$

i.e., the mean value of the process.

Thus far, we have assumed that $x(n)$ is measured in the absence of noise. Unfortunately, this is typically not the case. A more realistic model for linear prediction is the one shown in Fig. 7.4 in which the signal that is to be predicted is measured in the presence of noise. With the input to the Wiener filter given by

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

the goal is to design a filter that will estimate $x(n+1)$ in terms of a linear combination of p previous values of $y(n)$

$$\hat{x}(n+1) = \sum_{k=0}^{p-1} w(k)y(n-k) = \sum_{k=0}^{p-1} w(k)\big[x(n-k) + v(n-k)\big]$$

The Wiener-Hopf equations are

$$\mathbf{R}_y\mathbf{w} = \mathbf{r}_{dy}$$

If the noise $v(n)$ is uncorrelated with the signal $x(n)$, then $\mathbf{R}_y$, the autocorrelation matrix for $y(n)$, is

$$r_y(k) = E\{y(n)y^*(n-k)\} = r_x(k) + r_v(k)$$

and $\mathbf{r}_{dy}$, the vector of cross-correlations between $d(n)$ and $y(n)$, is

$$r_{dy}(k) = E\{d(n)y^*(n-k)\} = E\{x(n+1)y^*(n-k)\} = r_x(k+1)$$

Thus, the only difference between linear prediction with and without noise is in the autocorrelation matrix for the input signal where, in the case of noise that is uncorrelated with $x(n)$, $\mathbf{R}_x$ is replaced with $\mathbf{R}_y = \mathbf{R}_x + \mathbf{R}_v$.

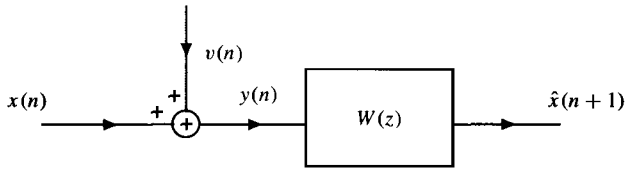


Figure 7.4 *Wiener prediction in a noisy environment.*

Example 7.2.3 *Linear Prediction in Noise*

Let us reconsider the linear prediction problem in Example 7.2.2 when the measurement of $x(n)$ is noisy. Suppose that

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

where $v(n)$ is zero mean white noise with a variance of σ_v^2. Assuming that $v(n)$ is uncorrelated with $x(n)$, the Wiener-Hopf equations are

$$\left[\mathbf{R}_x + \sigma_v^2\mathbf{I}\right]\mathbf{w} = \mathbf{r}_{dy}$$

where

$$r_{dy}(k) = r_x(k+1)$$

If $x(n)$ is an AR(1) process with an autocorrelation sequence

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

for a first-order predictor

$$W(z) = w(0) + w(1)z^{-1}$$

the Wiener-Hopf equations are

$$\begin{bmatrix} 1+\sigma_v^2 & \alpha \\ \alpha & 1+\sigma_v^2 \end{bmatrix}\begin{bmatrix} w(0) \\ w(1) \end{bmatrix} = \begin{bmatrix} \alpha \\ \alpha^2 \end{bmatrix}$$

Solving for the predictor coefficients, we find

$$\begin{bmatrix} w(0) \\ w(1) \end{bmatrix} = \frac{\alpha}{(1+\sigma_v^2)^2 - \alpha^2}\begin{bmatrix} 1+\sigma_v^2-\alpha^2 \\ \alpha\sigma_v^2 \end{bmatrix} \tag{7.18}$$

Note that as $\sigma_v^2 \to 0$ the predictor coefficients approach the solution in the noise-free case, i.e., $\mathbf{w} = [\alpha, 0]^T$.

The previous two examples considered the problem of predicting $x(n+1)$ in terms of a linear combination of the current and previous values of $x(n)$. This problem, sometimes referred to as *one-step linear prediction*, may be generalized to the problem of *multistep prediction* shown in Fig. 7.5 in which $x(n+\alpha)$ is to be predicted in terms of a linear combination of the p values $x(n), x(n-1), \ldots, x(n-p+1)$,

$$\hat{x}(n+\alpha) = \sum_{k=0}^{p-1} w(k)x(n-k)$$

where α is a positive integer. In setting up the Wiener-Hopf equations for the multistep predictor, the only term that changes from the one-step linear predictor is the cross-correlation vector $\mathbf{r}_{dx}$. In multistep prediction, since $d(n) = x(n+\alpha)$, then

$$r_{dx}(k) = E\left\{x(n+\alpha)x^*(n-k)\right\} = r_x(\alpha+k)$$

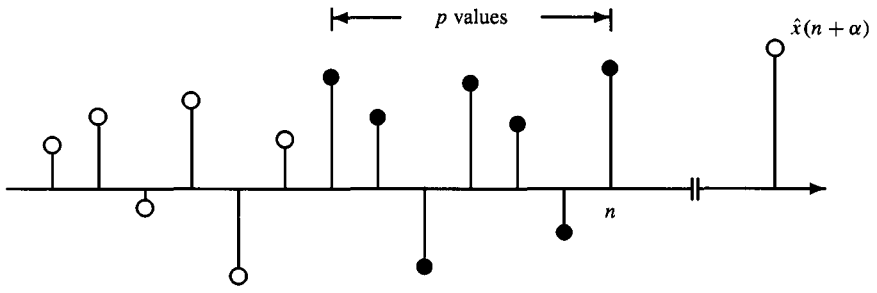


Figure 7.5 *Multistep linear prediction. Based on a linear combination of the p values* $x(n), x(n-1), \ldots, x(n-p+1)$*, the minimum mean-square estimate of* $x(n+\alpha)$ *is to be found.*

and the Wiener-Hopf equations become

$$\begin{bmatrix} r_x(0) & r_x^*(1) & r_x^*(2) & \cdots & r_x^*(p-1) \\ r_x(1) & r_x(0) & r_x^*(1) & \cdots & r_x^*(p-2) \\ r_x(2) & r_x(1) & r_x(0) & \cdots & r_x^*(p-3) \\ \vdots & \vdots & \vdots & & \vdots \\ r_x(p-1) & r_x(p-2) & r_x(p-3) & \cdots & r_x(0) \end{bmatrix} \begin{bmatrix} w(0) \\ w(1) \\ w(2) \\ \vdots \\ w(p-1) \end{bmatrix} = \begin{bmatrix} r_x(\alpha) \\ r_x(\alpha+1) \\ r_x(\alpha+2) \\ \vdots \\ r_x(\alpha+p-1) \end{bmatrix} \tag{7.19}$$

Equation (7.19) may be written in matrix form as

$$\mathbf{R}_x \mathbf{w} = \mathbf{r}_\alpha$$

where $\mathbf{r}_\alpha$ is the vector of autocorrelations beginning with $r_x(\alpha)$. Finally, for the minimum mean-square error it follows that

$$\xi_{\min} = r_x(0) - \sum_{k=0}^{p-1} w(k) r_x^*(k+\alpha) = r_x(0) - \mathbf{r}_\alpha^H \mathbf{w}$$

We now look at an example of multistep linear prediction.

Example 7.2.4 ***Multistep Linear Prediction***

Let us consider the problem of multistep prediction for a random process having an autocorrelation sequence of the form

$$r_x(k) = \delta(k) + (0.9)^{|k|} \cos(\pi k/4)$$

Thus, the first eight autocorrelation values are

$$\mathbf{r}_x = [2.0,\ 0.6364,\ 0,\ -0.5155,\ -0.6561,\ -0.4175,\ 0,\ 0.3382]^T$$

We will begin with a first-order one-step linear predictor, which is the solution to the following set of linear equations,

$$\begin{bmatrix} r_x(0) & r_x(1) \\ r_x(1) & r_x(0) \end{bmatrix} \begin{bmatrix} w(0) \\ w(1) \end{bmatrix} = \begin{bmatrix} r_x(1) \\ r_x(2) \end{bmatrix}$$

Using the given autocorrelation values and solving for the predictor coefficients, we find

$$\begin{bmatrix} w(0) \\ w(1) \end{bmatrix} = \begin{bmatrix} 0.3540 \\ -0.1127 \end{bmatrix}$$

Thus, the optimum one-step predictor is

$$\hat{x}(n+1) = 0.3540x(n) - 0.1127x(n-1)$$

and the minimum mean-square error, which we will denote by $\{\xi_1\}_{\min}$, is

$$\{\xi_1\}_{\min} = r_x(0) - w(0)r_x(1) - w(1)r_x(2) = 1.7747$$

Now, let us consider a three-step predictor. In this case, the Wiener-Hopf equations become

$$\begin{bmatrix} r_x(0) & r_x(1) \\ r_x(1) & r_x(0) \end{bmatrix} \begin{bmatrix} w(0) \\ w(1) \end{bmatrix} = \begin{bmatrix} r_x(3) \\ r_x(4) \end{bmatrix}$$

which leads to the following set of predictor coefficients,

$$\begin{bmatrix} w(0) \\ w(1) \end{bmatrix} = \begin{bmatrix} -0.1706 \\ -0.2738 \end{bmatrix}$$

i.e.,

$$\hat{x}(n+3) = -0.1706x(n) - 0.2738x(n-1)$$

For the mean-square error, $\{\xi_3\}_{\min}$, we have

$$\{\xi_3\}_{\min} = r_x(0) - w(0)r_x(3) - w(1)r_x(4) = 1.7324$$

which is smaller than the mean-square prediction error using a one-step predictor. Therefore, we have an interesting case in which it is easier to predict three samples ahead than it is to predict the next sample. Although perhaps at first surprising, if we compare the values of the autocorrelation sequence at lags $k = 1, 2$ (used in the one-step predictor), with the values at lags $k = 3, 4$ (used in the three-step predictor), we find that there is a higher degree of correlation between $x(n+3)$ and the two signal values that are used in the predictor, $x(n)$ and $x(n-1)$, than there is between $x(n+1)$ and the same two signal values. It is for this reason that the minimum mean-square error is smaller for the three-step predictor than it is for the one-step predictor.

Another way to look at the multistep predictor is in terms of a one-step predictor that uses a linear combination of the values of $x(n)$ over the interval extending from $(n-\alpha-p+2)$ to $(n-\alpha+1)$. Thus, as illustrated in Fig. 7.6, the multistep predictor may be expressed in the form

$$\hat{x}(n+1) = \sum_{k=0}^{p-1} w(k)x(n-k-\alpha+1) \tag{7.20}$$

This interpretation also suggests the following interesting variation to the linear prediction problem. With a linear predictor of the form given in Eq. (7.20), suppose that the delay α is a free parameter and that the problem is to find the coefficients, $w(k)$, along with the delay α that minimize the mean-square prediction error. This design problem, summarized in Fig. 7.7, is considered in the following example.

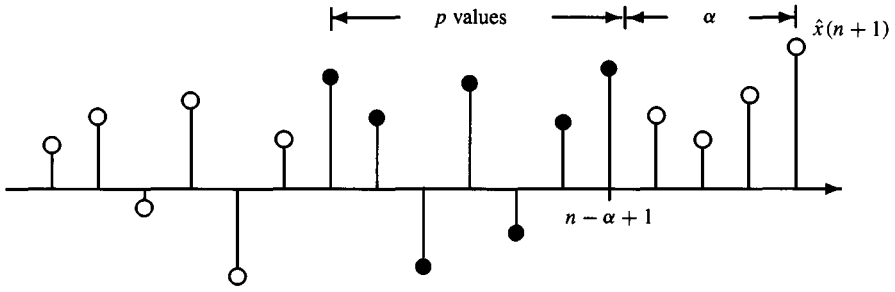


Figure 7.6 *Another interpretation of the multistep linear predictor.*

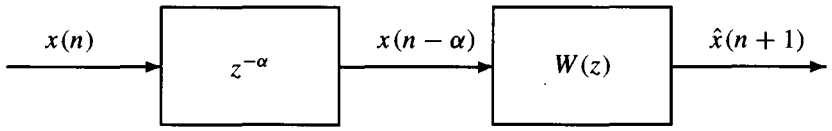


Figure 7.7 *Linear prediction with a delay.*

Example 7.2.5 *Linear Prediction with the Optimum Delay*

Suppose we reconsider Example 7.2.4 and allow the delay α to be a free variable. To design the optimum linear predictor it is necessary to find the value for α that gives the smallest mean-square prediction error. This search may be simplified by writing the mean-square error in the form

$$\{\xi_\alpha\}_{\min} = r_x(0) - \mathbf{r}_\alpha^H \mathbf{w}$$

and using the fact that the vector of predictor coefficients may be written as

$$\mathbf{w} = \mathbf{R}_x^{-1}\mathbf{r}_\alpha$$

Therefore, the minimum mean-square error is given by

$$\{\xi_\alpha\}_{\min} = r_x(0) - \mathbf{r}_\alpha^H \mathbf{R}_x^{-1}\mathbf{r}_\alpha$$

and, by precomputing $\mathbf{R}_x^{-1}$, the errors may be easily evaluated. Using this approach, the first six values of $\{\xi_\alpha\}_{\min}$ are found to be

$$\begin{aligned}
\{\xi_1\}_{\min} &= 1.7747 \\
\{\xi_2\}_{\min} &= 1.8522 \\
\{\xi_3\}_{\min} &= 1.7324 \\
\{\xi_4\}_{\min} &= 1.7605 \\
\{\xi_5\}_{\min} &= 1.9030 \\
\{\xi_6\}_{\min} &= 1.9364
\end{aligned}$$

Thus, for values of α between 1 and 6, $\alpha = 3$ gives the minimum mean-square error (this also turns out to be the global minimum over all values of α).

7.2.3 Noise Cancellation

Another application of Wiener filtering is the problem referred to as *noise cancellation*. As in the filtering problem, the goal of a noise canceller is to estimate a signal $d(n)$ from a noise corrupted observation

$$x(n) = d(n) + v_1(n)$$

that is recorded by a primary sensor. However, unlike the filtering problem, which requires that the autocorrelation of the noise be known, with a noise canceller this information is obtained from a secondary sensor that is placed within the noise field as illustrated in Fig. 7.8. Although the noise measured by this secondary sensor, $v_2(n)$, will be *correlated* with the noise in the primary sensor, the two processes will not be equal. There may be a number of reasons for this, such as differences in the sensor characteristics, differences in the propagation paths from the noise source to the two sensors, and leakage of the signal $d(n)$ into the measurements made by the secondary sensor. Since $v_1(n) \neq v_2(n)$ it is not possible to estimate $d(n)$ by simply subtracting $v_2(n)$ from $x(n)$. Instead, the noise canceller consists of a Wiener filter that is designed to estimate the noise $v_1(n)$ from the signal received by the secondary sensor. This estimate, $\hat{v}_1(n)$, is then subtracted from the primary signal $x(n)$, to form an estimate of $d(n)$, which is given by

$$\hat{d}(n) = x(n) - \hat{v}_1(n)$$

An example of where such a system may be useful is in air-to-air communications between pilots in fighter aircraft or in air-to-ground communications between a pilot and the control tower. Since there is often a large amount of engine and wind noise within the cockpit of the fighter aircraft, communication is often a difficult problem. However, if a secondary microphone is placed within the cockpit of an aircraft, then one may estimate the noise that is transmitted when the pilot speaks into the microphone, and subtract this estimate from the transmitted signal, thereby increasing the signal-to-noise ratio.

The Wiener-Hopf equations for the noise cancellation system in Fig. 7.8 may be derived as follows. With $v_2(n)$ the input to the Wiener filter that is used to estimate the noise $v_1(n)$, the Wiener-Hopf equations are

$$\mathbf{R}_{v_2}\mathbf{w} = \mathbf{r}_{v_1 v_2}$$

where $\mathbf{R}_{v_2}$ is the autocorrelation matrix of $v_2(n)$ and $\mathbf{r}_{v_1 v_2}$ is the vector of cross-correlations between the desired signal $v_1(n)$ and Wiener filter input, $v_2(n)$. For the cross-correlation

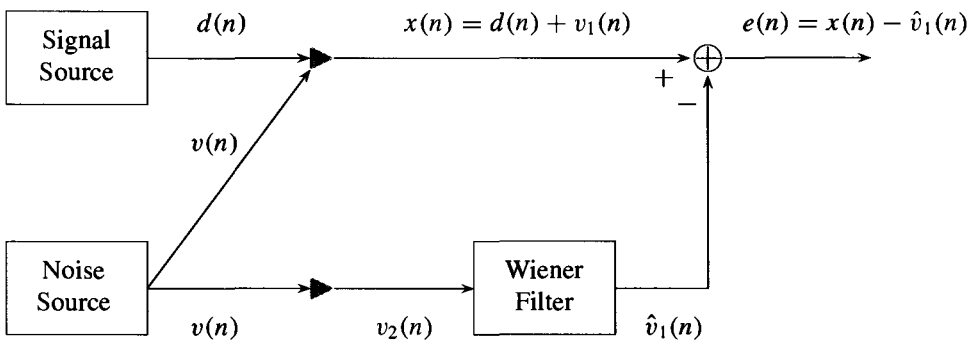


Figure 7.8 *Wiener noise cancellation using a secondary sensor to measure the additive noise* $v_1(n)$.

between $v_1(n)$ and $v_2(n)$ we have

$$r_{v_1v_2}(k) = E\{v_1(n)v_2^*(n-k)\} = E\{[x(n)-d(n)]v_2^*(n-k)\}$$

$$= E\{x(n)v_2^*(n-k)\} - E\{d(n)v_2^*(n-k)\} \tag{7.21}$$

If we assume that $v_2(n)$ is uncorrelated with $d(n)$, then the second term is zero and the cross-correlation becomes

$$r_{v_1v_2}(k) = E\{x(n)v_2^*(n-k)\} = r_{xv_2}(k) \tag{7.22}$$

Therefore, the Wiener-Hopf equations are

$$\mathbf{R}_{v_2}\mathbf{w} = \mathbf{r}_{xv_2} \tag{7.23}$$

We will now look at a specific example.

Example 7.2.6 *Noise Cancellation*

Suppose that the desired signal $d(n)$ in Fig. 7.8 is a sinusoid

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

with $\omega_0 = 0.05\pi$, and that the noise sequences $v_1(n)$ and $v_2(n)$ are AR(1) processes that are generated by the first-order difference equations

$$v_1(n) = 0.8v_1(n-1) + g(n)$$

$$v_2(n) = -0.6v_2(n-1) + g(n)$$

where $g(n)$ is zero-mean, unit variance white noise that is uncorrelated with $d(n)$. Shown in Fig. 7.9*a* is a plot of 200 samples of $x(n) = d(n) + v_1(n)$ with the desired signal, $d(n)$, indicated by the dashed line, and shown in Fig. 7.9*b* is the reference signal $v_2(n)$ that is used to estimate $v_1(n)$. Estimating $r_{v_2}(k)$ using the sample autocorrelation

$$\hat{r}_{v_2}(k) = \frac{1}{N}\sum_{n=0}^{N-1} v_2(n)v_2(n-k)$$

and $r_{xv_2}(k)$ using the sample cross-correlation

$$\hat{r}_{xv_2}(k) = \frac{1}{N}\sum_{n=0}^{N-1} x(n)v_2(n-k)$$

FIR Wiener filters of orders $p = 6$ and $p = 12$ were found by solving Eq. (7.23). Using these filters to estimate $v_1(n)$, the sinusoid $d(n)$ was then estimated by subtracting $\hat{v}_1(n)$ from $x(n)$. The results are shown in Figs. 7.9*c* and *d*.

In typical applications, $d(n)$ and $v_1(n)$ are often found to be non-stationary processes. Therefore, the use of a linear shift-invariant Wiener filter will not be optimum. However, as we will see in Chapter 9, an adaptive Wiener filter that has filter coefficients that are allowed to vary as a function of time may provide effective noise cancellation in nonstationary environments.

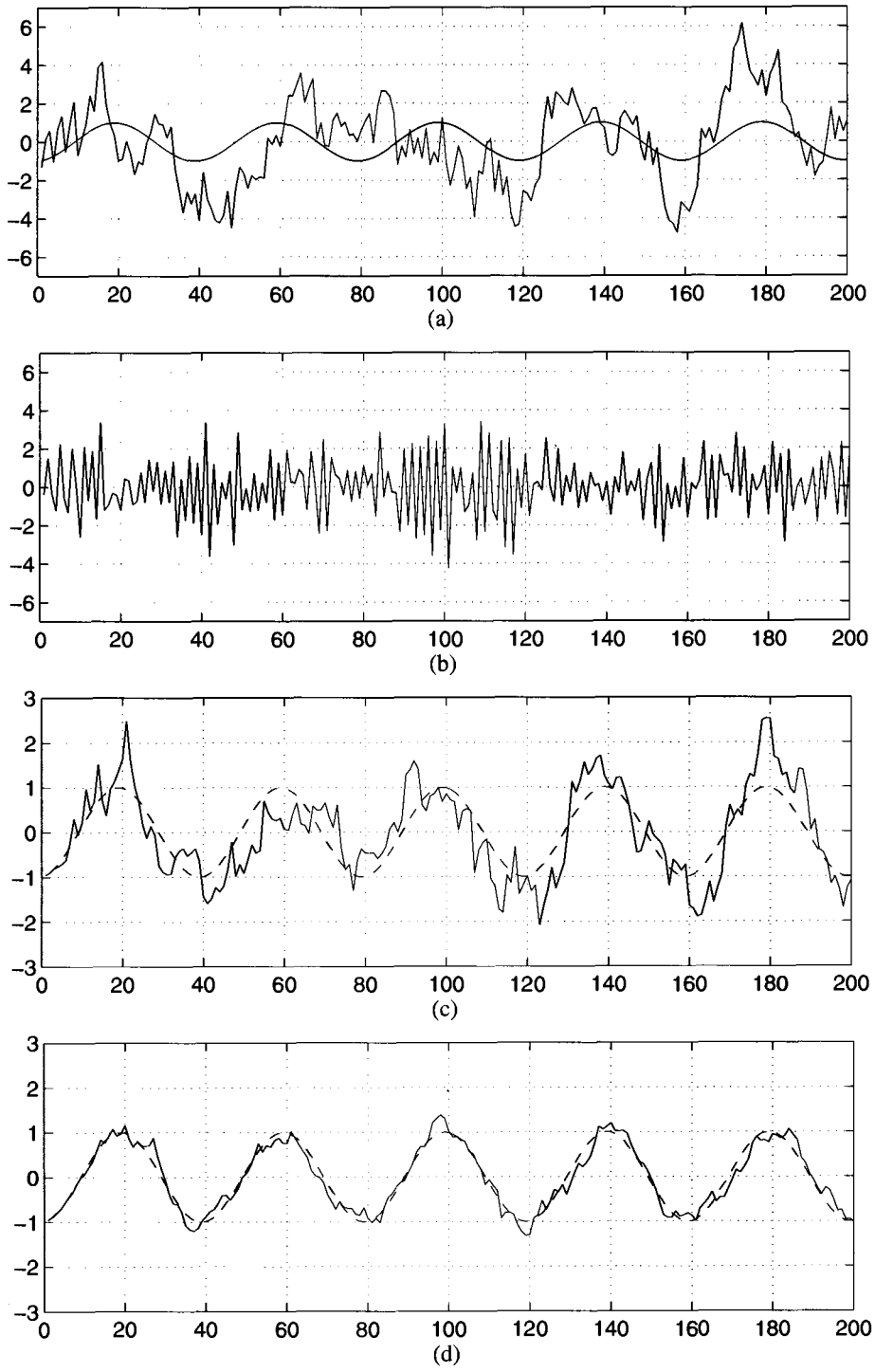


Figure 7.9 *Noise cancellation example. (a) Noise corrupted sinusoid, (b) Reference signal used by secondary sensor, (c) Output of sixth-order Wiener noise canceller, (d) Output of twelth-order Wiener noise canceller.*

7.2.4 Lattice Representation for the FIR Wiener Filter

In this section, we derive a lattice filter implementation for an FIR Wiener filter. We begin by noting that the output of the Wiener filter may be written in vector form as follows:

$$\hat{d}(n) = \mathbf{x}^T(n)\mathbf{w}$$

where

$$\mathbf{x}(n) = \left[x(n),\ x(n-1),\ \ldots,\ x(n-p)\right]^T$$

and where $\mathbf{w}$ is the solution to the Wiener-Hopf equations given in Eq. (7.9). Using the *triangular decomposition* for the inverse of $\mathbf{R}_x$ given by Eq. (5.102)

$$\mathbf{R}_x^{-1} = \mathbf{A}_{p-1}\mathbf{D}_{p-1}^{-1}\mathbf{A}_{p-1}^H$$

where $\mathbf{D}_{p-1} = \text{diag}\{\epsilon_0, \epsilon_1, \ldots, \epsilon_{p-1}\}$ is a diagonal matrix of forward prediction errors and where

$$\mathbf{A}_{p-1} = \begin{bmatrix} 1 & a_1^*(1) & a_2^*(2) & \cdots & a_{p-1}^*(p-1) \\ 0 & 1 & a_2^*(1) & \cdots & a_{p-1}^*(p-2) \\ 0 & 0 & 1 & \cdots & a_{p-1}^*(p-3) \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix} \tag{7.24}$$

is the matrix of prediction error filters as defined in Eq. (5.84). Therefore, the output of the Wiener filter may be written as

$$\hat{d}(n) = \mathbf{x}^T(n)\mathbf{R}_x^{-1}\mathbf{r}_{dx} = \mathbf{x}^T(n)\mathbf{A}_{p-1}\mathbf{D}_{p-1}^{-1}\mathbf{A}_{p-1}^H\mathbf{r}_{dx} \tag{7.25}$$

Recall, however, from Eq. (6.101) that

$$\mathbf{A}_{p-1}^T\mathbf{x}(n) = \mathbf{e}^-(n)$$

where

$$\mathbf{e}^-(n) = \begin{bmatrix} e_0^-(n) \\ e_1^-(n) \\ \vdots \\ e_{p-1}^-(n) \end{bmatrix}$$

is the vector of backward prediction errors at time n. With

$$\boxed{\mathbf{b} = \mathbf{D}_{p-1}^{-1}\mathbf{A}_{p-1}^H\mathbf{r}_{dx}} \tag{7.26}$$

it follows therefore, that $\hat{d}(n) = \mathbf{b}^T\mathbf{e}^-(n)$, i.e.,

$$\hat{d}(n) = \sum_{k=0}^{p-1} b(k)e_k^-(n) \tag{7.27}$$

Therefore, the Wiener estimate $\hat{d}(n)$ is formed by taking a linear combination of the backward prediction errors using the filter coefficients $b(k)$, as shown in Fig. 7.10. This Wiener lattice filter, sometimes called *joint process estimator*, may be viewed as two filters operating jointly. The first is a *lattice predictor* that transforms the sequence of correlated

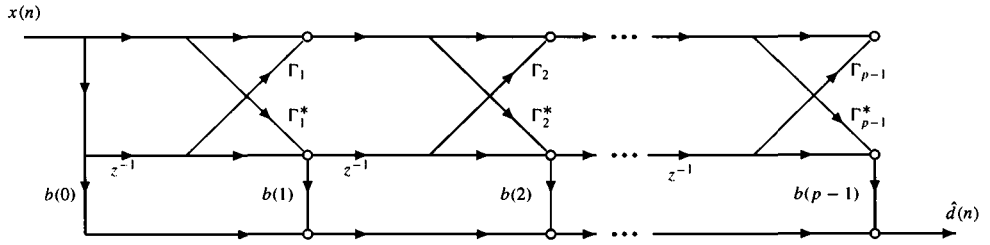


Figure 7.10 *A lattice filter implementation of an FIR Wiener filter.*

input samples, $x(n), x(n-1), \ldots, x(n-p)$, into a sequence of uncorrelated backward prediction errors, $e_0^-(n), e_1^-(n), \ldots, e_p^-(n)$ (see the discussion leading up to Eq. (6.102) in Section 6.6 in Chapter 6). The second filter, called a *multiple regression filter* [5] which is characterized by the weights $b(k)$, uses the backward prediction errors $e_k^-(n)$ as inputs to produce an estimate of the desired signal $d(n)$.

7.3 THE IIR WIENER FILTER

Having solved the FIR Wiener filter design problem and having explored a few Wiener filtering applications, we now consider the design of an IIR digital Wiener filter. As was the case for an FIR Wiener filter, given a process $x(n)$ our goal is to design a filter $h(n)$ that produces an output $y(n) = x(n) * h(n)$ that is as close as possible in the mean-square sense to a desired process, $d(n)$. Although the problem formulation is the same for both the FIR and the IIR Wiener filters, there is an important difference that significantly changes the solution. Specifically, for the FIR Wiener filter there are only a finite number of filter coefficients that must be determined, whereas for the IIR Wiener filter there are an infinite number of unknowns, i.e., the values of $h(n)$ for all n. In this section, we will consider two cases. First, in Section 7.3.1, we will solve the Wiener filter design problem for the case in which there are no constraints placed on the solution. What we will find is that the optimum filter will, in general, be noncausal and therefore unrealizable. Then, in Section 7.3.2, we will constrain the solution to be causal by forcing $h(n)$ to be zero for $n \leq 0$. For the noncausal Wiener filter, we will find a simple closed form expression for the frequency response, whereas for the causal Wiener filter, we will only be able to specify the system function implicitly in terms of a spectral factorization.

7.3.1 Noncausal IIR Wiener Filter

For a noncausal (unconstrained) IIR Wiener filter, the problem is to find the unit sample response, $h(n)$, of the IIR filter

$$H(z) = \sum_{n=-\infty}^{\infty} h(n) z^{-n}$$

that minimizes the mean-square error

$$\xi = E\{|e(n)|^2\} \tag{7.28}$$

where $e(n)$ is the difference between the desired process $d(n)$ and the output of the Wiener filter, $\hat{d}(n)$,

$$e(n) = d(n) - \hat{d}(n) = d(n) - \sum_{l=-\infty}^{\infty} h(l)x(n-l) \tag{7.29}$$

This problem may be solved in exactly the same way that we solved the FIR Wiener filtering problem, i.e., by differentiating ξ with respect to $h^*(k)$ for each k and setting the derivatives equal to zero. Performing this differentiation we have

$$\frac{\partial \xi}{\partial h^*(k)} = -E\{e(n)x^*(n-k)\} = 0 \quad ; \quad -\infty < k < \infty$$

or

$$\boxed{E\{e(n)x^*(n-k)\} = 0 \quad ; \quad -\infty < k < \infty} \tag{7.30}$$

Equation (7.30) is referred to as the *orthogonality principle*, and it is identical to the orthogonality principle for the FIR Wiener filter given in Eq. (7.5) except that here the equality must hold for all k. Substituting Eq. (7.29) for $e(n)$ into Eq. (7.30) and rearranging terms gives

$$\sum_{l=-\infty}^{\infty} h(l)E\{x(n-l)x^*(n-k)\} = E\{d(n)x^*(n-k)\} \quad ; \quad -\infty < k < \infty \tag{7.31}$$

Note that the expectation on the left is the autocorrelation of $x(n)$, and the expectation on the right is the cross-correlation between $x(n)$ and $d(n)$. Therefore, Eq. (7.31) may be written as

$$\boxed{\sum_{l=-\infty}^{\infty} h(l)r_x(k-l) = r_{dx}(k) \quad ; \quad -\infty < k < \infty} \tag{7.32}$$

which are the Wiener-Hopf equations of the noncausal IIR Wiener filter. Comparing the Wiener-Hopf equations for the FIR Wiener filter and the noncausal IIR Wiener filter, Eqs. (7.7) and (7.32), respectively, we see that the only difference is in the limits on the summation and the range of values for which the equations must hold. Although Eq. (7.32) corresponds to an infinite set of linear equations with an infinite number of unknowns, finding the solution to these equations is straightforward if we write the left side of the equation as the convolution of $h(k)$ with $r_x(k)$,

$$h(k) * r_x(k) = r_{dx}(k) \tag{7.33}$$

In the frequency domain, Eq. (7.33) becomes

$$H(e^{j\omega})P_x(e^{j\omega}) = P_{dx}(e^{j\omega}) \tag{7.34}$$

Therefore, it follows that the frequency response of the IIR Wiener filter is

$$\boxed{H(e^{j\omega}) = \frac{P_{dx}(e^{j\omega})}{P_x(e^{j\omega})}} \tag{7.35}$$

and the system function is

$$H(z) = \frac{P_{dx}(z)}{P_x(z)} \tag{7.36}$$

Note that the denominator of $H(z)$ is a power spectral density, $P_x(z)$, and that the numerator, $P_{dx}(z)$, is a cross-power spectral density.

Having found the noncausal IIR Wiener filter, we now evaluate the mean-square error. Following the steps that were taken for an FIR Wiener filter, we find that the mean-square error is

$$\xi_{\min} = r_d(0) - \sum_{l=-\infty}^{\infty} h(l) r_{dx}^*(l) \tag{7.37}$$

Note that the only difference between this and the mean-square error for the FIR Wiener filter given in Eq. (7.11) is in the limits on the summation. Using Parseval's theorem this error may be expressed in the frequency domain as follows

$$\boxed{\xi_{\min} = r_d(0) - \frac{1}{2\pi} \int_{-\pi}^{\pi} H(e^{j\omega}) P_{dx}^*(e^{j\omega}) d\omega} \tag{7.38}$$

Since

$$r_d(0) = \frac{1}{2\pi} \int_{-\pi}^{\pi} P_d(e^{j\omega}) d\omega$$

then we also have

$$\xi_{\min} = \frac{1}{2\pi} \int_{-\pi}^{\pi} \left[P_d(e^{j\omega}) - H(e^{j\omega}) P_{dx}^*(e^{j\omega}) \right] d\omega \tag{7.39}$$

This error may also be expressed in terms of the complex variable z as follows:

$$\xi_{\min} = r_d(0) - \frac{1}{2\pi j} \oint_C H(z) P_{dx}^*(1/z^*) z^{-1} dz \tag{7.40}$$

or, equivalently,

$$\xi_{\min} = \frac{1}{2\pi j} \oint_C \left[P_d(z) - H(z) P_{dx}^*(1/z^*) \right] z^{-1} dz \tag{7.41}$$

where the contour, C, may be taken to be the unit circle. The noncausal Wiener filtering equations are summarized in Table 7.2.

We conclude this section with a derivation of the Wiener smoothing filter for producing the minimum mean-square estimate of a process $d(n)$ using the noisy observations

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

for all n. Since the optimum noncausal IIR Wiener filter is given by Eq. (7.35), all that must be done is to find $P_x(e^{j\omega})$ and $P_{dx}(e^{j\omega})$. Assuming that $d(n)$ and $v(n)$ are uncorrelated zero mean random processes, the autocorrelation of $x(n)$ is

$$r_x(k) = r_d(k) + r_v(k)$$

and the power spectrum is

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

Table 7.2 *The Frequency Response and Minimum Error for a Noncausal Wiener Filter*

Wiener-Hopf equations

$$H(e^{j\omega}) = \frac{P_{dx}(e^{j\omega})}{P_x(e^{j\omega})}$$

Correlations

$$r_x(k) = E\{x(n)x^*(n-k)\}$$
$$r_{dx}(k) = E\{d(n)x^*(n-k)\}$$

Minimum error

$$\xi_{\min} = r_d(0) - \sum_{l=-\infty}^{\infty} h(l)r_{dx}^*(l)$$
$$= \frac{1}{2\pi}\int_{-\pi}^{\pi}\left[P_d(e^{j\omega}) - H(e^{j\omega})P_{dx}^*(e^{j\omega})\right]d\omega$$

Furthermore, the cross-correlation, $r_{dx}(k)$, is

$$r_{dx}(k) = E\{d(n)x^*(n-k)\} = E\{d(n)d^*(n-k)\} + E\{d(n)v^*(n-k)\} = r_d(k)$$

Therefore,

$$P_{dx}(e^{j\omega}) = P_d(e^{j\omega})$$

and the IIR Wiener smoothing filter is

$$H(e^{j\omega}) = \frac{P_d(e^{j\omega})}{P_d(e^{j\omega}) + P_v(e^{j\omega})} \tag{7.42}$$

Note that for those values of ω for which $P_d(e^{j\omega}) \gg P_v(e^{j\omega})$, the signal-to-noise ratio is high and $|H(e^{j\omega})| \approx 1$. Therefore, over those frequency bands where the signal dominates, the filter passes the process with little attenuation. On the other hand, for those values of ω for which the signal-to-noise ratio is small, $P_d(e^{j\omega}) \ll P_v(e^{j\omega})$, the frequency response is small, $|H(e^{j\omega})| \approx 0$. Therefore, over those frequency bands where the noise dominates, $H(e^{j\omega})$ is small in order to filter out or suppress the noise. Finally, since

$$P_{dx}(e^{j\omega}) = P_d(e^{j\omega})$$

if we evaluate the mean-square error using Eq. (7.39) and use the fact that $P_d(e^{j\omega})$ is real we have

$$\xi_{\min} = \frac{1}{2\pi}\int_{-\pi}^{\pi}\left[P_d(e^{j\omega}) - H(e^{j\omega})P_{dx}^*(e^{j\omega})\right]d\omega = \frac{1}{2\pi}\int_{-\pi}^{\pi}P_d(e^{j\omega})\left[1 - H(e^{j\omega})\right]d\omega$$

Substituting Eq. (7.42) for the frequency response of the Wiener smoothing filter we find that the minimum mean-square error is

$$\xi_{\min} = \frac{1}{2\pi}\int_{-\pi}^{\pi}P_d(e^{j\omega})\frac{P_v(e^{j\omega})}{P_d(e^{j\omega}) + P_v(e^{j\omega})}d\omega = \frac{1}{2\pi}\int_{-\pi}^{\pi}P_v(e^{j\omega})H(e^{j\omega})d\omega \tag{7.43}$$

which, if expressed in the z-domain, becomes

$$\xi_{\min} = \frac{1}{2\pi j}\oint_C P_v(z)H(z)z^{-1}dz \tag{7.44}$$

Example 7.3.1 *Noncausal Wiener Smoothing*

Let $d(n)$ be a real-valued AR(1) process with power spectrum

$$P_d(z) = \frac{b^2(0)}{(1-\alpha z^{-1})(1-\alpha z)}$$

and suppose that $d(n)$ is observed in the presence of zero mean white noise with a variance σ_v^2,

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

Assuming that $v(n)$ is uncorrelated with $d(n)$, we will design a noncausal IIR Wiener smoothing filter for estimating $d(n)$ from $x(n)$ and find the mean-square error in the estimation of $d(n)$.

From Eq. (7.42) it follows that the system function of the noncausal Wiener smoothing filter is

$$H(z) = \frac{P_d(z)}{P_d(z) + P_v(z)}$$

Substituting the given expression for $P_d(z)$ into $H(z)$ and setting $P_v(z) = \sigma_v^2$, we have

$$H(z) = \frac{b^2(0)}{b^2(0) + \sigma_v^2(1-\alpha z^{-1})(1-\alpha z)}$$

Evaluating the minimum error using Eq. (7.43), with $P_v(z) = \sigma_v^2$ we have the remarkably simple result

$$\xi_{\min} = \frac{1}{2\pi}\int_{-\pi}^{\pi} P_v(e^{j\omega})H(e^{j\omega})d\omega = \sigma_v^2 \frac{1}{2\pi}\int_{-\pi}^{\pi} H(e^{j\omega})\,d\omega = \sigma_v^2 h(0)$$

Now consider the specific case in which $b^2(0) = 0.25$, $\alpha = 0.5$, and $\sigma_v^2 = 0.25$. For the Wiener filter we have

$$H(z) = \frac{0.25}{0.25 + 0.25(1-0.5z^{-1})(1-0.5z)} = \frac{2(0.2344)}{(1-0.2344z^{-1})(1-0.2344z)}$$

Using the z-transform pair

$$\alpha^{|n|} \longleftrightarrow \frac{1-\alpha^2}{(1-\alpha z^{-1})(1-\alpha z)}$$

we have for the unit sample response,

$$h(n) = 0.4960\,(0.2344)^{|n|}$$

which, clearly, is noncausal. For the minimum mean-square error we have

$$\xi_{\min} = \sigma_v^2 h(0) = (0.25)(0.4960) = 0.1240$$

Finally, it is interesting to see how much the error is reduced as a result of filtering $x(n)$ with a Wiener filter. Without a filter, setting $\hat{d}(n) = x(n)$ the mean-square error is

$$E\left\{|e(n)|^2\right\} = E\left\{|v(n)|^2\right\} = 0.25$$

Thus, the noncausal Wiener filter reduces the mean-square error by approximately a factor of two.

7.3.2 The Causal IIR Wiener Filter

In the previous section, we considered the design of an IIR digital Wiener filter, and placed no constraints on the form of the solution. In this section, we reconsider the design for the case in which the Wiener filter is constrained to be causal. For a causal filter, the unit sample response is zero for $n < 0$ and the estimate of $d(n)$ takes the form

$$\hat{d}(n) = x(n) * h(n) = \sum_{k=0}^{\infty} h(k)x(n-k)$$

To find the filter coefficients that minimize the mean-square error, we proceed in exactly the same way that we did for the noncausal Wiener filter. Specifically, differentiating ξ with respect to $h^*(k)$ for $k \geq 0$ and setting the derivatives to zero we find

$$\boxed{\sum_{l=0}^{\infty} h(l)r_x(k-l) = r_{dx}(k) \quad ; \quad 0 \leq k < \infty} \tag{7.45}$$

which are the Wiener-Hopf equations for the causal IIR Wiener filter. Note that the only differences between Eqs. (7.32) and (7.45) are in the limits on the summation and the values of k for which the equations must hold. This restriction on k is important since it implies that $r_{dx}(k)$ may no longer be expressed as the convolution of $h(k)$ and $r_x(k)$. As a result, solving Eq. (7.45) for the coefficients $h(l)$ of the Wiener filter is much more difficult than solving Eq. (7.32).

To solve the Wiener-Hopf equations, we begin by looking at the special case in which the input to the filter is unit variance white noise, $\epsilon(n)$. Denoting the coefficients of the Wiener filter by $g(n)$, the Wiener-Hopf equations are

$$\sum_{l=0}^{\infty} g(l)r_\epsilon(k-l) = r_{d\epsilon}(k) \quad ; \quad 0 \leq k < \infty \tag{7.46}$$

With $r_\epsilon(k) = \delta(k)$, the left side of Eq. (7.46) reduces to $g(k)$. Therefore, $g(k) = r_{d\epsilon}(k)$ for $k \geq 0$ and, since the Wiener filter is causal, $g(k) = 0$ for $n < 0$. Thus, the causal Wiener filter for a white noise input $\epsilon(n)$ is

$$g(n) = r_{d\epsilon}(n)u(n) \tag{7.47}$$

where $u(n)$ is the unit step function. We will express this solution in the z-domain as follows:

$$\boxed{G(z) = [P_{d\epsilon}(z)]_+} \tag{7.48}$$

where the subscript "+" is used to indicate the "positive-time part" of the sequence whose z-transform is contained within the brackets.

In a typical Wiener filtering application, it is unlikely that the input to the Wiener filter will be white noise. Therefore, suppose that $x(n)$ is a random process with a rational power spectrum that has no poles or zeros on the unit circle. We may then perform a spectral factorization and write $P_x(z)$ as follows (see p. 105)

$$P_x(z) = \sigma_0^2 Q(z)Q^*(1/z^*) \tag{7.49}$$

where $Q(z)$ is minimum phase and of the form

$$Q(z) = 1 + q(1)z^{-1} + q(2)z^{-2} + \cdots = \frac{N(z)}{D(z)}$$

with $N(z)$ and $D(z)$ minimum phase monic polynomials. If $x(n)$ is filtered with a filter having a system function of the form (see Fig. 7.11)

$$F(z) = \frac{1}{\sigma_0 Q(z)} \tag{7.50}$$

then the power spectrum of the output process, $\epsilon(n)$, will be

$$P_\epsilon(z) = P_x(z)F(z)F^*(1/z^*) = 1$$

Therefore, $\epsilon(n)$ is white noise and $F(z)$ is referred to as a *whitening filter*. Note that since $Q(z)$ is minimum phase, then $F(z)$ is stable and causal and has a stable and causal inverse, $F^{-1}(z)$. As a result, $x(n)$ may be recovered from $\epsilon(n)$ by filtering with the inverse filter, $F^{-1}(z)$. In other words, there is no loss of information in the linear transformation that produces the white noise process from $x(n)$.

With this background, we are now in a position to derive the optimum causal Wiener filter when the input to the filter $x(n)$ has a rational power spectrum. Let $H(z)$ be the causal Wiener filter that produces the minimum mean-square estimate of $d(n)$ from $x(n)$, and suppose that $x(n)$ is filtered with a cascade of three filters, $F(z)$, $F^{-1}(z)$, and $H(z)$ as shown in Fig. 7.12, where $F(z)$ is the causal whitening filter for $x(n)$ and $F^{-1}(z)$ is the causal inverse. The cascade

$$G(z) = F^{-1}(z)H(z)$$

is the causal Wiener filter that produces the minimum mean-square estimate of $d(n)$ from the white noise process $\epsilon(n)$. The causality of $G(z)$ follows from the fact that both $F^{-1}(z)$ and $H(z)$ are causal. The optimality of the filter follows from the observation that if there were another filter, $G'(z)$, that produced an estimate of $d(n)$ having a smaller mean-square error

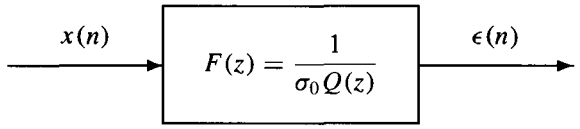


Figure 7.11 *A whitening filter that produces white noise with power spectrum $P_\epsilon(e^{j\omega}) = 1$ when the input, $x(n)$, has a power spectrum $P_x(z) = \sigma_0^2 Q(z)Q^*(1/z^*)$.*

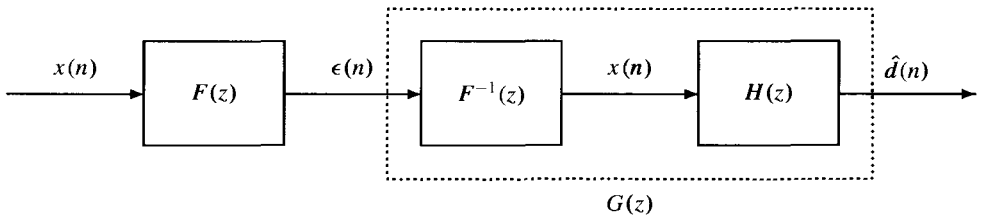


Figure 7.12 *A causal Wiener filter, $G(z)$, operating on a whitened input, $\epsilon(n)$, with $H(z)$ the causal Wiener filter for estimating $d(n)$ from $x(n)$.*

than $G(z)$, then $H'(z) = F(z)G'(z)$ would produce an estimate of $d(n)$ with a smaller mean-square error than $H(z)$, the causal Wiener filter. This, however, contradicts the assumed optimality of $H(z)$.

With $\epsilon(n)$ a white noise process, we see from Eq. (7.48) that $G(z)$, the causal IIR Wiener filter for estimating $d(n)$ from $\epsilon(n)$, is $G(z) = [P_{d\epsilon}(z)]_+$. Since $\epsilon(n)$ is formed by filtering $x(n)$ with the whitening filter $f(n)$, then the cross-correlation between $d(n)$ and $\epsilon(n)$ is

$$r_{d\epsilon}(k) = E\{d(n)\epsilon^*(n-k)\} = E\left\{d(n)\left[\sum_{l=-\infty}^{\infty} f(l)x(n-k-l)\right]^*\right\}$$

$$= \sum_{l=-\infty}^{\infty} f^*(l)r_{dx}(k+l) \tag{7.51}$$

Therefore, the cross-power spectral density, $P_{d\epsilon}(z)$, is

$$P_{d\epsilon}(z) = P_{dx}(z)F^*(1/z^*) = \frac{P_{dx}(z)}{\sigma_0 Q^*(1/z^*)}$$

and the causal Wiener filter for estimating $d(n)$ from $\epsilon(n)$ is

$$G(z) = \frac{1}{\sigma_0}\left[\frac{P_{dx}(z)}{Q^*(1/z^*)}\right]_+ \tag{7.52}$$

The design is completed by recalling that the causal Wiener filter for estimating $d(n)$ from $x(n)$ is the cascade of $F(z)$ and $G(z)$,

$$H(z) = F(z)G(z)$$

Thus, combining Eqs. (7.50) and (7.52) leads to the desired solution

$$\boxed{H(z) = \frac{1}{\sigma_0^2 Q(z)}\left[\frac{P_{dx}(z)}{Q^*(1/z^*)}\right]_+} \tag{7.53}$$

In the case of real processes, $h(n)$ is real and the causal Wiener filter takes the form

$$H(z) = \frac{1}{\sigma_0^2 Q(z)}\left[\frac{P_{dx}(z)}{Q(z^{-1})}\right]_+ \tag{7.54}$$

Finally, as with the noncausal IIR Wiener filter, the mean-square error for the causal IIR Wiener filter is

$$\xi_{\min} = r_d(0) - \sum_{l=0}^{\infty} h(l)r_{dx}^*(l) \tag{7.55}$$

where the sum now extends only over the interval $0 \le l < \infty$ since $h(l) = 0$ for $l < 0$. In the frequency domain, this error may be written as

$$\xi_{\min} = \frac{1}{2\pi}\int_{-\pi}^{\pi}\left[P_d(e^{j\omega}) - H(e^{j\omega})P_{dx}^*(e^{j\omega})\right]d\omega \tag{7.56}$$

or, equivalently,

$$\xi_{\min} = \frac{1}{2\pi j}\oint_C \left[P_d(z) - H(z)P_{dx}^*(1/z^*)\right]z^{-1}dz \tag{7.57}$$

The causal Wiener filtering equations are summarized in Table 7.3.

Table 7.3 The System Function and Minimum Error for a Causal Wiener Filter

System function	$H(z) = \frac{1}{\sigma_0^2 Q(z)} \left[\frac{P_{dx}(z)}{Q^*(1/z^*)} \right]_+$
Spectral Factorization	$P_x(z) = \sigma_0^2 Q(z) Q^*(1/z^*)$
Minimum error	$\xi_{\min} = r_d(l) - \sum_{l=0}^{\infty} h(l) r_{dx}^*(l)$ $= \frac{1}{2\pi} \int_{-\pi}^{\pi} \left[P_d(e^{j\omega}) - H(e^{j\omega}) P_{dx}^*(e^{j\omega}) \right] d\omega$

An interesting interpretation of the causal Wiener filter follows if we compare Eq. (7.53) to the noncausal Wiener filter given in Eq. (7.36). Denoting the noncausal Wiener filter by $H_{\text{nc}}(z)$ and using the spectral factorization of $P_x(z)$ given in Eq. (7.49), we see that the noncausal Wiener filter may be written as

$$H_{\text{nc}}(z) = \frac{P_{dx}(z)}{\sigma_0^2 Q(z) Q^*(1/z^*)} \tag{7.58}$$

Viewed as a cascade of two filters, the noncausal Wiener filter is

$$\boxed{H_{\text{nc}}(z) = \frac{1}{\sigma_0^2 Q(z)} \left[\frac{P_{dx}(z)}{Q^*(1/z^*)} \right]} \tag{7.59}$$

as shown in Fig. 7.13*a*. Note that the first filter is the causal whitening filter that generates the white noise process $\epsilon(n)$ from $x(n)$, and the second filter is the *noncausal* filter that produces the minimum mean-square estimate of $d(n)$ from the whitened signal. Comparing Eq. (7.59) to Eq. (7.53), note that the causal IIR Wiener filter shown in Fig. 7.13*b* is formed by taking the causal part of $[P_{dx}(z)/Q^*(1/z^*)]$. We now look at some applications of causal Wiener filtering.

7.3.3 Causal Wiener Filtering

In Section 7.3.1, we considered the problem of noncausal Wiener smoothing for estimating a process $d(n)$ from the noisy observations

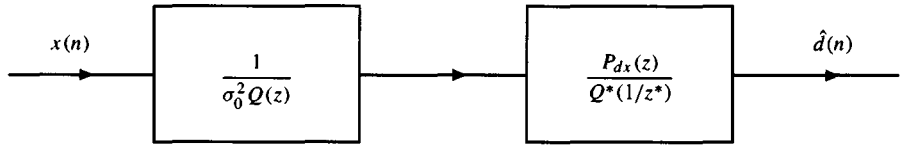
$$x(n) = d(n) + v(n)$$

The system function of the causal Wiener filter for estimating $d(n)$ is given in Eq. (7.53). If the noise $v(n)$ is uncorrelated with $d(n)$ then $P_{dx}(z) = P_d(z)$ and the causal Wiener filter becomes

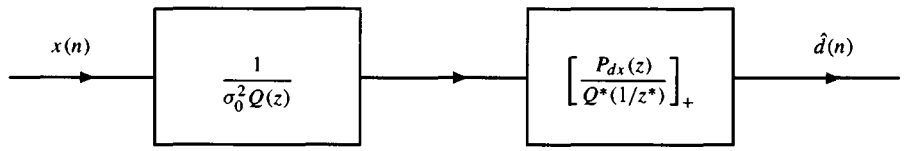
$$H(z) = \frac{1}{\sigma_0^2 Q(z)} \left[\frac{P_d(z)}{Q^*(1/z^*)} \right]_+ \tag{7.60}$$

where

$$P_x(z) = P_d(z) + P_v(z) = \sigma_0^2 Q(z) Q^*(1/z^*) \tag{7.61}$$



(a) Noncausal Wiener filter.



(b) Causal Wiener filter

Figure 7.13 *(a) A noncausal Wiener filter expressed in terms of a cascade of a causal filter with a noncausal filter. (b) The causal Wiener filter that is formed by taking the causal part of the noncausal factor in (a).*

However, without specific expressions for the power spectral densities $P_d(z)$ and $P_v(z)$, Eq. (7.60) cannot be simplified any further. In the following example we design a causal Wiener filter for estimating an AR(1) process that is measured in white noise.

Example 7.3.2 *Causal Wiener Filtering*

Suppose that we wish to estimate a signal $d(n)$ from the noisy observation

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

where $v(n)$ is unit variance white noise that is uncorrelated with $d(n)$. The signal $d(n)$ is an AR(1) process that is generated by the difference equation

$$d(n) = 0.8d(n-1) + w(n)$$

where $w(n)$ is white noise with variance $\sigma_w^2 = 0.36$. Therefore $r_d(k) = (0.8)^{|k|}$.

To find the optimum *causal* Wiener filter for estimating $d(n)$ from $x(n)$ we begin by noting that

$$\begin{aligned} P_{dx}(z) &= P_d(z) \\ P_x(z) &= P_d(z) + P_v(z) = P_d(z) + 1 \end{aligned}$$

Therefore, with

$$P_d(z) = \frac{0.36}{(1-0.8z^{-1})(1-0.8z)}$$

the power spectrum of $x(n)$ is

$$P_x(z) = 1 + \frac{0.36}{(1-0.8z^{-1})(1-0.8z)} = 1.6\frac{(1-0.5z^{-1})(1-0.5z)}{(1-0.8z^{-1})(1-0.8z)}$$

and, since $x(n)$ is real, then $P_x(z) = \sigma_0^2 Q(z)Q(z^{-1})$ with

$$\sigma_0^2 = 1.6 \quad \text{and} \quad Q(z) = \frac{(1 - 0.5z^{-1})}{(1 - 0.8z^{-1})}$$

Since the causal Wiener filter is

$$H(z) = \frac{1}{\sigma_0^2 Q(z)} \left[\frac{P_{dx}(z)}{Q(z^{-1})} \right]_+$$

then we have

$$\begin{aligned} \frac{P_{dx}(z)}{Q(z^{-1})} &= \frac{0.36}{(1 - 0.8z^{-1})(1 - 0.8z)} \frac{(1 - 0.8z)}{(1 - 0.5z)} \\ &= \frac{0.36z^{-1}}{(1 - 0.8z^{-1})(z^{-1} - 0.5)} \\ &= \frac{0.6}{1 - 0.8z^{-1}} + \frac{0.3}{z^{-1} - 0.5} \end{aligned}$$

Therefore,

$$\left[\frac{P_{dx}(z)}{Q(z^{-1})} \right]_+ = \frac{0.6}{1 - 0.8z^{-1}}$$

and the Wiener filter is

$$H(z) = \frac{1}{1.6} \frac{(1 - 0.8z^{-1})}{(1 - 0.5z^{-1})} \frac{0.6}{(1 - 0.8z^{-1})} = \frac{0.375}{1 - 0.5z^{-1}}$$

or,

$$h(n) = 0.375 \left(\tfrac{1}{2}\right)^n u(n)$$

Since $\hat{D}(z) = H(z)X(z)$, the estimate of $d(n)$ may be computed recursively as follows

$$\hat{d}(n) = 0.5\hat{d}(n - 1) + 0.375x(n)$$

Finally, for the mean-square error in the estimate of $d(n)$ we have

$$\xi_{\min} = E\left\{ \left[d(n) - \hat{d}(n) \right]^2 \right\} = r_d(0) - \sum_{l=0}^{\infty} h(l) r_{dx}(l) = 1 - \tfrac{3}{8} \sum_{l=0}^{\infty} \left(\tfrac{1}{2}\right)^l (0.8)^l = 0.3750$$

Note that, for the second-order FIR Wiener filter designed in Example 7.2.1, the mean-square error was $\xi_{\min} = 0.4048$. Therefore, using *all* of the previous observations of $x(n)$ only slightly improves the performance of the Wiener filter. For another comparison, let us find the *noncausal* Wiener filter and evaluate the mean-square error. The system function of the noncausal Wiener filter is

$$H(z) = \frac{P_{dx}(z)}{P_x(z)} = \frac{P_d(z)}{P_x(z)} = \frac{0.36/1.6}{(1 - 0.5z^{-1})(1 - 0.5z)}$$

and the unit sample response is

$$h(n) = \tfrac{3}{10}\left(\tfrac{1}{2}\right)^{|n|}$$

Computing the mean-square error we find

$$\xi_{\min} = r_d(0) - \sum_{l=-\infty}^{\infty} h(l) r_{dx}(l) = 1 - 2\sum_{k=0}^{\infty} \tfrac{3}{10}\left(\tfrac{1}{2}\right)^k (0.8)^k + \tfrac{3}{10} = 0.3$$

which is smaller, as it should be. Alternatively, as we saw in Example 7.3.2,

$$\xi_{\min} = r_v^2 h(0) = 0.3$$

as above.

In the previous example we looked at the problem of estimating a process $d(n)$ from noisy measurements

$$x(n) = d(n) + v(n) \tag{7.62}$$

What we found was that if $d(n)$ is generated by the difference equation

$$d(n) = 0.8d(n-1) + w(n) \tag{7.63}$$

then the estimate of $d(n)$, when computed recursively, is given by

$$\hat{d}(n) = 0.5\hat{d}(n-1) + 0.375x(n)$$

What is interesting to note is that this recursive estimator may also be rewritten in the form

$$\hat{d}(n) = 0.8\hat{d}(n-1) + 0.375\Big[x(n) - 0.8\hat{d}(n-1)\Big] \tag{7.64}$$

The interpretation of this equation is as follows. The quantity $\hat{d}(n)$ is the minimum mean-square estimate of $d(n)$ that is based on all observations of $x(n)$ up to time n. Similarly, $\hat{d}(n-1)$ is the minimum mean-square estimate of $d(n-1)$ that is formed from all of the observations of $x(n)$ up to time $n-1$. Given $\hat{d}(n-1)$, before the next value of $x(n)$ is observed, we may "predict" what the next value of $d(n)$ should be. In light of Eq. (7.63), since $w(n)$ is a zero mean process then this estimate should be $0.8\hat{d}(n-1)$. Given that $x(n) = d(n) + v(n)$ we may then use this prediction to "predict" what the next measurement should be,

$$\hat{x}(n) = 0.8\hat{d}(n-1)$$

Finally, with the arrival of the new measurement, $x(n)$, since the prediction will not be perfect, there will be an error

$$\alpha(n) = x(n) - \hat{x}(n)$$

This error, called the *innovations* process, represents the "new information" that is brought with the observation $x(n)$. In other words, $\alpha(n)$ corresponds to that part of $x(n)$ that cannot be predicted. Therefore, the estimate $\hat{d}(n)$ is modified by adding a correction, which is the innovations process after it has been scaled by a *gain*, K, referred to as the *Kalman Gain*. As we will soon discover in Section 7.4, Eq. (7.64) is the steady-state Kalman filter for estimating the stationary process $d(n)$.

7.3.4 Causal Linear Prediction

In Section 7.2.2, we derived the FIR Wiener filter for linear prediction. In this section, we look at the design of the optimum causal linear predictor

$$\hat{x}(n+1) = \sum_{k=0}^{\infty} h(k)x(n-k)$$

that produces the best estimate of $x(n+1)$ given $x(k)$ for all $k \leq n$. Since the infinite past is now being used to predict the next value of $x(n)$, we expect a smaller mean-square prediction error than for an FIR linear predictor.

For linear prediction, $d(n) = x(n+1)$ and the cross-correlation between $x(n)$ and $d(n)$ is

$$r_{dx}(k) = E\{d(n)x^*(n-k)\} = E\{x(n+1)x^*(n-k)\} = r_x(k+1)$$

Therefore, $P_{dx}(z) = zP_x(z)$ and the causal Wiener predictor is

$$H(z) = \frac{1}{\sigma_0^2 Q(z)}\left[\frac{zP_x(z)}{Q^*(1/z^*)}\right]_+ \tag{7.65}$$

However, since $P_x(z) = \sigma_0^2 Q(z)Q^*(1/z^*)$, then Eq. (7.65) may be simplified as follows

$$H(z) = \frac{1}{Q(z)}\Big[z\,Q(z)\Big]_+ \tag{7.66}$$

Now, recall that $Q(z)$ is a monic polynomial

$$Q(z) = 1 + q(1)z^{-1} + q(2)z^{-2} + q(3)z^{-3} + \cdots$$

Therefore, the positive-time part of $zQ(z)$ is

$$\begin{aligned}\Big[zQ(z)\Big]_+ &= \Big[z + q(1) + q(2)z^{-1} + q(3)z^{-2} + \cdots\Big]_+ \\ &= q(1) + q(2)z^{-1} + q(3)z^{-2} + \cdots \\ &= z\Big[Q(z) - 1\Big]\end{aligned} \tag{7.67}$$

Substituting Eq. (7.67) into Eq. (7.66) the causal linear predictor becomes

$$\boxed{H(z) = \frac{1}{Q(z)}z\Big[Q(z) - 1\Big] = z\Big[1 - \frac{1}{Q(z)}\Big]} \tag{7.68}$$

Finally, for the minimum mean-square error, we have

$$\xi_{\min} = \frac{1}{2\pi j}\oint_C \Big[P_d(z) - H(z)P_{dx}^*(1/z^*)\Big]z^{-1}dz \tag{7.69}$$

Since $P_d(z) = P_x(z)$ and $P_{dx}(z) = zP_x(z)$, it follows that the minimum mean-square error is

$$\xi_{\min} = \frac{1}{2\pi j}\oint_C \Big[P_x(z) - z^{-1}H(z)P_x^*(1/z^*)\Big]z^{-1}dz$$

Using the symmetry of the power spectrum, $P_x(z) = P_x^*(1/z^*)$, this becomes

$$\xi_{\min} = \frac{1}{2\pi j}\oint_C P_x(z)\Big[1 - z^{-1}H(z)\Big]z^{-1}dz$$

Substituting the system function for the causal Wiener linear predictor, Eq. (7.68), into this

expression yields

$$\xi_{\min} = \frac{1}{2\pi j}\oint_C P_x(z)\left[1-\left(1-\frac{1}{Q(z)}\right)\right]z^{-1}dz$$

$$= \frac{1}{2\pi j}\oint_C \frac{P_x(z)}{Q(z)}z^{-1}dz = \frac{1}{2\pi j}\oint_C \sigma_0^2 Q^*(1/z^*)z^{-1}dz$$

$$= \sigma_0^2 q(0) \tag{7.70}$$

Thus, since $Q(z)$ is monic with $q(0) = 1$ we have the remarkably simple result

$$\boxed{\xi_{\min} = \sigma_0^2} \tag{7.71}$$

An interesting relationship between the minimum mean-square error and the power spectrum follows from Eq. (3.103) on p. 105. Specifically,

$$\boxed{\xi_{\min} = \exp\left\{\frac{1}{2\pi}\int_{-\pi}^{\pi} \ln P_x(e^{j\omega})d\omega\right\}} \tag{7.72}$$

which is known as the *Kolmogorov-Szegö formula.*

Let us now apply these results to the linear prediction of an autoregressive process. If $x(n)$ is an AR(p) process with a power spectrum

$$P_x(z) = \frac{\sigma_0^2}{A(z)A^*(1/z^*)}$$

where

$$A(z) = 1 + \sum_{k=1}^{p} a(k)z^{-k}$$

is a minimum phase polynomial having all of its roots *inside* the unit circle, then the optimum linear predictor is

$$H(z) = z\Big[1 - A(z)\Big] = -a(1) - a(2)z^{-1} - \cdots - a(p)z^{-p+1} \tag{7.73}$$

which is an FIR filter. Therefore, given the entire past history of $x(n)$, only the last p values of $x(n)$ are used in the prediction of $x(n+1)$. However, this should not be surprising if we recall that an AR(p) random process satisfies a difference equation of the form

$$x(n) = -a(1)x(n-1) - a(2)x(n-2) - \cdots - a(p)x(n-p) + w(n)$$

where $w(n)$ is white noise. Since $w(n+1)$ cannot be predicted from $x(n)$ or previous values of $x(n)$ then the best that can be done in predicting $x(n+1)$ is to use the model for $x(n)$ and ignore the noise

$$\hat{x}(n+1) = -a(1)x(n) - a(2)x(n-1) - \cdots - a(p)x(n-p+1)$$

This predictor, of course, is the same as the one given in Eq. (7.73).

Example 7.3.3 ***Causal Linear Prediction for an AR Process***

Consider the real-valued AR(2) process

$$x(n) = 0.9x(n-1) - 0.2x(n-2) + w(n) \tag{7.74}$$

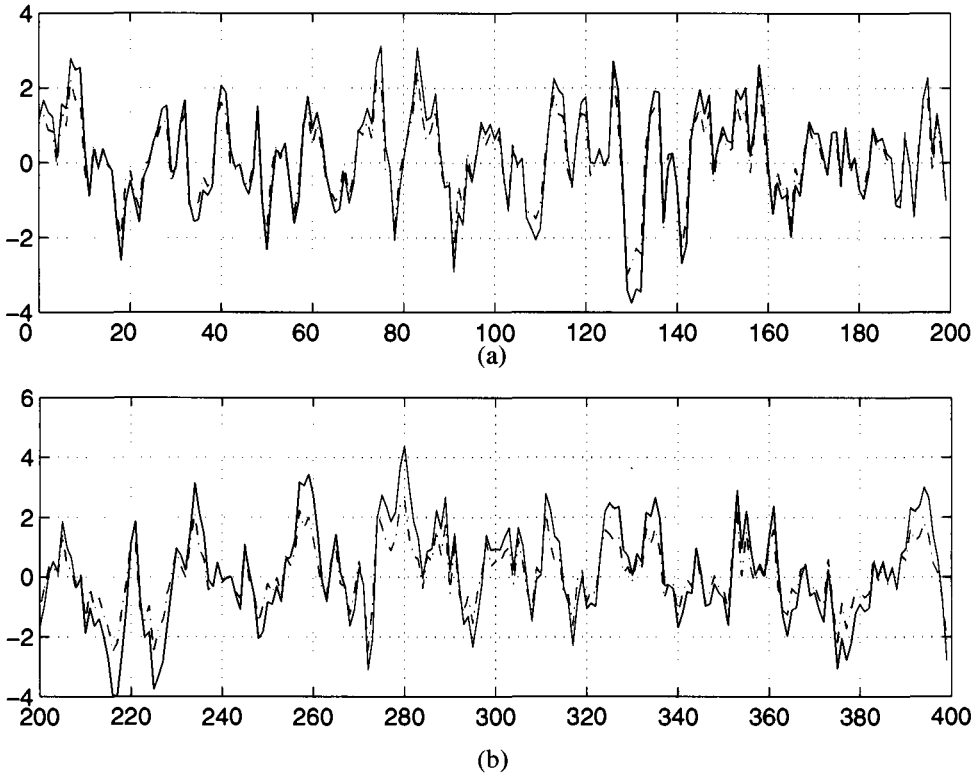


Figure 7.14 *Linear prediction of an AR(2) process using an IIR Wiener filter. The process is represented by the solid line and the prediction by a dotted line. (a) Prediction assuming that the model parameters are known. (b) Prediction using estimated model parameters.*

where $w(n)$ is unit variance zero mean white noise. Since

$$P_x(z) = \frac{1}{A(z)A(z^{-1})}$$

where

$$A(z) = 1 - 0.9z^{-1} + 0.2z^{-2}$$

then the optimum linear predictor is

$$H(z) = z\Big[1 - A(z)\Big] = z\Big[0.9z^{-1} - 0.2z^{-2}\Big] = 0.9 - 0.2z^{-1}$$

and the prediction of $x(n)$ is formed as follows

$$\hat{x}(n+1) = 0.9x(n) - 0.2x(n-1) \tag{7.75}$$

As a specific example, an AR(2) process $x(n)$ that is generated according to Eq. (7.74) is shown in Fig. 7.14*a*. Also shown is the prediction of $x(n)$ using the predictor given in Eq. (7.75), which has an average squared prediction error of

$$\xi = \frac{1}{N}\sum_{n=0}^{N-1}\big[x(n+1) - \hat{x}(n+1)\big]^2 = 0.0324$$

This example, however, is somewhat contrived since, in practice, we cannot expect to have prior knowledge of the statistics of $x(n)$. Therefore, a more realistic example is the

following. Using the given data, $x(n)$, in Fig. 7.14*a* we first estimate the AR parameters using the autocorrelation method. With

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

where $N = 200$ we find

$$r_x(0) = 2.1904, \quad r_x(1) = 1.5462, \quad \text{and} \quad r_x(2) = 0.8670$$

Thus, the normal equations are

$$\begin{bmatrix} 2.1904 & 1.5462 \\ 1.5462 & 2.1904 \end{bmatrix}\begin{bmatrix} a(1) \\ a(2) \end{bmatrix} = -\begin{bmatrix} 1.5462 \\ 0.8670 \end{bmatrix}$$

Solving for $a(1)$ and $a(2)$ we find

$$\begin{bmatrix} a(1) \\ a(2) \end{bmatrix} = \begin{bmatrix} -0.8500 \\ 0.2042 \end{bmatrix}$$

Therefore, the predictor becomes

$$\hat{x}(n+1) = 0.85x(n) - 0.2042x(n-1)$$

Now, instead of using the predictor on the data that is used to estimate the predictor coefficients, we apply the predictor to the next sequence of 200 data values. The results are shown in Fig. 7.14*b* and we see that the performance of the predictor is similar to that in Fig. 7.14*a*.

In the previous example, we looked at linear prediction for an autoregressive random process. Although these predictors are FIR filters, as we see in the next example, the Wiener predictor for an ARMA processes is an IIR filter.

Example 7.3.4 ***Causal Linear Prediction for an ARMA Process***

In this example, we consider the design of the optimum causal linear predictor for a process, $x(n)$, that has a power spectrum

$$P_x(z) = \frac{(1 - 0.6z^{-1} + 0.36z^{-2})(1 - 0.6z + 0.36z^2)}{(1 - 0.8z^{-1})(1 - 0.8z)}$$

The optimum predictor is

$$H(z) = z\left[1 - \frac{1}{Q(z)}\right]$$

where $Q(z)$ is the minimum phase spectral factor of $P_x(z)$. Since

$$Q(z) = \frac{1 - 0.6z^{-1} + 0.36z^{-2}}{1 - 0.8z^{-1}}$$

then the system function of the linear predictor is

$$H(z) = z\left[1 - \frac{1 - 0.8z^{-1}}{1 - 0.6z^{-1} + 0.36z^{-2}}\right] = z\left[\frac{0.2z^{-1} + 0.36z^{-2}}{1 - 0.6z^{-1} + 0.36z^{-2}}\right]$$

$$= \frac{0.2 + 0.36z^{-1}}{1 - 0.6z^{-1} + 0.36z^{-2}}$$

which is an IIR filter. Therefore, the prediction of $x(n+1)$ is computed using the recursive equation

$$\hat{x}(n+1) = 0.6\hat{x}(n) - 0.36\hat{x}(n-1) + 0.2x(n) + 0.36x(n-1)$$

7.3.5 Wiener Deconvolution

We conclude this section on Wiener filtering with a discussion of an extremely difficult and important problem that arises in many applications. This problem, known as *deconvolution*, is concerned with the recovery of a signal $d(n)$ that has been convolved with a filter $g(n)$ that may not be precisely known,

$$x(n) = d(n) * g(n) \tag{7.76}$$

Convolutional distortion is often introduced in the process of measuring or recording data. For example, a camera that is slightly out of focus and a band-limited communication channel may be modeled as systems that introduce convolutional distortion. In some applications, convolutional distortion may be inadvertently introduced as would be the case if a camera were in motion during photographic exposure [1]. If the "blurring" function, $g(n)$, is known perfectly and if $g(n)$ has an inverse $g^{-1}(n)$ so that

$$g(n) * g^{-1}(n) = \delta(n)$$

then there is, in theory, no difficulty in recovering $d(n)$ from $x(n)$ since the inverse filter would perform the desired restoration, i.e.,

$$d(n) = x(n) * g^{-1}(n) \tag{7.77}$$

or, in the frequency domain,

$$D(e^{j\omega}) = \frac{X(e^{j\omega})}{G(e^{j\omega})}$$

In practice, however, precise knowledge of $g(n)$ or $G(e^{j\omega})$ is generally not available. Another problem is that the frequency response, $G(e^{j\omega})$, is often equal to zero at one or more frequencies or is very small over a band of frequencies. This results in either a noninvertible $G(e^{j\omega})$ or one that is ill-conditioned. In addition, since noise is invariably introduced in the measurement process, a more accurate model for the observed signal would be

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

where $w(n)$ is additive noise that is often assumed to be uncorrelated with $d(n)$. In this case, even if the inverse filter $G^{-1}(e^{j\omega})$ exists and is well-behaved, when the inverse filter is applied to $x(n)$ the restored signal is

$$\hat{D}(e^{j\omega}) = D(e^{j\omega}) + \underbrace{\frac{W(e^{j\omega})}{G(e^{j\omega})}}_{\text{noise}} = D(e^{j\omega}) + V(e^{j\omega})$$

which, in the time domain, becomes

$$\hat{d}(n) = d(n) + v(n) \tag{7.78}$$

Thus, $\hat{d}(n)$ is the sum of the original signal $d(n)$ and a filtered noise term $v(n)$. The difficulty with this solution is that, if $G(e^{j\omega}) \approx 0$ over some range of frequencies, then the inverse filter, $1/G(e^{j\omega})$, becomes large and the noise $w(n)$ will be amplified.

Another approach to the deconvolution problem is to design a Wiener filter that produces the minimum mean-square estimate of $d(n)$ from $x(n)$. Thus, let $h(n)$ be an IIR linear shift-invariant filter and let $\hat{d}(n)$ be the estimate of $d(n)$ that is produced by filtering $x(n)$ with $h(n)$

$$\hat{d}(n) = x(n) * h(n) = \sum_{l=-\infty}^{\infty} h(l)x(n-l)$$

Note that we are assuming that the filter is noncausal. The filter coefficients $h(n)$ that minimize the mean-square error

$$\xi = E\{|d(n) - \hat{d}(n)|^2\}$$

are the solution to the Wiener-Hopf equations, which, in the frequency domain, becomes

$$H(e^{j\omega}) = \frac{P_{dx}(e^{j\omega})}{P_x(e^{j\omega})} \tag{7.79}$$

Therefore, all that needs to be done in the design of $H(e^{j\omega})$ is to find the power spectral densities $P_{dx}(e^{j\omega})$ and $P_x(e^{j\omega})$. Since $w(n)$ is assumed to be uncorrelated with $d(n)$, then $w(n)$ will also be uncorrelated with $d(n) * g(n)$. As a result, the power spectral density of $x(n)$ is the sum of the power spectrum of $d(n) * g(n)$ and the power spectrum of $w(n)$,

$$P_x(e^{j\omega}) = P_d(e^{j\omega})|G(e^{j\omega})|^2 + P_w(e^{j\omega}) \tag{7.80}$$

In addition, the cross-power spectral density $P_{dx}(e^{j\omega})$ is

$$P_{dx}(e^{j\omega}) = P_d(e^{j\omega})G^*(e^{j\omega}) \tag{7.81}$$

Therefore, substituting Eqs. (7.80) and (7.81) into Eq. (7.79) we find that the optimum Wiener filter for deconvolution is given by

$$H(e^{j\omega}) = \frac{P_d(e^{j\omega})G^*(e^{j\omega})}{P_d(e^{j\omega})|G(e^{j\omega})|^2 + P_w(e^{j\omega})}$$

It is interesting to note that if we assume that there are no values of ω for which $G(e^{j\omega})$ is equal to zero, and if we factor out the inverse filter $1/G(e^{j\omega})$ from $H(e^{j\omega})$, then the noncausal Wiener filter may be written as

$$H(e^{j\omega}) = \frac{1}{G(e^{j\omega})}\left[\frac{P_d(e^{j\omega})}{P_d(e^{j\omega}) + P_w(e^{j\omega})/|G(e^{j\omega})|^2}\right] \tag{7.82}$$

Since the power spectrum of the filtered noise, $v(n)$, in Eq. (7.78) is

$$P_v(e^{j\omega}) = P_w(e^{j\omega})/|G(e^{j\omega})|^2$$

it follows that the term in brackets in Eq. (7.82) may be written as

$$F(e^{j\omega}) = \frac{P_d(e^{j\omega})}{P_d(e^{j\omega}) + P_v(e^{j\omega})} \tag{7.83}$$

Therefore, comparing Eq. (7.83) with Eq. (7.42) we see that $F(e^{j\omega})$ is the noncausal IIR Wiener smoothing filter for estimating $d(n)$ from

$$y(n) = d(n) + v(n) \tag{7.84}$$

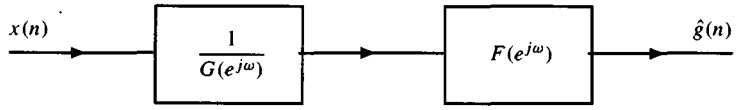


Figure 7.15 *Wiener deconvolution realized as the cascade of an inverse filter with a smoothing filter* $F(e^{j\omega})$.

Thus, as shown in Fig. 7.15, the Wiener filter $H(e^{j\omega})$ may be viewed as a cascade of the inverse filter $1/G(e^{j\omega})$ followed by a noncausal Wiener smoothing filter for reducing the filtered noise.

7.4 DISCRETE KALMAN FILTER

In Section 7.3.2 we considered the problem of designing a causal Wiener filter to estimate a process $d(n)$ from a set of noisy observations $x(n) = d(n) + v(n)$. The primary limitation with the solution that was derived is that it requires that $d(n)$ and $x(n)$ be jointly wide-sense stationary processes. Since most processes encountered in practice are nonstationary, this constraint limits the usefulness of the Wiener filter. Therefore, in this section we re-examine this estimation problem within the context of nonstationary processes and derive what is known as the *discrete Kalman filter.*

To begin, let us look briefly once again at the causal Wiener filter for estimating a process $x(n)$ from the noisy measurements[3]

$$y(n) = x(n) + v(n) \tag{7.85}$$

In Example 7.3.2 we considered the specific problem of estimating an AR(1) process of the form[4]

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

from

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

where $w(n)$ and $v(n)$ are uncorrelated white noise processes. What we discovered was that the optimum linear estimate of $x(n)$, using *all* of the measurements, $y(k)$, for $k \leq n$, could be computed with a recursion of the form

$$\hat{x}(n) = a(1)\hat{x}(n-1) + K\Big[y(n) - a(1)\hat{x}(n-1)\Big] \tag{7.86}$$

where K is a constant, referred to as the *Kalman gain,* that minimizes the mean-square error $E\{|x(n) - \hat{x}(n)|^2\}$. However, there are two problems with this solution that need to be

[3]Here we have introduced a slight change in notation in order to be consistent with the notation that is commonly used in the Kalman filtering literature. Instead of $d(n)$, the signal that is to be estimated will be denoted by $x(n)$ and the noisy observations denoted by $y(n)$.

[4]In the literature on Kalman filtering, the process $x(n)$ is usually assumed to be generated according to the difference equation

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

This difference is not significant, however, since both processes have the same autocorrelation.

addressed. First is the requirement that $x(n)$ and $y(n)$ be jointly wide-sense stationary processes. For example, Eq. (7.86) is not the optimum linear estimate if $x(n)$ is a nonstationary process, such as the one that is generated by the time-varying difference equation

$$x(n) = a_{n-1}(1)x(n-1) + w(n)$$

Nevertheless, what we will discover is that the optimum estimate may be written as

$$\hat{x}(n) = a_{n-1}(1)\hat{x}(n-1) + K(n)\Big[y(n) - a_{n-1}(1)\hat{x}(n-1)\Big] \tag{7.87}$$

where $K(n)$ is a suitably chosen (time-varying) gain. The second problem with the Wiener solution is that it does not allow the filter to be "turned on" at time $n = 0$. In other words, implicit in the development of the causal Wiener filter is the assumption that the observations $y(k)$ are available for all $k \leq n$. Again, however, we will find that this problem is addressed with an estimate of the form given in Eq. (7.87).

Although the discussion above is only concerned with the estimation of an AR(1) process from noisy measurements, using state variables we may easily extend these results to more general processes. For example, let $x(n)$ be an AR(p) process that is generated according to the difference equation

$$x(n) = \sum_{k=1}^{p} a(k)x(n-k) + w(n) \tag{7.88}$$

and suppose that $x(n)$ is measured in the presence of additive noise

$$y(n) = x(n) + v(n) \tag{7.89}$$

If we let $\mathbf{x}(n)$ be the p-dimensional state vector

$$\mathbf{x}(n) = \begin{bmatrix} x(n) \\ x(n-1) \\ \vdots \\ x(n-p+1) \end{bmatrix}$$

then Eqs. (7.88) and (7.89) may be written in terms of $\mathbf{x}(n)$ as follows

$$\mathbf{x}(n) = \begin{bmatrix} a(1) & a(2) & \cdots & a(p-1) & a(p) \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \cdots & & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix} \mathbf{x}(n-1) + \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} w(n) \tag{7.90}$$

and

$$y(n) = \begin{bmatrix} 1, & 0, & \ldots, & 0 \end{bmatrix} \mathbf{x}(n) + v(n) \tag{7.91}$$

Using matrix notation to simplify these equations we have

$$\begin{aligned} \mathbf{x}(n) &= \mathbf{A}\mathbf{x}(n-1) + \mathbf{w}(n) \\ y(n) &= \mathbf{c}^T\mathbf{x}(n) + v(n) \end{aligned} \tag{7.92}$$

where $\mathbf{A}$ is a $p \times p$ *state transition matrix*, $\mathbf{w}(n) = \begin{bmatrix} w(n), & 0, & \ldots, & 0 \end{bmatrix}^T$ is a vector noise process, and $\mathbf{c}$ is a unit vector of length p. As in Eq. (7.87) for the case of an AR(1) process,

the optimum estimate of the state vector $\mathbf{x}(n)$, using all of the measurements up to time n, may be expressed in the form

$$\hat{\mathbf{x}}(n) = \mathbf{A}\hat{\mathbf{x}}(n-1) + \mathbf{K}\Big[y(n) - \mathbf{c}^T\mathbf{A}\hat{\mathbf{x}}(n-1)\Big] \tag{7.93}$$

where $\mathbf{K}$ is a Kalman gain vector.

Although only applicable to stationary AR(p) processes, Eq. (7.92) may be easily generalized to nonstationary processes as follows. Let $\mathbf{x}(n)$ be a state vector of dimension p that evolves according to the difference equation

$$\boxed{\mathbf{x}(n) = \mathbf{A}(n-1)\mathbf{x}(n-1) + \mathbf{w}(n)} \tag{7.94}$$

where $\mathbf{A}(n-1)$ is a time-varying $p \times p$ state transition matrix and $\mathbf{w}(n)$ is a vector of zero-mean white noise processes with

$$E\Big\{\mathbf{w}(n)\mathbf{w}^H(k)\Big\} = \begin{cases} \mathbf{Q}_w(n) & ; \quad k = n \\ \mathbf{0} & ; \quad k \neq n \end{cases} \tag{7.95}$$

In addition, let $\mathbf{y}(n)$ be a *vector* of observations that are formed according to

$$\boxed{\mathbf{y}(n) = \mathbf{C}(n)\mathbf{x}(n) + \mathbf{v}(n)} \tag{7.96}$$

where $\mathbf{y}(n)$ is a vector of length q, $\mathbf{C}(n)$ is a time-varying $q \times p$ matrix, and $\mathbf{v}(n)$ is a vector of zero mean white noise processes that are statistically independent of $\mathbf{w}(n)$ with

$$E\Big\{\mathbf{v}(n)\mathbf{v}^H(k)\Big\} = \begin{cases} \mathbf{Q}_v(n) & ; \quad k = n \\ \mathbf{0} & ; \quad k \neq n \end{cases} \tag{7.97}$$

Generalizing the result given in Eq. (7.93), we expect the optimum *linear* estimate of $\mathbf{x}(n)$ to be expressible in the form

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

With the appropriate *Kalman gain matrix* $\mathbf{K}(n)$, this recursion corresponds to the *discrete Kalman filter*. We will now show that the optimum *linear* recursive estimate of $\mathbf{x}(n)$ has this form and derive the optimum Kalman gain $\mathbf{K}(n)$ that minimizes the mean-square estimation error. In the following discussion it is assumed that $\mathbf{A}(n)$, $\mathbf{C}(n)$, $\mathbf{Q}_w(n)$, and $\mathbf{Q}_v(n)$ are known.

In our development of the discrete Kalman filter, we will let $\hat{\mathbf{x}}(n|n)$ denote the best linear estimate of $\mathbf{x}(n)$ at time n given the observations $\mathbf{y}(i)$ for $i = 1, 2, \ldots, n$, and we will let $\hat{\mathbf{x}}(n|n-1)$ denote the best estimate given the observations up to time $n-1$. With $\mathbf{e}(n|n)$ and $\mathbf{e}(n|n-1)$ the corresponding state estimation errors,

$$\begin{aligned} \mathbf{e}(n|n) &= \mathbf{x}(n) - \hat{\mathbf{x}}(n|n) \\ \mathbf{e}(n|n-1) &= \mathbf{x}(n) - \hat{\mathbf{x}}(n|n-1) \end{aligned}$$

and $\mathbf{P}(n|n)$ and $\mathbf{P}(n|n-1)$ the error covariance matrices,

$$\begin{aligned} \mathbf{P}(n|n) &= E\Big\{\mathbf{e}(n|n)\mathbf{e}^H(n|n)\Big\} \\ \mathbf{P}(n|n-1) &= E\Big\{\mathbf{e}(n|n-1)\mathbf{e}^H(n|n-1)\Big\} \end{aligned} \tag{7.98}$$

the problem that we would like to solve is the following. Suppose that matrix we are given an estimate $\hat{\mathbf{x}}(0|0)$ of the state $\mathbf{x}(0)$, and that the error covariance matrix for this estimate, $\mathbf{P}(0|0)$, is known. When the measurement $\mathbf{y}(1)$ becomes available the goal is to update $\hat{\mathbf{x}}(0|0)$ and find the estimate $\hat{\mathbf{x}}(1|1)$ of the state at time $n = 1$ that minimizes the mean-square error

$$\xi(1) = E\{\|\mathbf{e}(1|1)\|^2\} = \text{tr}\{\mathbf{P}(1|1)\} = \sum_{i=0}^{p-1} E\{|e_i(1|1)|^2\} \tag{7.99}$$

After $\hat{\mathbf{x}}(1|1)$ has been determined and the error covariance $\mathbf{P}(1|1)$ found, the estimation is repeated for the next observation $\mathbf{y}(2)$. Thus, for each $n > 0$, given $\hat{\mathbf{x}}(n-1|n-1)$ and $\mathbf{P}(n-1|n-1)$, when a new observation, $y(n)$, becomes available, the problem is to find the minimum mean-square estimate $\hat{\mathbf{x}}(n|n)$ of the state vector $\mathbf{x}(n)$. The solution to this problem will be derived in two steps. First, given $\hat{\mathbf{x}}(n-1|n-1)$ we will find $\hat{\mathbf{x}}(n|n-1)$, which is the best estimate of $\mathbf{x}(n)$ *without* the observation $\mathbf{y}(n)$. Then, given $\mathbf{y}(n)$ and $\hat{\mathbf{x}}(n|n-1)$ we will estimate $\mathbf{x}(n)$.

In the first step, since no new measurements are used to estimate $\mathbf{x}(n)$, all that is known is that $\mathbf{x}(n)$ evolves according to the state equation

$$\mathbf{x}(n) = \mathbf{A}(n-1)\mathbf{x}(n-1) + \mathbf{w}(n)$$

Since $\mathbf{w}(n)$ is a zero mean white noise process (and the values of $\mathbf{w}(n)$ are unknown), then we may predict $\mathbf{x}(n)$ as follows,

$$\boxed{\hat{\mathbf{x}}(n|n-1) = \mathbf{A}(n-1)\hat{\mathbf{x}}(n-1|n-1)} \tag{7.100}$$

which has an estimation error given by

$$\begin{aligned}\mathbf{e}(n|n-1) &= \mathbf{x}(n) - \hat{\mathbf{x}}(n|n-1) \\ &= \mathbf{A}(n-1)\mathbf{x}(n-1) + \mathbf{w}(n) - \mathbf{A}(n-1)\hat{\mathbf{x}}(n-1|n-1) \\ &= \mathbf{A}(n-1)\mathbf{e}(n-1|n-1) + \mathbf{w}(n)\end{aligned} \tag{7.101}$$

Note that since $\mathbf{w}(n)$ has zero mean, if $\hat{\mathbf{x}}(n-1|n-1)$ is an unbiased estimate of $\mathbf{x}(n-1)$, i.e.,

$$E\{\mathbf{e}(n-1|n-1)\} = 0$$

then $\hat{\mathbf{x}}(n|n-1)$ will be an unbiased estimate of $\mathbf{x}(n)$,

$$E\{\mathbf{e}(n|n-1)\} = 0$$

Finally, since the estimation error $\mathbf{e}(n-1|n-1)$ is uncorrelated with $\mathbf{w}(n)$ (a consequence of the fact that $\mathbf{w}(n)$ is a white noise sequence), then

$$\boxed{\mathbf{P}(n|n-1) = \mathbf{A}(n-1)\mathbf{P}(n-1|n-1)\mathbf{A}^H(n-1) + \mathbf{Q}_w(n)} \tag{7.102}$$

where $\mathbf{Q}_w(n)$ is the covariance matrix for the noise process $\mathbf{w}(n)$. This completes the first step of the Kalman filter.

In the second step we incorporate the new measurement $\mathbf{y}(n)$ into the estimate $\hat{\mathbf{x}}(n|n-1)$. A linear estimate of $\mathbf{x}(n)$ that is based on $\hat{\mathbf{x}}(n|n-1)$ and $\mathbf{y}(n)$ is of the form

$$\hat{\mathbf{x}}(n|n) = \mathbf{K}'(n)\hat{\mathbf{x}}(n|n-1) + \mathbf{K}(n)\mathbf{y}(n) \tag{7.103}$$

where $\mathbf{K}(n)$ and $\mathbf{K}'(n)$ are matrices, yet to be specified. The requirement that is imposed on $\hat{\mathbf{x}}(n|n)$ is that it be *unbiased*, $E\{\mathbf{e}(n|n)\} = 0$, and that it minimize the mean-square error, $E\{\|\mathbf{e}(n|n)\|^2\}$. Using Eq. (7.103) we may express $\mathbf{e}(n|n)$ in terms of $\mathbf{e}(n|n-1)$ as follows

$$\begin{aligned}\mathbf{e}(n|n) &= \mathbf{x}(n) - \mathbf{K}'(n)\hat{\mathbf{x}}(n|n-1) - \mathbf{K}(n)\mathbf{y}(n) \\ &= \mathbf{x}(n) - \mathbf{K}'(n)\Big[\mathbf{x}(n) - \mathbf{e}(n|n-1)\Big] - \mathbf{K}(n)\Big[\mathbf{C}(n)\mathbf{x}(n) + \mathbf{v}(n)\Big] \\ &= \Big[\mathbf{I} - \mathbf{K}'(n) - \mathbf{K}(n)\mathbf{C}(n)\Big]\mathbf{x}(n) + \mathbf{K}'(n)\mathbf{e}(n|n-1) - \mathbf{K}(n)\mathbf{v}(n) \quad (7.104)\end{aligned}$$

Since $E\{\mathbf{v}(n)\} = 0$ and $E\{e(n|n-1)\} = 0$, then $\hat{\mathbf{x}}(n|n)$ will be unbiased for any $\mathbf{x}(n)$ only if the term in brackets is zero,

$$\mathbf{K}'(n) = \mathbf{I} - \mathbf{K}(n)\mathbf{C}(n)$$

With this constraint, it follows from Eq. (7.103) that $\hat{\mathbf{x}}(n|n)$ has the form

$$\hat{\mathbf{x}}(n|n) = \Big[\mathbf{I} - \mathbf{K}(n)\mathbf{C}(n)\Big]\hat{\mathbf{x}}(n|n-1) + \mathbf{K}(n)\mathbf{y}(n) \quad (7.105)$$

or

$$\boxed{\hat{\mathbf{x}}(n|n) = \hat{\mathbf{x}}(n|n-1) + \mathbf{K}(n)\Big[\mathbf{y}(n) - \mathbf{C}(n)\hat{\mathbf{x}}(n|n-1)\Big]} \quad (7.106)$$

and the error is

$$\begin{aligned}\mathbf{e}(n|n) &= \mathbf{K}'(n)\mathbf{e}(n|n-1) - \mathbf{K}(n)\mathbf{v}(n) \\ &= \Big[\mathbf{I} - \mathbf{K}(n)\mathbf{C}(n)\Big]\mathbf{e}(n|n-1) - \mathbf{K}(n)\mathbf{v}(n)\end{aligned} \quad (7.107)$$

Since $\mathbf{v}(n)$ is uncorrelated with $\mathbf{w}(n)$, then $\mathbf{v}(n)$ is uncorrelated with $\mathbf{x}(n)$ and, therefore, it is uncorrelated with $\hat{\mathbf{x}}(n|n-1)$. In addition, since $\mathbf{e}(n|n-1) = \mathbf{x}(n) - \hat{\mathbf{x}}(n|n-1)$, then $\mathbf{v}(n)$ is uncorrelated with $\mathbf{e}(n|n-1)$,

$$E\{\mathbf{e}(n|n-1)v(n)\} = 0$$

Thus, the error covariance matrix for $\mathbf{e}(n|n)$ is

$$\mathbf{P}(n|n) = E\{\mathbf{e}(n|n)\mathbf{e}^H(n|n)\} \quad (7.108)$$

$$= \Big[\mathbf{I} - \mathbf{K}(n)\mathbf{C}(n)\Big]\mathbf{P}(n|n-1)\Big[\mathbf{I} - \mathbf{K}(n)\mathbf{C}(n)\Big]^H + \mathbf{K}(n)\mathbf{Q}_v(n)\mathbf{K}^H(n) \quad (7.109)$$

Next, we must find the value for the Kalman gain $\mathbf{K}(n)$ that minimizes the mean-square error

$$\xi(n) = \mathrm{tr}\{\mathbf{P}(n|n)\}$$

This may be accomplished in a couple of different ways. Although requiring some special matrix differentiation formulas, we will take the most expedient approach of differentiating $\xi(n)$ with respect to $\mathbf{K}(n)$, setting the derivative to zero, and solving for $\mathbf{K}(n)$. Using the matrix differentiation formulas

$$\frac{d}{d\mathbf{K}}\mathrm{tr}(\mathbf{KA}) = \mathbf{A}^H \quad (7.110)$$

and

$$\frac{d}{d\mathbf{K}}\mathrm{tr}(\mathbf{KAK}^H) = 2\mathbf{KA} \quad (7.111)$$

we have

$$\frac{d}{d\mathbf{K}}\text{tr}\{\mathbf{P}(n|n)\} = -2\Big[\mathbf{I} - \mathbf{K}(n)\mathbf{C}(n)\Big]\mathbf{P}(n|n-1)\mathbf{C}^H(n) + 2\mathbf{K}(n)\mathbf{Q}_v(n) = 0 \tag{7.112}$$

Solving for $\mathbf{K}(n)$ gives the desired expression for the Kalman gain,

$$\boxed{\mathbf{K}(n) = \mathbf{P}(n|n-1)\mathbf{C}^H(n)\Big[\mathbf{C}(n)\mathbf{P}(n|n-1)\mathbf{C}^H(n) + \mathbf{Q}_v(n)\Big]^{-1}} \tag{7.113}$$

Having found the Kalman gain vector, we may simplify the expression given in Eq. (7.109) for the error covariance. First, we rewrite the expression for $\mathbf{P}(n|n)$ as follows,

$$\begin{aligned}\mathbf{P}(n|n) = &\Big[\mathbf{I} - \mathbf{K}(n)\mathbf{C}(n)\Big]\mathbf{P}(n|n-1) \\ &-\Big\{\big[\mathbf{I} - \mathbf{K}(n)\mathbf{C}(n)\big]\mathbf{P}(n|n-1)\mathbf{C}^H(n) + \mathbf{K}(n)\mathbf{Q}_v(n)\Big\}\mathbf{K}^H(n)\end{aligned}$$

From Eq. (7.112), however, it follows that the second term is equal to zero, which leads to the desired expression for the error covariance matrix

$$\boxed{\mathbf{P}(n|n) = \Big[\mathbf{I} - \mathbf{K}(n)\mathbf{C}(n)\Big]\mathbf{P}(n|n-1)} \tag{7.114}$$

Thus far we have derived the Kalman filtering equations for recursively estimating the state vector $\mathbf{x}(n)$. All that needs to be done to complete the recursion is to determine how the recursion should be initialized at time $n = 0$. Since the value of the initial state is unknown, in the absence of any observed data at time $n = 0$, the initial estimate is chosen to be

$$\hat{\mathbf{x}}(0|0) = E\{\mathbf{x}(0)\}$$

and, for the initial value for the error covariance matrix, we have

$$\mathbf{P}(0|0) = E\{\mathbf{x}(0)\mathbf{x}^H(0)\}$$

This choice for the initial conditions makes $\hat{\mathbf{x}}(0|0)$ an unbiased estimate of $\mathbf{x}(0)$ and ensures that $\hat{\mathbf{x}}(n|n)$ will be unbiased for all n (recall that the Kalman filtering update equations were derived with the constraint that $\hat{\mathbf{x}}(n|n)$ be unbiased). This completes the derivation of the discrete Kalman filter which is summarized in Table 7.4. One interesting property to note about the Kalman filter is that the Kalman gain $\mathbf{K}(n)$ and the error covariance matrix $\mathbf{P}(n|n)$ do not depend on the data $\mathbf{x}(n)$. Therefore, it is possible for both of these terms to be computed off-line prior to any filtering.

Example 7.4.1 *Using a Kalman Filter to Estimate an Unknown Constant*

Let us consider the problem of estimating the value of an (unknown) constant x given measurements that are corrupted by uncorrelated, zero mean white noise $v(n)$ that has a variance σ_v^2. Since the value of x does not change with time n, then the state equation is

$$x(n) = x(n-1)$$

The measurement equation, on the other hand, is

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

Therefore, $\mathbf{A}(n) = 1$, $\mathbf{C}(n) = 1$, $\mathbf{Q}_w(n) = 0$, and $\mathbf{Q}_v(n) = \sigma_v^2$. Since $x(n)$ is a scalar, the error covariance is also a scalar and will be denoted by

$$P(n|n) = E\{e^2(n|n)\}$$

Table 7.4 *The Discrete Kalman Filter*

State Equation	$\mathbf{x}(n) = \mathbf{A}(n-1)\mathbf{x}(n-1) + \mathbf{w}(n)$
Observation Equation	$\mathbf{y}(n) = \mathbf{C}(n)\mathbf{x}(n) + \mathbf{v}(n)$
Initialization:	$\hat{\mathbf{x}}(0\|0) = E\{\mathbf{x}(0)\}$
	$\mathbf{P}(0\|0) = E\{\mathbf{x}(0)\mathbf{x}^H(0)\}$
Computation:	For $n = 1, 2, \ldots$ compute
	$\hat{\mathbf{x}}(n\|n-1) = \mathbf{A}(n-1)\hat{\mathbf{x}}(n-1\|n-1)$
	$\mathbf{P}(n\|n-1) = \mathbf{A}(n-1)\mathbf{P}(n-1\|n-1)\mathbf{A}^H(n-1) + \mathbf{Q}_w(n)$
	$\mathbf{K}(n) = \mathbf{P}(n\|n-1)\mathbf{C}^H(n)\Big[\mathbf{C}(n)\mathbf{P}(n\|n-1)\mathbf{C}^H(n) + \mathbf{Q}_v(n)\Big]^{-1}$
	$\hat{\mathbf{x}}(n\|n) = \hat{\mathbf{x}}(n\|n-1) + \mathbf{K}(n)\Big[\mathbf{y}(n) - \mathbf{C}(n)\hat{\mathbf{x}}(n\|n-1)\Big]$
	$\mathbf{P}(n\|n) = \Big[\mathbf{I} - \mathbf{K}(n)\mathbf{C}(n)\Big]\mathbf{P}(n\|n-1)$

where $e(n|n) = x(n) - \hat{x}(n|n)$. From Eq. (7.102) it follows that $P(n|n-1)$ and $P(n-1|n-1)$ are equal,

$$P(n|n-1) = P(n-1|n-1)$$

Therefore, in order to simplify notation we will use $P(n-1)$ to denote both $P(n-1|n-1)$ and $P(n|n-1)$. From the expression for the Kalman gain in Eq. (7.113) we have

$$K(n) = P(n-1)\Big[P(n-1) + \sigma_v^2\Big]^{-1} \tag{7.115}$$

Thus, from Eq. (7.114) it follows that the update for $P(n|n)$ is

$$\begin{aligned} P(n) &= \Big[1 - K(n)\Big]P(n-1) \\ &= \left[1 - \frac{P(n-1)}{P(n-1) + \sigma_v^2}\right]P(n-1) \\ &= \frac{P(n-1)\sigma_v^2}{P(n-1) + \sigma_v^2} \end{aligned}$$

We may solve this difference equation recursively as follows

$$\begin{aligned} P(1) &= \frac{P(0)\sigma_v^2}{P(0) + \sigma_v^2} \\ P(2) &= \frac{P(1)\sigma_v^2}{P(1) + \sigma_v^2} = \frac{P(0)\sigma_v^2}{2P(0) + \sigma_v^2} \\ P(3) &= \frac{P(2)\sigma_v^2}{P(2) + \sigma_v^2} = \frac{P(0)\sigma_v^2}{3P(0) + \sigma_v^2} \\ &\vdots \end{aligned}$$

Thus, for $P(n)$ we have, in general,

$$P(n) = \frac{P(0)\sigma_v^2}{nP(0) + \sigma_v^2}$$

Incorporating this into the expression for the Kalman gain given in Eq. (7.115) we have

$$K(n) = \frac{P(n-1)}{P(n-1)+\sigma_v^2} = \frac{P(0)}{nP(0)+\sigma_v^2}$$

Finally, for the discrete Kalman filter we have

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

Note that as $n \to \infty$ then $K(n) \to 0$ and $\hat{x}(n)$ approaches a steady state value.

There are some special cases of this Kalman filter that are of interest. First, note that if $\sigma_v^2 \to \infty$, which implies that the measurements are completely unreliable, then the Kalman gain goes to zero and the estimate becomes

$$\hat{x}(n) = \hat{x}(n-1)$$

Thus, the measurements are ignored and $\hat{x}(n) = \hat{x}(0)$, the initial estimate, which has an error variance $P(0)$. Second, suppose that $\hat{x}(0) = 0$ and $P(0) \longrightarrow \infty$. This corresponds to the case of no a priori information about x. In this case $K(n) = 1/n$ and the estimate becomes

$$\hat{x}(n) = \hat{x}(n-1) + \frac{1}{n}\big[y(n) - \hat{x}(n-1)\big] = \frac{n-1}{n}\hat{x}(n-1) + \frac{1}{n}y(n)$$

Note, however, that this is simply a recursive implementation of the sample mean,

$$\hat{x}(n) = \frac{1}{n}\sum_{k=1}^{n} y(k)$$

In the next example, we consider the use of a Kalman filter to solve the filtering problem considered in Example 7.3.2. As we will see, the Kalman filter is time-varying due to the fact that, unlike the Wiener filter, the filter begins operation at time $n = 0$. However, since the processes are stationary, after the initial transients have died out, the Kalman filter settles down into its steady-state behavior, which is equivalent to the causal Wiener filter solution.

Example 7.4.2 *Using a Kalman Filter to Estimate an AR(1) Process*

Let $x(n)$ be the AR(1) process

$$x(n) = 0.8x(n-1) + w(n)$$

where $w(n)$ is white noise with a variance $\sigma_w^2 = 0.36$, and let

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

be noisy measurements of $x(n)$ where $v(n)$ is unit variance white noise that is uncorrelated with $w(n)$. Thus, with $\mathbf{A}(n) = 0.8$ and $\mathbf{C}(n) = 1$ the Kalman filter state estimation equation is

$$\hat{x}(n) = 0.8\hat{x}(n-1) + K(n)\big[y(n) - 0.8\hat{x}(n-1)\big]$$

Since the state vector is a scalar, the equations for computing the Kalman gain are scaler equations,

$$\begin{aligned} P(n|n-1) &= (0.8)^2 P(n-1|n-1) + 0.36 \\ K(n) &= P(n|n-1)\big[P(n|n-1)+1\big]^{-1} \\ P(n|n) &= \big[1 - K(n)\big]P(n|n-1) \end{aligned}$$

With $\hat{x}(0) = E\{x(0)\} = 0$ and $P(0|0) = E\{|x(0)|^2\} = 1$, the Kalman gain and the

Table 7.5 The Kalman Gain and Error Covariances

n	$P(n\|n-1)$	$K(n)$	$P(n\|n)$
1	1.0000	0.5000	0.5000
2	0.6800	0.4048	0.4048
3	0.6190	0.3824	0.3824
4	0.6047	0.3768	0.3768
5	0.6012	0.3755	0.3755
6	0.6003	0.3751	0.3751
⋮	⋮	⋮	⋮
∞	0.6000	0.3750	0.3750

error covariances for the first few values of n are shown in Table 7.5. Note that after a few iterations the Kalman filter settles down into its steady state solution

$$\hat{d}(n) = 0.8\hat{d}(n-1) + 0.375\Big[x(n) - 0.8\hat{d}(n-1)\Big]$$

with a final mean-square error of $\xi = 0.375$ which, as we see from our discussion following Example 7.3.2 (p. 364), is identical to the causal Wiener filter.

The goal of the discrete Kalman filter is to use the measurements, $\mathbf{y}(n)$, to estimate the state $\mathbf{x}(n)$ of a dynamic system. The Kalman filter is a remarkably versatile and powerful recursive estimation algorithm that has found applications in a wide variety of different areas including spacecraft orbit determination, radar tracking, estimation and prediction of target trajectories, adaptive equalization of telephone channels, adaptive equalization of fading dispersive channels, and adaptive antenna arrays. In this section our intent was only to provide a brief introduction to the problem of recursive estimation. Since a detailed study on the use, application, and numerical properties of the Kalman filter would fill an entire textbook, we have only touched upon the problem. A more detailed treatment of the discrete Kalman filter may be found in many excellent references such as [2,3,5,6]. In addition, some signal processing applications of Kalman filtering may be found in [4,5,8], and a historical perspective of Kalman filtering is given in [9] which traces its roots back to the invention of least squares theory by Gauss in 1809. Finally, an interesting collection of papers on the theory and application of Kalman filtering may be found in [10].

7.5 SUMMARY

In this chapter, we considered the problem of designing the optimum filter for estimating a process $d(n)$ in terms of measurements of a related process $x(n)$. The first problem that we considered was the design of the optimum FIR filter that minimizes the mean-square estimation error $\xi = E\{|d(n) - \hat{d}(n)|^2\}$. Assuming that $d(n)$ and $x(n)$ are jointly wide-sense stationary processes with known autocorrelation $r_x(k)$ and cross-correlation $r_{dx}(k)$, the solution is given by the Wiener-Hopf equations, which are a set of linear Toeplitz equations. These equations are a generalization of the linear Toeplitz equations derived in Chapter 4 for all-pole signal modeling and, as we saw in Chapter 5, may be solved using the general Levinson recursion. Since the Wiener-Hopf equations apply to the estimation of *any* process $d(n)$,

we then considered the special cases of filtering, linear prediction, and noise cancellation. It was then shown how the FIR Wiener filter could be implemented in lattice filter form.

Next, we considered the design of an IIR Wiener filter. Without imposing a causality constraint on the solution, what we discovered was that these filters are generally noncausal and, therefore, unrealizable. As a result, these filters would not generally be appropriate in real-time signal processing applications. Nevertheless, assuming that the power spectrum of $x(n)$ and the cross-power spectral density between $x(n)$ and $d(n)$ are known, these filters are easily designed and may be used to set an upper bound on the performance of an FIR Wiener filter or a causal IIR Wiener filter.

After looking at the use of a noncausal Wiener filter for smoothing, we then considered the design of a causal IIR Wiener filter. What we found was that, by imposing a causality constraint on the filter, it becomes necessary to perform a spectral factorization of the power spectrum of the input process $x(n)$. As a result, compared with the design of an FIR Wiener filter or a noncausal Wiener filter, these filters are generally much more difficult to design. We then looked at specific examples of designing causal Wiener filters, including filtering, linear prediction, and deconvolution.

Finally, we briefly considered the problem of recursive filtering and derived the discrete Kalman filter. Unlike the Wiener filter, the Kalman filter may by used for nonstationary processes as well as stationary ones, and may be initialized to begin operating at time $n = 0$. The next step is to relax the requirement that the statistics of $x(n)$ and $d(n)$ be known. This is the subject of Chapter 9.

References

1. H. C. Andrews and B. R. Hunt, *Digital Image Restoration*, Prentice-Hall, Englewood Cliffs, NJ, 1977.
2. R. G. Brown, *Introduction to Random Signal Analysis and Kalman Filtering*, John Wiley & Sons, New York, 1983.
3. A. Gelb, Ed., *Applied Optimal Estimation*, M.I.T. Press, Cambridge, MA, 1974.
4. D. Godard, "Channel equalization using a Kalman filter for fast data transmission," *IBS J. Res. Dev.*, vol. 18, pp. 267–273, 1974.
5. S. Haykin, *Adaptive Filter Theory*, Prentice-Hall, Englewood Cliffs, NJ, 1986.
6. T. Kailath, "An innovations approach to least-squares estimation – Part I: Linear filtering in additive noise," *IEEE Trans. Autom. Control*, vol. AC-13, pp. 641–655, 1968.
7. R. E. Kalman, "A new approach to linear filtering and prediction problems", *Trans. ASME, J. Basic Eng.*, Ser. 82D, pp. 35–45, March 1960.
8. R. A. Monzingo and T. W. Miller, *Introduction to Adaptive Arrays*, Wiley-Interscience, New York, 1980.
9. H. W. Sorensen, "Least-squares estimation: From Gauss to Kalman," *IEEE Spectrum*, vol. 7, pp. 63–68, July 1970.
10. H. W. Sorensen, ed., *Kalman Filtering: Theory and Application*, IEEE Press, New York, 1985.
11. N. Wiener, *Extrapolation, Interpolation, and Smoothing of Stationary Time Series with Engineering Applications*, MIT Press, Cambridge, MA, 1949.

7.6 PROBLEMS

7.1. A random process $x(n)$ is generated as follows

$$x(n) = \alpha x(n-1) + v(n) + \beta v(n-1)$$

where $v(n)$ is white noise with mean m_v and variance σ_v^2.

(a) Design a first-order linear predictor

$$\hat{x}(n+1) = w(0)x(n) + w(1)x(n-1)$$

that minimizes the mean-square error in the prediction of $x(n+1)$, and find the minimum mean-square error.

(b) Now consider a predictor of the form

$$\hat{x}(n+1) = c + w(0)x(n) + w(1)x(n-1)$$

Find the values for c, $w(0)$, and $w(1)$ that minimize the mean-square error, and compare the mean-square error of this predictor with that found in part (a).

7.2. In this problem we consider the design of a three-step predictor using a first-order filter

$$W(z) = w(0) + w(1)z^{-1}$$

In other words, with $x(n)$ the input to the predictor $W(z)$, the output

$$\hat{x}(n+3) = w(0)x(n) + w(1)x(n-1)$$

is the minimum mean-square estimate of $x(n+3)$.

(a) What are the Wiener-Hopf equations for the Wiener three-step predictor?

(b) If the values of $r_x(k)$ for lags $k = 0$ to $k = 4$ are

$$\mathbf{r}_x = [1.0,\ 0,\ 0.1,\ -0.2,\ -0.9]^T$$

solve the Wiener-Hopf equations and find the optimum three-step predictor.

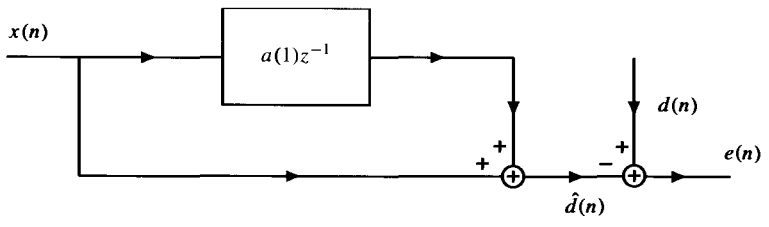
(c) Does the prediction error filter

$$F(z) = 1 + w(0)z^{-3} + w(1)z^{-4}$$

have minimum phase, i.e., are the zeros of $F(z)$ inside the unit circle? How does this compare to what you know about the prediction error filter for a one-step predictor?

7.3. Repeat Example 7.2.1 using a second-order Wiener filter, and compare the mean-square error for the second-order filter with the mean-square error of the first-order filter.

7.4. Consider the system shown in the figure below for estimating a process $d(n)$ from $x(n)$.



If $\sigma_d^2 = 4$ and

$$\mathbf{r}_x = [1.0,\ 0.5,\ 0.25]^T \quad ; \quad \mathbf{r}_{dx} = [-1.0,\ 1.0]^T$$

find the value of $a(1)$ that minimizes the mean-square error $\xi = E\{|e(n)|^2\}$, and find the minimum mean-square error.

7.5. In this problem we consider linear prediction in a noisy environment. Suppose that a signal $d(n)$ is corrupted by noise,

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

where $r_w(k) = 0.5\delta(k)$ and $r_{dw}(k) = 0$. The signal $d(n)$ is an AR(1) process that satisfies the difference equation

$$d(n) = 0.5d(n-1) + v(n)$$

where $v(n)$ is white noise with variance $\sigma_v^2 = 1$. Assume that $w(n)$ and $v(n)$ are uncorrelated.

(a) Design a first-order FIR linear predictor $W(z) = w(0) + w(1)z^{-1}$ for $d(n)$ and find the mean-square prediction error $\xi = E\{[d(n+1) - \hat{d}(n+1)]^2\}$.

(b) Design a causal Wiener predictor and compare the mean-square prediction error with that found in part (a).

7.6. Suppose that a process $x(n)$ has been recorded, but there is a missing gap of data over the interval $[N_1, N_2]$, i.e., $x(n)$ is unknown over this interval.

(a) Derive the optimum estimate of $x(N_1)$ using the data in the semi-infinite interval $(-\infty, N_1 - 1]$.

(b) Derive the optimum estimate of $x(N_1)$ using the data in the semi-infinite interval $[N_2 + 1, \infty)$.

(c) Derive the optimum estimate of $x(N_1)$ that is formed by combining together the two estimates found in parts (a) and (b).

(d) Generalize your result in part (c) to find the optimum estimate of $x(n)$ at an arbitrary point n in the interval $[N_1, N_2]$.

7.7. In this problem we consider the design of a causal IIR Wiener filter for p-step prediction,

$$\hat{x}(n+p) = \sum_{k=0}^{\infty} h(k)x(n-k)$$

(a) If $x(n)$ is a real-valued random process with power spectral density

$$P_x(z) = \sigma_x^2 Q(z)Q(z^{-1})$$

find the system function of the causal Wiener filter that minimizes the mean-square error

$$\xi = E\left\{|\hat{x}(n+p) - x(n+p)|^2\right\}$$

(b) If $x(n)$ is an AR(1) process with power spectrum

$$P_x(z) = \frac{1-a^2}{(1-az^{-1})(1-az)}$$

find the causal p-step linear predictor and evaluate the mean-square error.

(c) If $x(n)$ is an MA(2) process that is generated by the difference equation

$$x(n) = 4v(n) - 2v(n-1) + v(n-2)$$

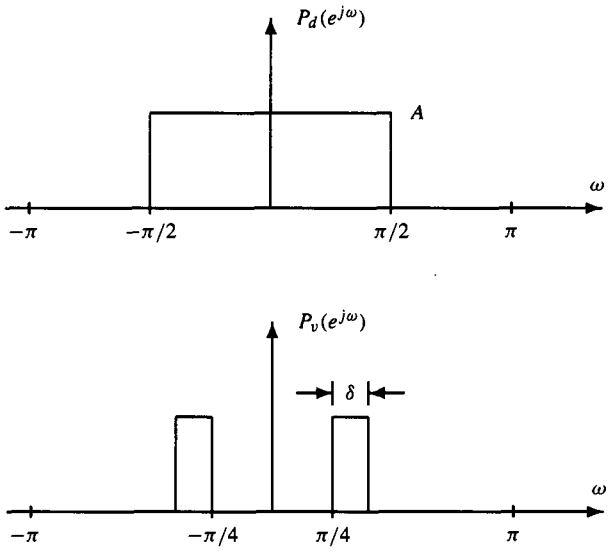
where $v(n)$ is zero mean unit variance white noise, find the system function of the two-step ($p = 2$) predictor and evaluate the mean-square error.

(d) Repeat part (c) for a three-step predictor.

7.8. Suppose that we would like to estimate a process $d(n)$ from the noisy observations

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

where the noise, $v(n)$, is uncorrelated with $d(n)$. The power spectral densities of $d(n)$ and $v(n)$ are shown in the following figure.



(a) Design a *noncausal* Wiener smoothing filter for estimating $d(n)$ from $x(n)$,

$$\hat{d}(n) = \sum_{k=-\infty}^{\infty} h(k)x(n-k)$$

(b) Compute the mean-square error $E\{|d(n) - \hat{d}(n)|^2\}$ and compare it to the mean-square error that results when $h(n) = \delta(n)$, i.e., with no filtering of $x(n)$.

(c) Design a second-order FIR Wiener filter

$$W(z) = w(0) + w(1)z^{-1} + w(2)z^{-2}$$

for estimating $d(n)$ from $x(n)$, and compare the mean-square error in your estimate to that found in parts (a) and (b).

7.9. We would like to estimate a process $d(n)$ from noisy observations,

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

where $v(n)$ is white noise with variance $\sigma_v^2 = 1$ and $d(n)$ is a wide-sense stationary random process with the first four values of the autocorrelation sequence given by

$$\mathbf{r}_x = [1.5,\ 0,\ 1.0,\ 0]^T$$

Assume that $d(n)$ and $v(n)$ are uncorrelated. Our goal is to design an FIR filter to reduce the noise in $d(n)$. Hardware constraints, however, limit the filter to only three nonzero coefficients in $W(z)$.

(a) Derive the optimal three-multiplier causal filter

$$W(z) = w(0) + w(1)z^{-1} + w(2)z^{-2}$$

for estimating $d(n)$, and evaluate the mean-square error $E\{|d(n) - \hat{d}(n)|^2\}$.

(b) Repeat part (a) for the noncausal FIR filter

$$W(z) = w(-1)z + w(0) + w(1)z^{-1}$$

(c) Can you suggest a way to reduce the mean-square error below that obtained for the filters designed in parts (a) and (b) without using any more than three filter coefficients?

7.10. Suppose that a signal $x(n)$ is recorded and that, due to measurement errors, there are *outliers* in the data, i.e., for some values of n there is a large error in the measured value of $x(n)$. Instead of eliminating these data values, suppose that we perform a *minimum mean-square interpolation* as follows. Given a "bad" data value at time $n = n_0$, consider an estimate for $x(n_0)$ of the form

$$\hat{x}(n_0) = ax(n_0 - 1) + bx(n_0 + 1)$$

(a) Assuming that $x(n)$ is a wide-sense stationary random process with autocorrelation sequence $r_x(k)$, find the values for a and b that minimize the mean-square error

$$\xi = E\{|x(n_0) - \hat{x}(n_0)|^2\}$$

(b) If $r_x(k) = (0.5)^{|k|}$, evaluate the mean-square error for the interpolator found in part (a).

(c) Discuss when it may be better to use an estimator of the form

$$\hat{x}(n_0) = ax(n_0 - 1) + bx(n_0 - 2)$$

or explain why such an estimator should not be used.

(d) Given an autocorrelation sequence $r_x(k)$, derive the Wiener-Hopf equations that define the optimum filter for interpolating $x(n)$ to produce the best estimate of $x(n_0)$ in terms of the $2p$ data values

$$x(n_0 - 1), x(n_0 - 2), \ldots, x(n_0 - p) \quad \text{and} \quad x(n_0 + 1), x(n_0 + 2), \ldots, x(n_0 + p)$$

(e) Find an expression for the minimum mean-square error for your estimate in part (d).

7.11. In this problem we consider the design of an optimum smoothing filter for estimating a process $d(n)$ from the measurements

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

Our goal is to use a noncausal FIR filter that has a system function of the form:

$$W(z) = \sum_{k=-p}^{p} w(k)z^{-k}$$

In other words, we want to produce an estimate of $d(n)$ as follows

$$\hat{d}(n) = \sum_{k=-p}^{p} w(k)x(n - k)$$

(a) Derive the Wiener-Hopf equations that define the set of coefficients that minimize the mean-square error

$$\xi = E\{|d(n) - \hat{d}(n)|^2\}$$

(b) How would the Wiener-Hopf equations derived in part (a) change if we used a causal filter with the same number of coefficients? In other words, if the system function were of the form

$$W(z) = \sum_{k=0}^{2p} w(k)z^{-k}$$

how would you modify your equations in (a)?

(c) State qualitatively when you might prefer the noncausal filter over the causal filter and vice versa. For example, for what types of signals and for what types of noise would you expect a causal filter to be superior to the noncausal filter?

(d) FIR digital filters with linear phase (or zero phase) are important in signal processing applications where frequency dispersion due to nonlinear phase is harmful. An FIR filter with zero phase is characterized by the property that

$$w(n) = w(-n)$$

Thus, the system function may be written as

$$W(z) = w(0) + \sum_{k=1}^{p} w(k)[z^{-k} + z^{k}]$$

Derive the Wiener-Hopf equations that define the optimum zero phase smoothing filter.

(e) With $r_d(k) = 4(0.5)^{|k|}$ and $r_v(k) = \delta(k)$, find the optimum values for the filter coefficients $w(0)$ and $w(1)$ in the zero phase filter

$$W(z) = w(0) + w(1)[z + z^{-1}]$$

7.12. We observe a signal, $x(n)$, in a noisy and reverberant environment,

$$y(n) = x(n) + 0.8x(n-1) + v(n)$$

where $v(n)$ is white noise with variance $\sigma_w^2 = 1$ that is uncorrelated with $x(n)$. We know that $x(n)$ is a wide-sense stationary AR(1) random process with autocorrelation values

$$\mathbf{r}_x = [4,\ 2,\ 1,\ 0.5]^T$$

(a) Find the noncausal IIR Wiener filter, $H(z)$, that produces the minimum mean-square estimate of $x(n)$.

(b) Design a causal IIR Wiener filter, $H(z)$, that produces the minimum mean-square estimate of $x(n)$.

7.13. A wide-sense stationary random process has an autocorrelation sequence of the form

$$r_x(k) = \sigma_x^2 \alpha^{|k|}$$

where $|\alpha| < 1$. Over a given time interval, $[n_A, n_B]$, the process $x(n)$ is only known at the end points, i.e., the only given data is $x(n_A)$ and $x(n_B)$. Based on these two observations, determine the optimum estimate

$$\hat{x}(n) = a(n)x(n_A) + b(n)x(n_B)$$

of $x(n)$ over each of the following intervals

(a) $n > n_B$.

(b) $n < n_A$.

(c) $n_A < n < n_B$.

7.14. As shown in Figure 7.12, the Wiener filter may be viewed as a cascade of a whitening filter with a causal filter that produces the minimum mean-square estimate of $d(n)$ from $\epsilon(n)$. For real processes, the system function of the cascade is

$$H(z) = F(z)G(z) = \frac{1}{\sigma_0^2 Q(z)} \left[\frac{P_{dx}(z)}{Q(z^{-1})} \right]_+$$

and the mean-square error is

$$\xi_{\min} = r_d(0) - \sum_{l=0}^{\infty} h(l) r_{dx}(l)$$

(a) If $r_{d\epsilon}(k) = \delta(k)$ and

$$P_x(e^{j\omega}) = \frac{4}{(1 - 0.5z^{-1})(1 - 0.5z)}$$

find the unit sample response, $h(n)$, of the causal Wiener filter.

(b) Derive an expression for the mean-square error that expresses $\xi_{\min}$ in terms of the cross-correlation, $r_{d\epsilon}(k)$, and evaluate the mean-square error when

$$r_{d\epsilon}(k) = (\tfrac{1}{2})^k u(k) + (\tfrac{1}{3})^{-k} u(-k-1)$$

and

$$E\{d^2(n)\} = 4$$

7.15. Let $x(n)$ be an AR(1) process of the following form

$$x(n) = a(1)x(n-1) + b(0)w(n)$$

where $w(n)$ is unit variance white noise, and let $y(n)$ be noisy measurements

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

where $v(n)$ is unit variance white noise that is uncorrelated with $w(n)$. We have seen that the causal Wiener filter for estimating $x(n)$ from $y(n)$ has the form

$$\hat{x}(n) = a(1)\hat{x}(n-1) + K\Big[y(n) - a(1)\hat{x}(n-1)\Big]$$

Find the value of K in terms of $a(1)$ and $b(0)$ that minimizes the mean-square error

$$E\left\{[x(n) - \hat{x}(n)]^2\right\}$$

7.16. In the derivation of the Kalman filtering equations, we made use of the following matrix differentiation formulas

$$\frac{d}{d\mathbf{K}} \mathrm{tr}(\mathbf{KA}) = \mathbf{A}^H$$

and

$$\frac{d}{d\mathbf{K}} \mathrm{tr}(\mathbf{KAK}^H) = 2\mathbf{KA}$$