PART 2
Linear Optimum Filtering

Part II of the book consists of Chapters 5 through 7. It is devoted to a detailed treatment of linear optimum filter theory for discrete-time wide-sense stationary stochastic processes. Adaptive filters are derived from this theory. Chapter 5 covers the classical Wiener filter. Chapter 6 builds on the Wiener filter theory to solve the linear prediction problem. Chapter 7 covers the classical Kalman filter for solving the optimum filtering problem (formulated in terms of a state vector) in a recursive manner.

CHAPTER

5

Wiener Filters

This chapter deals with a class of *linear optimum discrete-time filters* known collectively as *Wiener filters*. The theory for a Wiener filter is formulated for the general case of *complex-valued* time series with the filter specified in terms of its impulse response. The reason for using complex-valued time series is that in many practical situations (e.g., communications, radar, sonar) the *baseband* signal of interest appears in complex form; the term *baseband* is used to designate a band of frequencies representing the original signal as delivered by a source of information. The case of real-valued time series may of course be considered as a special case of this theory.

We begin our study of Wiener filters by outlining the linear optimum filtering problem and setting the stage for the rest of the chapter.

5.1 LINEAR OPTIMUM FILTERING: PROBLEM STATEMENT

Consider the block diagram of Fig. 5.1 built around a *linear discrete-time filter*. The filter *input* consists of a *time series* $u(0), u(1), u(2), \ldots$, and the filter is itself characterized by the *impulse response* $w_0, w_1, w_2, \ldots$. At some *discrete time* n, the filter produces an *output* denoted by $y(n)$. This output is used to provide an *estimate* of a *desired response* denoted by $d(n)$. With the filter input and the desired response representing single realizations of respective stochastic processes, the estimation is accompanied by an error with

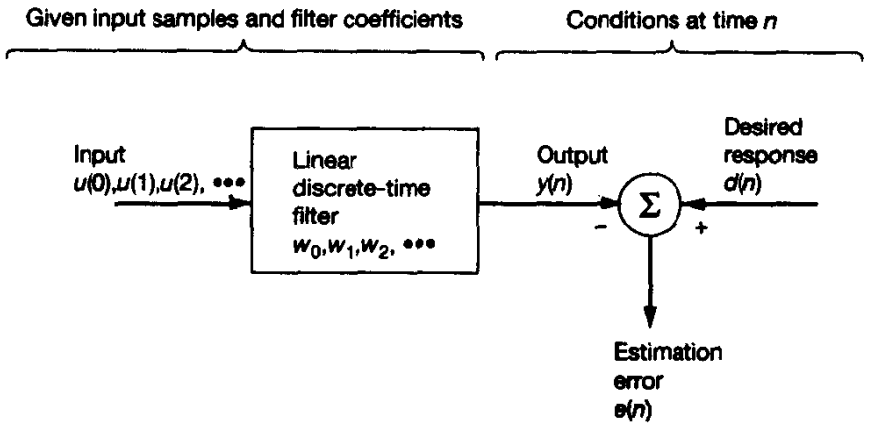


Figure 5.1 Block diagram representation of the statistical filtering problem.

statistical characteristics of its own. In particular, the *estimation error*, denoted by $e(n)$, is defined as the difference between the desired response $d(n)$ and the filter output $y(n)$. The requirement is to make the estimation error $e(n)$ "as small as possible" in some statistical sense.

Two restrictions have so far been placed on the filter:

1. The filter is *linear*, which makes the mathematical analysis easy to handle.
2. The filter operates in *discrete time*, which makes it possible for the filter to be implemented using digital hardware/software.

The final details of the filter specification, however, depend on two other choices that have to be made:

1. Whether the impulse response of the filter has *finite* or *infinite* duration.
2. The type of *statistical criterion* used for the optimization.

The choice of a *finite-duration impulse response* (FIR) or an *infinite-duration impulse response* (IIR) for the filter is dictated by *practical considerations*. The choice of a statistical criterion for optimizing the filter design is influenced by *mathematical tractability*. These two issues are considered in turn.

For the initial development of the Wiener filter theory, we will assume an IIR filter; the theory so developed includes that for FIR filters as a special case. However, for much of the material presented in this chapter, and indeed in the rest of the book, we will confine our attention to the use of FIR filters. We do so for the following reason. An FIR filter is *inherently stable*, because its structure involves the use of *forward paths only*. In other words, the only mechanism for input–output interaction in the filter is via forward paths from the filter input to its output. Indeed, it is this form of signal transmission through the filter that limits its impulse response to a finite duration. On the other hand, an IIR filter involves *both feedforward and feedback*. The presence of feedback means that

portions of the filter output and possibly other *internal* variables in the filter are fed back to the input. Consequently, unless it is properly designed, feedback in the filter can indeed make it *unstable* with the result that the filter *oscillates*; this kind of operation is clearly unacceptable when the requirement is that of filtering for which stability is a "must." By itself, the stability problem in IIR filters is manageable in both theoretical and practical terms. However, when the filter is required to be *adaptive*, bringing with it stability problems of its own, the inclusion of adaptivity combined with feedback that is inherently present in an IIR filter makes a difficult problem that much more difficult to handle. It is for this reason that we find that in the majority of applications requiring the use of adaptivity, the use of an FIR filter is preferred over an IIR filter even though the latter is less demanding in computational requirements.

Turning next to the issue of what criterion to choose for statistical optimization, there are indeed several criteria that suggest themselves. Specifically, we may consider optimizing the filter design by *minimizing a cost function*, or *index of performance*, selected from the following short list of possibilities:

1. Mean-square value of the estimation error
2. Expectation of the absolute value of the estimation error
3. Expectation of third or higher powers of the absolute value of the estimation error

Option 1 has a clear advantage over the other two, because it leads to tractable mathematics. In particular, the choice of *the mean-square error criterion results in a second-order dependence for the cost function on the unknown coefficients in the impulse response of the filter. Moreover, the cost function has a distinct minimum that uniquely defines the optimum statistical design of the filter.*

We may now summarize the essence of the filtering problem by making the following statement:

Design a linear discrete-time filter whose output $y(n)$ provides an estimate of a desired response $d(n)$, given a set of input samples $u(0), u(1), u(2), \ldots$, such that the mean-square value of the estimation error $e(n)$, defined as the difference between the desired response $d(n)$ and the actual response $y(n)$, is minimized.

We may develop the mathematical solution to this statistical optimization problem by following two entirely different approaches that are complementary. One approach leads to the development of an important theorem commonly known as the principle of orthogonality. The other approach highlights the error-performance surface that describes the second-order dependence of the cost function on the filter coefficients. We will proceed by deriving the principle of orthogonality first, because the derivation is relatively simple and because the principle of orthogonality is highly insightful.

5.2 PRINCIPLE OF ORTHOGONALITY

Consider again the statistical filtering problem described in Fig. 5.1. The filter input is denoted by the time series $u(0), u(1), u(2), \ldots$, and the impulse response of the filter is denoted by $w_0, w_1, w_2, \ldots$, both of which are assumed to have *complex values* and *infinite duration*. The filter output $y(n)$ at discrete time n is defined by the *linear convolution sum*:

$$y(n) = \sum_{k=0}^{\infty} w_k^* u(n-k), \qquad n = 0, 1, 2, \ldots \tag{5.1}$$

where the asterisk denotes *complex conjugation*. Note that in complex terminology, the term $w_k^* u(n-k)$ represents the scalar version of an *inner product of the filter coefficient w_k and the filter input $u(n-k)$*. Figure 5.2 illustrates the steps involved in computing the linear discrete-time form of convolution described in Eq. (5.1) for real data .

The purpose of the filter in Fig. 5.1 is to produce an estimate of the desired response $d(n)$. We assume that the filter input and the desired response are single realizations of *jointly wide-sense stationary stochastic processes*, both with zero mean. Accordingly, the estimation of $d(n)$ is accompanied by an error, defined by the difference

$$e(n) = d(n) - y(n) \tag{5.2}$$

The estimation error $e(n)$ is the sample value of a random variable. *To optimize the filter design, we choose to minimize the mean-square value of the estimation error $e(n)$*. We may thus define the cost function as the *mean-squared error*

$$\begin{aligned} J &= E[e(n)e^*(n)] \\ &= E[|e(n)|^2] \end{aligned} \tag{5.3}$$

where E denotes the *statistical expectation operator*. The problem is therefore to determine the operating conditions for which J attains its minimum value.

For complex input data, the filter coefficients are in general complex, too. Let the kth filter coefficient w_k be denoted in terms of its real and imaginary parts as follows:

$$w_k = a_k + jb_k, \qquad k = 0, 1, 2, \ldots \tag{5.4}$$

Correspondingly, we may define a *gradient operator* ∇, the kth element of which is written in terms of first-order partial derivatives with respect to the real part a_k and the imaginary part b_k, for the kth filter coefficient, as

$$\nabla_k = \frac{\partial}{\partial a_k} + j\frac{\partial}{\partial b_k}, \qquad k = 0, 1, 2, \ldots \tag{5.5}$$

Thus, for the situation at hand, applying the operator ∇ to the cost function J, we obtain a multidimensional complex *gradient vector* ∇J, the kth element of which is

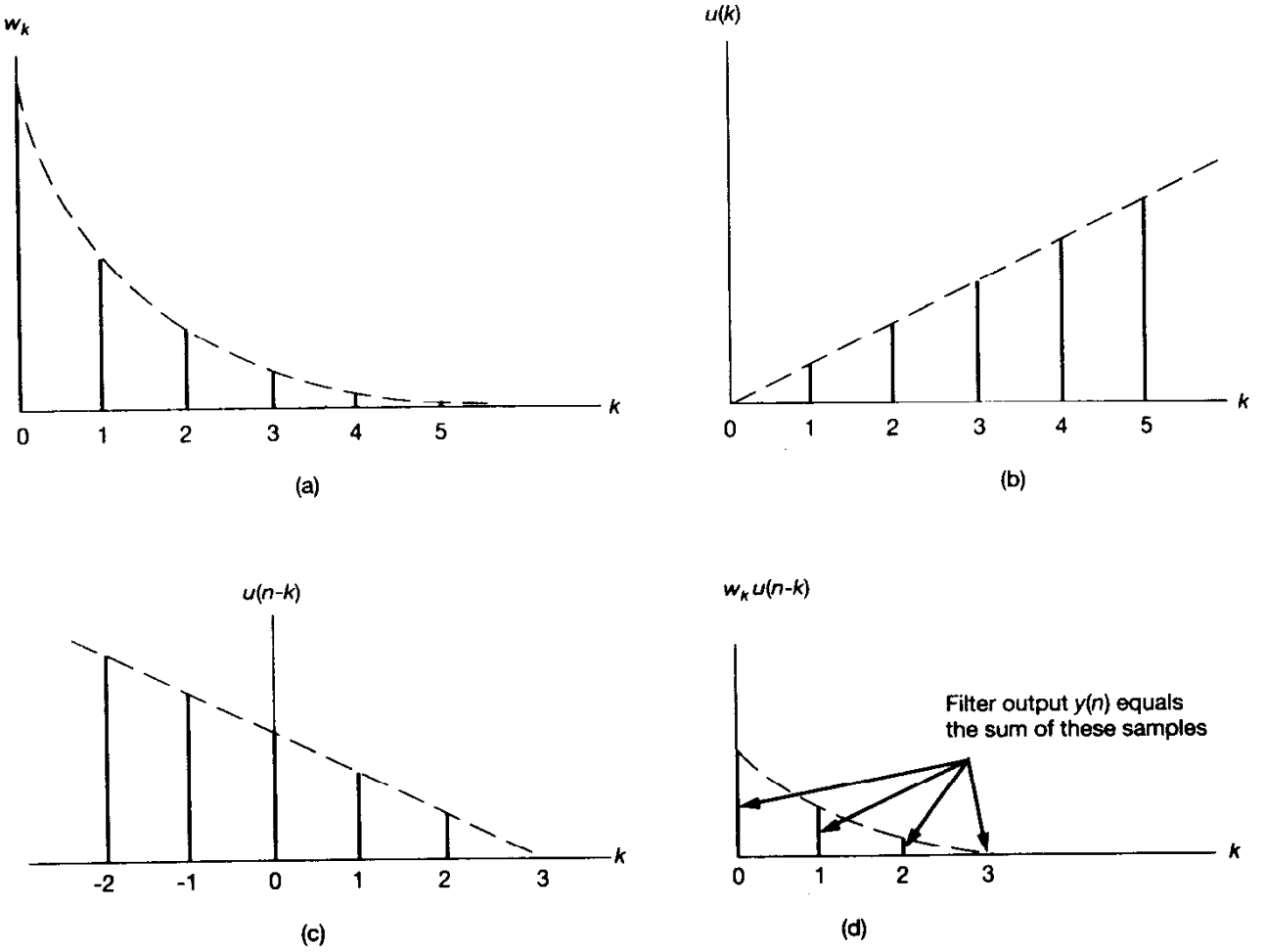


Figure 5.2 Linear convolution: (a) impulse response; (b) filter input; (c) time-reversed and shifted version of filter input; (d) calculation of filter output at time $n = 3$.

$$\nabla_k J = \frac{\partial J}{\partial a_k} + j\frac{\partial J}{\partial b_k}, \qquad k = 0, 1, 2, \ldots \tag{5.6}$$

Equation (5.6) represents a natural extension of the customary definition of a gradient for a function of real coefficients to the more general case of a function of complex coefficients.[1] Note that for the definition of the complex gradient given in Eq. (5.6) to be valid, it is essential that J be *real*. The gradient operator is always used in the context of finding the *stationary points* of a function of interest. This means that a complex constraint must be converted to a pair of *real* constraints. In Eq. (5.6), the pair of real constraints are obtained by setting both the real and imaginary parts of $\nabla_k J$ equal to zero.

For the cost function J to attain its minimum value, all the elements of the gradient vector ∇J must be simultaneously equal to zero, as shown by

$$\nabla_k J = 0, \qquad k = 0, 1, 2, \ldots \tag{5.7}$$

Under this set of conditions, the filter is said to be *optimum in the mean-squared-error sense.*[2]

According to Eq. (5.3), the cost function J is a scalar independent of time n. Hence, substituting the first line of Eq. (5.3) in (5.6), we get

$$\nabla_k J = E\left[\frac{\partial e(n)}{\partial a_k} e^*(n) + \frac{\partial e^*(n)}{\partial a_k} e(n) + \frac{\partial e(n)}{\partial b_k} je^*(n) + \frac{\partial e^*(n)}{\partial b_k} je(n)\right] \tag{5.8}$$

Using Eqs. (5.2) and (5.4), we get the following partial derivatives:

$$\begin{aligned}
\frac{\partial e(n)}{\partial a_k} &= -u(n-k) \\
\frac{\partial e(n)}{\partial b_k} &= ju(n-k) \\
\frac{\partial e^*(n)}{\partial a_k} &= -u^*(n-k) \\
\frac{\partial e^*(n)}{\partial b_k} &= -ju^*(n-k)
\end{aligned} \tag{5.9}$$

[1] The concept of a gradient is commonly discussed in books on optimization (see, e.g., Dorny, 1975). For the complex case, it is discussed in Widrow et al. (1975a) and Monzingo and Miller (1980).

Note that the cost function J, for the general case of complex data, is *not* an analytic function (see Problem 1). Hence the definition of the derivative of the cost function J with respect to a filter coefficient w_k, say, requires particular attention. This issue is discussed in Appendix B. In this appendix we also discuss the relationship between the concepts of a gradient and a derivative for the case of complex coefficients.

[2] Note that in Eq. (5.7), we have presumed optimality at a stationary point. In the linear filtering problem, finding a stationary point assures global optimization of the filter by virtue of the quadratic nature of the error-performance surface; see Section 5.5.

Thus, substituting these partial derivatives in Eq. (5.8) and then canceling common terms, we finally get the result

$$\nabla_k J = -2E[u(n-k)e^*(n)] \tag{5.10}$$

We are now ready to specify the operating conditions required for minimizing the cost function J. Let e_o *denote the special value of the estimation error that results when the filter operates in its optimum condition.* We then find that the conditions specified in Eq. (5.7) are indeed equivalent to

$$E[u(n-k)e_o^*(n)] = 0, \qquad k = 0, 1, 2, \ldots \tag{5.11}$$

In words, Eq. (5.11) states the following:

The necessary and sufficient condition for the cost function J to attain its minimum value is that the corresponding value of the estimation error $e_o(n)$ is orthogonal to each input sample that enters into the estimation of the desired response at time n.

Indeed, this statement constitutes the *principle of orthogonality*; it represents one of the most elegant theorems in the subject of linear optimum filtering. It also provides the mathematical basis of a procedure for testing that the linear filter is operating in its optimum condition.

Corollary to the Principle of Orthogonality

There is a corollary to the principle of orthogonality that we may derive by examining the correlation between the filter output $y(n)$ and the estimation error $e(n)$. Using Eq. (5.1), we may express this correlation as follows:

$$\begin{aligned} E[y(n)e^*(n)] &= E\left[\sum_{k=0}^{\infty} w_k^* u(n-k)e^*(n)\right] \\ &= \sum_{k=0}^{\infty} w_k^* E[u(n-k)e^*(n)] \end{aligned} \tag{5.12}$$

Let y_o denote the output produced by the filter optimized in the mean-squared-error sense, with $e_o(n)$ denoting the corresponding estimation error. Hence, using the principle of orthogonality described by Eq. (5.11), we get the desired result:

$$E[y_o(n)e_o^*(n)] = 0 \tag{5.13}$$

We may thus state the corollary to the principle of orthogonality as follows:

When the filter operates in its optimum condition, the estimate of the desired response defined by the filter output, $y_o(n)$, and the corresponding estimation error, $e_o(n)$, are orthogonal to each other.

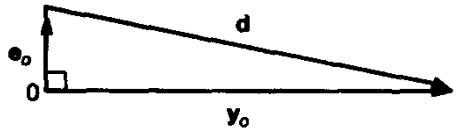


Figure 5.3 Geometric interpretation of the relationship between the desired response, the estimate at the filter output, and the estimation error.

Let $\hat{d}(n|\mathcal{U}_n)$ *denote the estimate of the desired response that is optimized in the mean-squared-error sense, given the input data that span the space* $\mathcal{U}_n$ *up to and including time* n.[3] We may then write

$$\hat{d}(n|\mathcal{U}_n) = y_o(n) \tag{5.14}$$

Note that the estimate $\hat{d}(n|\mathcal{U}_n)$ has zero mean, because the tap inputs are assumed to have zero mean. This condition matches the assumed zero mean of the desired response $d(n)$.

Geometric Interpretation of the Corollary to the Principle of Orthogonality

Equation (5.13) offers an interesting geometric interpretation of the conditions that exist at the output of the optimum filter, as illustrated in Fig. 5.3. In this figure, the desired response, the filter output, and the corresponding estimation error are represented by vectors labeled $\mathbf{d}$, $\mathbf{y}_o$, and $\mathbf{e}_o$, respectively; the subscript o in $\mathbf{y}_o$ and $\mathbf{e}_o$ refers to the optimum condition. We see that for the optimum filter the vector representing the estimation error is *normal* (i.e., perpendicular) to the vector representing the filter output. It should, however, be emphasized that the situation depicted in Fig. 5.3 is merely an *analogy*, where random variables and expectations are replaced with vectors and vector inner products, respectively. Also, for obvious reasons the geometry depicted in this figure may be viewed as a *Statistician's Pythagorean theorem* (Scharf and Thomas, 1995).

5.3 MINIMUM MEAN-SQUARED ERROR

When the linear discrete-time filter in Fig. 5.1 operates in its optimum condition, Eq. (5.2) takes on the following special form

$$\begin{aligned} e_o(n) &= d(n) - y_o(n) \\ &= d(n) - \hat{d}(n|\mathcal{U}_n) \end{aligned} \tag{5.15}$$

[3] If a space $\mathcal{U}_n$ consists of all linear combinations of random variables, $u_1, u_2, \ldots, u_n$, then these random variables are said to *span* that particular space. In other words, every random variable in $\mathcal{U}_n$ can be expressed as some combination of the u's, as shown by

$$u = w_1^* u_1 + \ldots + w_n^* u_n$$

for some coefficients w_n. This assumes that the space $\mathcal{U}_n$ has a finite dimension.

where, in the second line, we have made use of Eq. (5.14). Rearranging the terms in Eq. (5.15), we have

$$d(n) = \hat{d}(n|\mathcal{U}_n) + e_o(n) \tag{5.16}$$

Let $J_{\min}$ denote the minimum mean-squared error, defined by

$$J_{\min} = E[|e_o(n)|^2] \tag{5.17}$$

Hence, evaluating the mean-square values of both sides of Eq. (5.16), and applying to it the corollary to the principle of orthogonality described by Eqs. (5.13) and (5.14), we get

$$\sigma_d^2 = \sigma_{\hat{d}}^2 + J_{\min} \tag{5.18}$$

where σ_d^2 is the variance of the desired response, and $\sigma_{\hat{d}}^2$ is the variance of the estimate $\hat{d}(n|\mathcal{U}_n)$; both of these random variables are assumed to be of zero mean. Solving Eq. (5.18) for the minimum mean-squared error, we get

$$J_{\min} = \sigma_d^2 - \sigma_{\hat{d}}^2 \tag{5.19}$$

This relation shows that for the optimum filter, the minimum mean-squared error equals the difference between the variance of the desired response and the variance of the estimate that the filter produces at its output.

It is convenient to normalize the expression in Eq. (5.19) in such a way that the minimum value of the mean-squared error always lies between zero and one. We may do this by dividing both sides of Eq. (5.19) by σ_d^2, obtaining

$$\frac{J_{\min}}{\sigma_d^2} = 1 - \frac{\sigma_{\hat{d}}^2}{\sigma_d^2} \tag{5.20}$$

Clearly, this is possible because σ_d^2 is never zero, except in the trivial case of a desired response $d(n)$ that is zero for all n. Let

$$\epsilon = \frac{J_{\min}}{\sigma_d^2} \tag{5.21}$$

The quantity ϵ is called the *normalized mean-squared error*, in terms of which we may rewrite Eq. (5.20) in the form

$$\epsilon = 1 - \frac{\sigma_{\hat{d}}^2}{\sigma_d^2} \tag{5.22}$$

We note that (1) the ratio ϵ can never be negative, and (2) the ratio $\sigma_{\hat{d}}^2/\sigma_d^2$ is always positive. We therefore have

$$0 \leq \epsilon \leq 1 \tag{5.23}$$

If ϵ is zero, the optimum filter operates perfectly in the sense that there is complete agreement between the estimate $\hat{d}(n|\mathcal{U}_n)$ at the filter output and the desired response $d(n)$. On the other hand, if ϵ is unity, there is no agreement whatsoever between these two quantities; this corresponds to the worst possible situation.

5.4 WIENER–HOPF EQUATIONS

The principle of orthogonality, described in Eq. (5.11), specifies the necessary and sufficient condition for the optimum operation of the filter. We may reformulate the necessary and sufficient condition for optimality by substituting Eqs. (5.1) and (5.2) in (5.11). In particular, we may write

$$E\left[u(n-k)\left(d^*(n)-\sum_{i=0}^{\infty} w_{oi}u^*(n-i)\right)\right]=0, \qquad k=0,1,2,\ldots$$

where w_{oi} is the ith coefficient in the impulse response of the optimum filter. Expanding this equation and rearranging terms, we get

$$\sum_{i=0}^{\infty} w_{oi}E[u(n-k)u^*(n-i)]=E[u(n-k)d^*(n)], \qquad k=0,1,2,\ldots \tag{5.24}$$

The two expectations in Eq. (5.24) may be interpreted as follows:

1. The expectation $E[u(n-k)u^*(n-i)]$ is equal to the *autocorrelation function of the filter input* for a lag of $i-k$. We may thus express this expectation as

$$r(i-k)=E[u(n-k)u^*(n-i)] \tag{5.25}$$

2. The expectation $E[u(n-k)d^*(n)]$ is equal to the cross-correlation between the filter input $u(n-k)$ and the desired response $d(n)$ for a lag of $-k$. We may thus express this second expectation as

$$p(-k)=E[u(n-k)d^*(n)] \tag{5.26}$$

Accordingly, using the definitions of Eqs. (5.25) and (5.26) in (5.24), we get an infinitely large system of equations as the necessary and sufficient condition for the optimality of the filter:

$$\sum_{i=0}^{\infty} w_{oi}r(i-k)=p(-k), \qquad k=0,1,2,\ldots \tag{5.27}$$

The system of equations (5.27) defines the optimum filter coefficients, in the most general setting, in terms of two correlation functions: the autocorrelation function of the filter input, and the cross-correlation between the filter input and the desired response. These equations are called the *Wiener–Hopf equations.*[4]

[4] In order to solve the Wiener–Hopf equations for the optimum filter coefficients, we need to use a special technique known as *spectral factorization*. For a description of this technique and its use in solving the Wiener–Hopf equations (5.27), the interested reader is referred to Haykin (1989).

It should also be noted that the defining equation for a linear optimum filter was formulated originally by Wiener and Hopf (1931) for the case of a continuous-time filter, whereas, of course the system of equations (5.27) is formulated for a discrete-time filter.

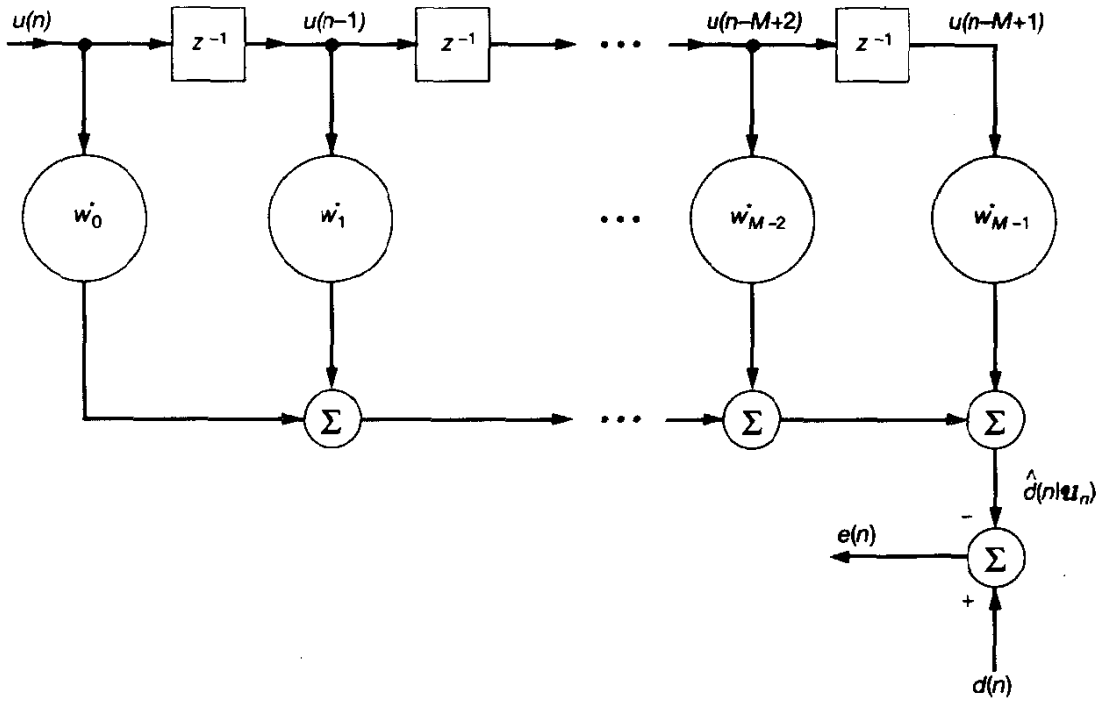


Figure 5.4 Transversal filter.

Solution of the Wiener–Hopf Equations for Linear Transversal Filters

The solution of the set of Wiener–Hopf equations is greatly simplified for the special case when a *linear transversal filter*, or FIR filter, is used to perform the estimation of desired response $d(n)$ in Fig. 5.1. Consider then the structure shown in Fig. 5.4. The transversal filter involves a combination of three basic operations: *storage*, *multiplication*, and *addition*, as described here:

1. The storage is represented by a cascade of $M - 1$ *one-sample delays*, with the block for each such unit labeled as z^{-1}. We refer to the various points at which the one-sample delays are accessed as *tap points*. The tap inputs are denoted by $u(n)$, $u(n-1), \ldots, u(n-M+1)$. Thus, with $u(n)$ viewed as the *current* value of the filter input, the remaining $M - 1$ tap inputs, $u(n-1), \ldots, u(n-M+1)$, represent *past values of the input*.
2. The scalar *inner products* of tap inputs $u(n)$, $u(n-1), \ldots, u(n-M+1)$ and *tap weights* $w_0, w_1, \ldots, w_{M-1}$, respectively, are formed by using a correspond-

ing set of multipliers. In particular, the multiplication involved in forming the scalar inner product of $u(n)$ and w_0 is represented by a block labeled w_0^*, and so on for the other inner products.

3. The function of the adders is to sum the multiplier outputs to produce an overall output for the filter.

The impulse response of the transversal filter in Fig. 5.4 is defined by the finite set of tap weights $w_0, w_1, \ldots, w_{M-1}$. Accordingly, the Wiener–Hopf equations (5.27) reduce to a *system of M simultaneous equations*, as shown by

$$\sum_{i=0}^{M-1} w_{oi} r(i-k) = p(-k), \qquad k = 0, 1, \ldots, M-1 \tag{5.28}$$

where $w_{o0}, w_{o1}, \ldots, w_{o,M-1}$ are the optimum values of the tap weights of the filter.

Matrix Formulation of the Wiener–Hopf Equations

Let $\mathbf{R}$ denote the M-by-M *correlation matrix* of the tap inputs $u(n)$, $u(n-1)$, $\ldots$, $u(n-M+1)$ in the transversal filter of Fig. 5.4:

$$\mathbf{R} = E[\mathbf{u}(n)\mathbf{u}^H(n)] \tag{5.29}$$

where $\mathbf{u}(n)$ is the M-by-1 tap-input vector:

$$\mathbf{u}(n) = [u(n), u(n-1), \ldots, u(n-M+1)]^T \tag{5.30}$$

In expanded form, we have

$$\mathbf{R} = \begin{bmatrix} r(0) & r(1) & \cdots & r(M-1) \\ r^*(1) & r(0) & \cdots & r(M-2) \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ r^*(M-1) & r^*(M-2) & \cdots & r(0) \end{bmatrix} \tag{5.31}$$

Correspondingly, let $\mathbf{p}$ denote the M-by-1 *cross-correlation vector* between the tap inputs of the filter and the desired response $d(n)$:

$$\mathbf{p} = E[\mathbf{u}(n)d^*(n)] \tag{5.32}$$

In expanded form, we have

$$\mathbf{p} = [p(0), p(-1), \ldots, p(1-M)]^T \tag{5.33}$$

Note that the lags used in the definition of $\mathbf{p}$ are either zero or else negative. We may thus rewrite the Wiener–Hopf equations (5.28) in the compact matrix form:

$$\mathbf{R}\mathbf{w}_o = \mathbf{p} \tag{5.34}$$

where $\mathbf{w}_o$ denotes the M-by-1 *optimum tap-weight vector* of the transversal filter; that is,

$$\mathbf{w}_o = [w_{o0}, w_{o1}, \ldots, w_{o,M-1}]^T \tag{5.35}$$

To solve the Wiener–Hopf equations (5.34) for $\mathbf{w}_o$, we assume that the correlation matrix $\mathbf{R}$ is nonsingular. We may then premultiply both sides of Eq. (5.34) by $\mathbf{R}^{-1}$, the *inverse* of the correlation matrix, obtaining

$$\mathbf{w}_o = \mathbf{R}^{-1}\mathbf{p} \tag{5.36}$$

The computation of the optimum tap-weight vector $\mathbf{w}_o$ requires knowledge of two quantities: (1) the correlation matrix $\mathbf{R}$ of the tap-input vector $\mathbf{u}(n)$, and (2) the cross-correlation vector $\mathbf{p}$ between the tap-input vector $\mathbf{u}(n)$ and the desired response $d(n)$.

5.5 ERROR-PERFORMANCE SURFACE

The Wiener–Hopf equations (5.34), as derived above, are traceable to the principle of orthogonality, which itself was derived in Section 5.2. We may also derive the Wiener–Hopf equations by examining the dependence of the cost function J on the tap weights of the transversal filter in Fig. 5.4. First, we write the estimation error $e(n)$ as follows:

$$e(n) = d(n) - \sum_{k=0}^{M-1} w_k^* u(n-k) \tag{5.37}$$

where $d(n)$ is the desired response; $w_0, w_1, \ldots, w_{M-1}$ are the tap weights of the filter; and $u(n), u(n-1), \ldots, u(n-M+1)$ are the corresponding tap inputs. Accordingly, we may define the cost function for the transversal filter structure of Fig. 5.4 as

$$\begin{aligned} J &= E[e(n)e^*(n)] \\ &= E[|d(n)|^2] - \sum_{k=0}^{M-1} w_k^* E[u(n-k)d^*(n)] - \sum_{k=0}^{M-1} w_k E[u^*(n-k)d(n)] \\ &\quad + \sum_{k=0}^{M-1}\sum_{i=0}^{M-1} w_k^* w_i E[u(n-k)u^*(n-i)] \end{aligned} \tag{5.38}$$

We may now recognize the four expectations on the right-hand side of the second line in Eq. (5.38), as follows:

1. For the first expectation, we have

$$\sigma_d^2 = E[|d(n)|^2] \tag{5.39}$$

where σ_d^2 is the *variance* of the desired response $d(n)$, assumed to be of zero mean.

2. For the second and third expectations, we have, respectively,

$$p(-k) = E[u(n-k)d^*(n)] \tag{5.40}$$

and

$$p^*(-k) = E[u^*(n-k)d(n)] \tag{5.41}$$

where $p(-k)$ is the cross-correlation between the tap input $u(n-k)$ and the desired response $d(n)$.

3. Finally, for the fourth expectation, we have

$$r(i-k) = E[u(n-k)u^*(n-i)] \tag{5.42}$$

where $r(i-k)$ is the autocorrelation function of the tap inputs for lag $i-k$.

We may thus rewrite Eq. (5.38) in the form

$$J = \sigma_d^2 - \sum_{k=0}^{M-1} w_k^* p(-k) + \sum_{k=0}^{M-1} w_k^* p^*(-k) + \sum_{k=0}^{M-1}\sum_{i=0}^{M-1} w_k^* w_i r(i-k) \tag{5.43}$$

Equation (5.43) states that for the case when the tap inputs of the transversal filter and the desired response are jointly stationary, the cost function, or mean-squared error, J is precisely a *second-order function of the tap weights in the filter*. Consequently, we may visualize the dependence of the cost function J on the tap weights $w_0, w_1, \ldots, w_{M-1}$ as a *bowl-shaped* $(M+1)$*-dimensional surface* with M *degrees of freedom represented by the tap weights of the filter*. This surface is characterized by a unique minimum. We refer to the surface so described as the *error-performance surface* of the transversal filter in Fig. 5.4.

At the *bottom* or *minimum point* of the error-performance surface, the cost function J attains its *minimum value* denoted by $J_{\min}$. At this point, the *gradient vector* ∇J is identically zero. In other words,

$$\nabla_k J = 0, \qquad k = 0, 1, \ldots, M-1 \tag{5.44}$$

where $\nabla_k J$ is the kth element of the gradient vector. As before, we write the kth tap weight w_k as

$$w_k = a_k + jb_k$$

Hence, using Eq. (5.43), we may express $\nabla_k J$ as

$$\begin{aligned} \nabla_k J &= \frac{\partial J}{\partial a_k} + j\frac{\partial J}{\partial b_k} \\ &= -2p(-k) + 2\sum_{i=0}^{M-1} w_i r(i-k) \end{aligned} \tag{5.45}$$

Applying the necessary and sufficient condition of Eq. (5.44) for optimality to Eq. (5.45), we find that the optimum tap weights $w_{o0}, w_{o1}, \ldots, w_{o,M-1}$ for the transversal filter in Fig. 5.4 are defined by the system of equations:

$$\sum_{i=0}^{M-1} w_{oi} r(i-k) = p(-k), \qquad k = 0, 1, \ldots, M-1$$

This system of equations is identical to the Wiener–Hopf equations (5.28) derived in Section 5.4.

Minimum Mean-squared Error

Let $\hat{d}(n|\mathcal{U}_n)$ denote the estimate of the desired response $d(n)$, produced at the output of the transversal filter in Fig. 5.4 that is optimized in the mean-squared-error sense, given the tap inputs $u(n), u(n-1), \ldots, u(n-M+1)$ that span the space $\mathcal{U}_n$. Then from Fig. 5.4 we deduce that

$$\begin{aligned} \hat{d}(n|\mathcal{U}_n) &= \sum_{k=0}^{M-1} w_{ok}^* u(n-k) \\ &= \mathbf{w}_o^H \mathbf{u}(n) \end{aligned} \tag{5.46}$$

where $\mathbf{w}_o$ is the tap-weight vector of the optimum filter with elements $w_{o0}, w_{o1}, \ldots, w_{o,M-1}$, and $\mathbf{u}(n)$ is the tap-input vector defined in Eq. (5.30). Note that $\mathbf{w}_o^H \mathbf{u}(n)$ denotes an inner product of the optimum tap-weight vector $\mathbf{w}_o$ and the tap-input vector $\mathbf{u}(n)$. We assume that $\mathbf{u}(n)$ has zero mean, making the estimate $\hat{d}(n|\mathcal{U}_n)$ have zero mean too. Hence, we may use Eq. (5.46) to evaluate the variance of $\hat{d}(n|\mathcal{U}_n)$, obtaining

$$\begin{aligned} \sigma_{\hat{d}}^2 &= E[\mathbf{w}_o^H \mathbf{u}(n)\mathbf{u}^H(n)\mathbf{w}_o] \\ &= \mathbf{w}_o^H E[\mathbf{u}(n)\mathbf{u}^H(n)]\mathbf{w}_o \\ &= \mathbf{w}_o^H \mathbf{R} \mathbf{w}_o \end{aligned} \tag{5.47}$$

where $\mathbf{R}$ is the correlation matrix of the tap-weight vector $\mathbf{u}(n)$, as defined in Eq. (5.29). We may eliminate the dependence of the variance $\sigma_{\hat{d}}^2$ on the optimum tap-weight vector $\mathbf{w}_o$ by using Eq. (5.34). In particular, we may rewrite Eq. (5.47) as

$$\begin{aligned} \sigma_{\hat{d}}^2 &= \mathbf{w}_o^H \mathbf{p} \\ &= \mathbf{p}^H \mathbf{w}_o \end{aligned} \tag{5.48}$$

To evaluate the minimum mean-squared error produced by the transversal filter in Fig. 5.4, we may use Eq. (5.48) in (5.19), obtaining

$$\begin{aligned} J_{\min} &= \sigma_d^2 - \mathbf{p}^H \mathbf{w}_o \\ &= \sigma_d^2 - \mathbf{p}^H \mathbf{R}^{-1} \mathbf{p} \end{aligned} \tag{5.49}$$

which is the desired result.

Canonical Form of the Error-Performance Surface

Equation (5.43) defines the expanded form of the mean-squared error J produced by the transversal filter in Fig. 5.4. We may rewrite this equation in matrix form, by using the definitions for the correlation matrix $\mathbf{R}$ and the cross-correlation vector $\mathbf{p}$ given in Eqs. (5.29) and (5.33), respectively, as shown by

$$J(\mathbf{w}) = \sigma_d^2 - \mathbf{w}^H\mathbf{p} - \mathbf{p}^H\mathbf{w} + \mathbf{w}^H\mathbf{R}\mathbf{w} \tag{5.50}$$

where the mean-squared error is written as $J(\mathbf{w})$ to emphasize its dependence on the tap-weight vector $\mathbf{w}$. As mentioned in Chapter 2 the correlation matrix $\mathbf{R}$ is almost always positive definite, so that the inverse matrix $\mathbf{R}^{-1}$ exists. Accordingly, expressing $J(\mathbf{w})$ as a "perfect square" in $\mathbf{w}$, we may rewrite Eq. (5.50) in the form

$$J(\mathbf{w}) = \sigma_d^2 - \mathbf{p}^H\mathbf{R}^{-1}\mathbf{p} + (\mathbf{w} - \mathbf{R}^{-1}\mathbf{p})^H\mathbf{R}(\mathbf{w} - \mathbf{R}^{-1}\mathbf{p}) \tag{5.51}$$

From Eq. (5.51), we now immediately see that

$$\min_{\mathbf{w}} J(\mathbf{w}) = \sigma_d^2 - \mathbf{p}^H\mathbf{R}^{-1}\mathbf{p}$$

for

$$\mathbf{w}_o = \mathbf{R}^{-1}\mathbf{p}$$

In effect, starting from Eq. (5.50), we have rederived the Wiener filter in a rather simple way. Moreover, we may use the defining equations for the Wiener filter to explicitly show the unique optimality of the minimizing tap-weight vector $\mathbf{w}_o$ by writing

$$J(\mathbf{w}) = J_{\min} + (\mathbf{w} - \mathbf{w}_o)^H\mathbf{R}(\mathbf{w} - \mathbf{w}_o) \tag{5.52}$$

This equation shows explicitly the unique optimality of the minimizing tap-weight vector $\mathbf{w}_o$.

Although the quadratic form on the right-hand side of Eq. (5.52) is quite informative, nevertheless, it is desirable to change the basis on which it is defined so that the representation of the error-performance surface is considerably simplified. To do this, we recall from Chapter 4 that the correlation matrix $\mathbf{R}$ of the tap-input vector may be expressed in terms of eigenvalues and eigenvectors as follows:

$$\mathbf{R} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^H \tag{5.53}$$

where $\mathbf{\Lambda}$ is a diagonal matrix consisting of the eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_M$ of the correlation matrix, and the matrix $\mathbf{Q}$ has for its columns the eigenvectors $\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_M$ associated with these eigenvalues, respectively. Hence, substituting Eq. (5.53) into (5.52), we get

$$J = J_{\min} + (\mathbf{w} - \mathbf{w}_o)^H\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^H(\mathbf{w} - \mathbf{w}_o) \tag{5.54}$$

Define a *transformed* version of the difference between the tap-weight vector $\mathbf{w}$ and the optimum solution $\mathbf{w}_o$ as

$$\boldsymbol{\nu} = \mathbf{Q}^H(\mathbf{w} - \mathbf{w}_o) \tag{5.55}$$

Then we may put the quadratic form of Eq.(5.54) into its *canonical form* defined by

$$J = J_{\min} + \boldsymbol{\nu}^H \boldsymbol{\Lambda} \boldsymbol{\nu} \tag{5.56}$$

This new formulation of the mean-squared error contains no cross-product terms, as shown by

$$\begin{aligned} J &= J_{\min} + \sum_{k=1}^{M} \lambda_k \nu_k \nu_k^* \\ &= J_{\min} + \sum_{k=1}^{M} \lambda_k |\nu_k|^2 \end{aligned} \tag{5.57}$$

where ν_k is the kth component of the vector $\boldsymbol{\nu}$. The feature that makes the canonical form of Eq. (5.57) a rather useful representation of the error-performance surface is the fact that the components of the transformed coefficient vector $\boldsymbol{\nu}$ constitute the *principal axes* of the error-performance surface. The practical significance of this result will become apparent in later chapters.

5.6 NUMERICAL EXAMPLE

To illustrate the filtering theory developed above, we consider the example depicted in Fig. 5.5. The desired response $d(n)$ is modeled as an AR process of order 1; that is, it may be produced by applying a white-noise process $v_1(n)$ of zero mean and variance $\sigma_1^2 = 0.27$ to the input of an all-pole filter of order 1, whose transfer function equals [see Fig. 5.5(a)]

$$H_1(z) = \frac{1}{1 + 0.8458z^{-1}}$$

The process $d(n)$ is applied to a communication channel modeled by the all-pole transfer function

$$H_2(z) = \frac{1}{1 - 0.9458z^{-1}}$$

The channel output $x(n)$ is corrupted by an additive white-noise process $v_2(n)$ of zero mean and variance $\sigma_2^2 = 0.1$, so a sample of the received signal $u(n)$ equals [see Fig. 5.5(b)]

$$u(n) = x(n) + v_2(n)$$

The white-noise processes $v_1(n)$ and $v_2(n)$ are uncorrelated. It is also assumed that $d(n)$ and $u(n)$, and therefore $v_1(n)$ and $v_2(n)$, are all real valued.

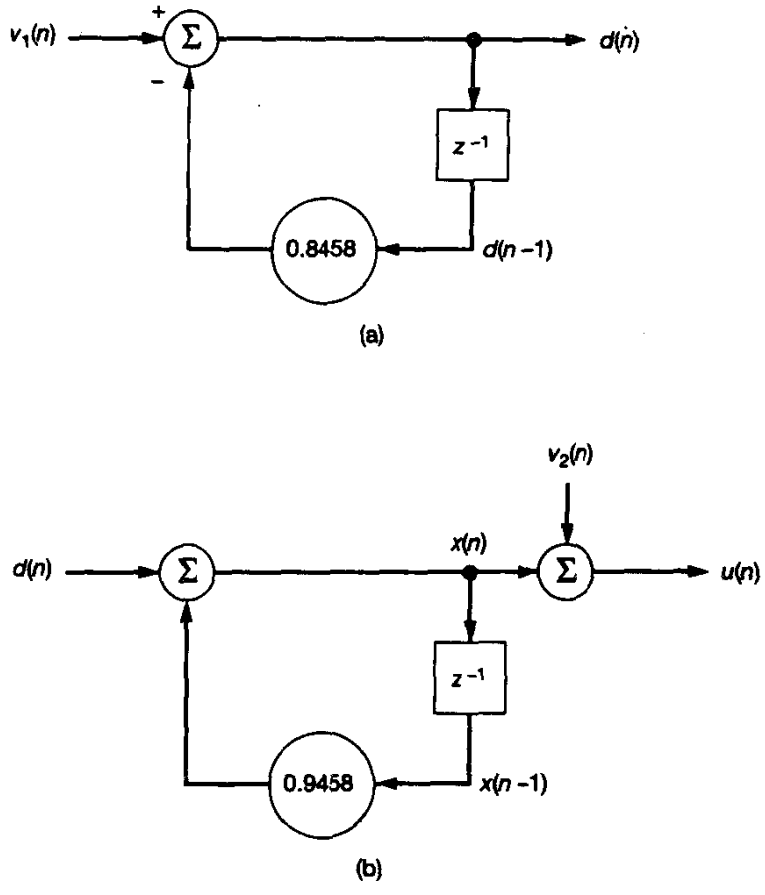


Figure 5.5 (a) Autoregressive model of desired response $d(n)$; (b) model of noisy communication channel.

The requirement is to specify a Wiener filter consisting of a transversal filter with two taps, which operates on the received signal $u(n)$ so as to produce an estimate of the desired response that is optimum in the mean-square sense.

Statistical Characterization of the Desired Response $d(n)$ and the Received Signal $u(n)$

We begin the analysis by considering the difference equations that characterize the various processes described by the models of Fig. 5.5. First, the generation of the desired response $d(n)$ is governed by the first-order difference equation

$$d(n) + a_1 d(n-1) = v_1(n) \tag{5.58}$$

where $a = 0.8458$. The variance of the process $d(n)$ equals (see Problem 4 of Chapter 2)

$$\begin{aligned}\sigma_d^2 &= \frac{\sigma_1^2}{1 - a_1^2} \\ &= \frac{0.27}{1 - (0.8458)^2} \\ &= 0.9486\end{aligned} \tag{5.59}$$

The process $d(n)$ acts as input to the channel. Hence, from Fig. 5.5(b), we find that the channel output $x(n)$ is related to the channel input $d(n)$ by the first-order difference equation

$$x(n) + b_1 x(n - 1) = d(n) \tag{5.60}$$

where $b_1 = -0.9458$. We also observe from the two parts of Fig. 5.5 that the channel output $x(n)$ may be generated by applying the white-noise process $v_1(n)$ to a second-order all-pole filter whose transfer function equals

$$\begin{aligned}H(z) &= H_1(z)H_2(z) \\ &= \frac{1}{(1 + 0.8458z^{-1})(1 - 0.9458z^{-1})}\end{aligned} \tag{5.61}$$

Accordingly, $x(n)$ is a second-order AR process described by the difference equation

$$x(n) + a_1 x(n - 1) + a_2 x(n - 2) = v(n) \tag{5.62}$$

where $a_1 = -0.1$ and $a_2 = -0.8$. Note that both AR processes $d(n)$ and $x(n)$ are wide-sense stationary.

To characterize the Wiener filter, we need to solve the Wiener–Hopf equations (5.34). This set of equations requires knowledge of two quantities: (1) the correlation matrix $\mathbf{R}$ pertaining to the received signal $u(n)$, and (2) the cross-correlation vector $\mathbf{p}$ between $u(n)$ and the desired response $d(n)$. In our example, $\mathbf{R}$ is a 2-by-2 matrix and $\mathbf{p}$ is a 2-by-1 vector, since the transversal filter used to implement the Wiener filter is assumed to have two taps.

The received signal $u(n)$ consists of the channel output $x(n)$ plus the additive white noise $v_2(n)$. Since the process $x(n)$ and $v_2(n)$ are uncorrelated, it follows that the correlation matrix $\mathbf{R}$ equals the correlation matrix of $x(n)$ plus the correlation matrix of $v_2(n)$. That is,

$$\mathbf{R} = \mathbf{R}_x + \mathbf{R}_2 \tag{5.63}$$

For the correlation matrix $\mathbf{R}_x$, we write [since the process $x(n)$ is real valued]

$$\mathbf{R}_x = \begin{bmatrix} r_x(0) & r_x(1) \\ r_x(1) & r_x(0) \end{bmatrix}$$

where $r_x(0)$ and $r_x(1)$ are the autocorrelation functions of the received signal $x(n)$ for lags of 0 and 1, respectively. From Section 2.9 we have

$$\begin{aligned} r_x(0) &= \sigma_x^2 \\ &= \left(\frac{1+a_2}{1-a_2}\right)\frac{\sigma_1^2}{[(1+a_2)^2 - a_1^2]} \\ &= \left(\frac{1-0.8}{1+0.8}\right)\frac{0.27}{[(1-0.8)^2 - (0.1)^2]} \\ &= 1 \end{aligned}$$

$$\begin{aligned} r_x(1) &= \frac{-a_1}{1+a_2} \\ &= \frac{0.1}{1-0.8} \\ &= 0.5 \end{aligned}$$

Hence,

$$\mathbf{R}_x = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix} \tag{5.64}$$

Next we observe that since $v_2(n)$ is a white-noise process of zero mean and variance $\sigma_2^2 = 0.1$, the 2-by-2 correlation matrix $\mathbf{R}_2$ of this process equals

$$\mathbf{R}_2 = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix} \tag{5.65}$$

Thus, substituting Eqs. (5.64) and (5.65) in Eq. (5.63), we find that the 2-by-2 correlation matrix of the received signal $x(n)$ equals

$$\mathbf{R} = \begin{bmatrix} 1.1 & 0.5 \\ 0.5 & 1.1 \end{bmatrix} \tag{5.66}$$

For the 2-by-1 cross-correlation vector $\mathbf{p}$, we write

$$\mathbf{p} = \begin{bmatrix} p(0) \\ p(-1) \end{bmatrix}$$

where $p(0)$ and $p(-1)$ are the cross-correlation functions between $d(n)$ and $u(n)$ for lags of 0 and -1, respectively. Since these two processes are real valued, we have

$$p(k) = p(-k) = E[u(n-k)d(n)], \qquad k = 0, 1 \tag{5.67}$$

Substituting Eqs. (5.57) and (5.60) into Eq. (5.67), and recognizing that the channel output $x(n)$ is uncorrelated with the white-noise process $v_2(n)$, we get

$$p(k) = r_x(k) + b_1 r_x(k-1), \qquad k = 0, 1$$

Putting $b_1 = -0.9458$ and using the element values for the correlation matrix $\mathbf{R}_x$ given in Eq. (5.64), we obtain

$$\begin{aligned} p(0) &= r_x(0) + b_1 r_x(-1) \\ &= 1 - 0.9458 \times 0.5 \\ &= 0.5272 \\ p(1) &= r_x(1) + b_1 r_x(0) \\ &= 0.5 - 0.9458 \times 1 \\ &= -0.4458 \end{aligned}$$

Hence,

$$\mathbf{p} = \begin{bmatrix} 0.5272 \\ -0.4458 \end{bmatrix} \tag{5.68}$$

Error-Performance Surface

The dependence of the mean-squared error on the 2-by-1 tap-weight vector $\mathbf{w}$ is defined by Eq. (5.50). Hence, substituting Eqs. (5.59), (5.66), and (5.68) into (5.50), we get

$$\begin{aligned} J(w_0, w_1) &= 0.9486 - 2[0.5272, -0.4458]\begin{bmatrix} w_0 \\ w_1 \end{bmatrix} + [w_0, w_1]\begin{bmatrix} 1.1 & 0.5 \\ 0.5 & 1.1 \end{bmatrix}\begin{bmatrix} w_0 \\ w_1 \end{bmatrix} \\ &= 0.9486 - 1.0544w_0 + 0.8916w_1 + w_0 w_1 + 1.1(w_0^2 + w_1^2) \end{aligned}$$

Using a three-dimensional computer plot, the mean-squared error $J(w_0, w_1)$ is plotted versus the tap weights w_0 and w_1. The result is shown in Fig. 5.6.

Figure 5.7 shows contour plots of the tap weight w_1 versus w_0 for varying values of the mean-squared error J. We see that the locus of w_1 versus w_0 for a fixed J is in the form of an ellipse. The elliptical locus shrinks in size as the mean-squared error J approaches the minimum value $J_{\min}$. For $J = J_{\min}$, the locus reduces to a point with coordinates w_{o0} and w_{o1}.

Wiener Filter

The 2-by-1 optimum tap-weight vector $\mathbf{w}_o$ of the Wiener filter is defined by Eq. (5.36). In particular, it consists of the inverse matrix $\mathbf{R}^{-1}$ multiplied by the cross-correlation vector $\mathbf{p}$. Inverting the correlation matrix $\mathbf{R}$ of Eq. (5.66), we get

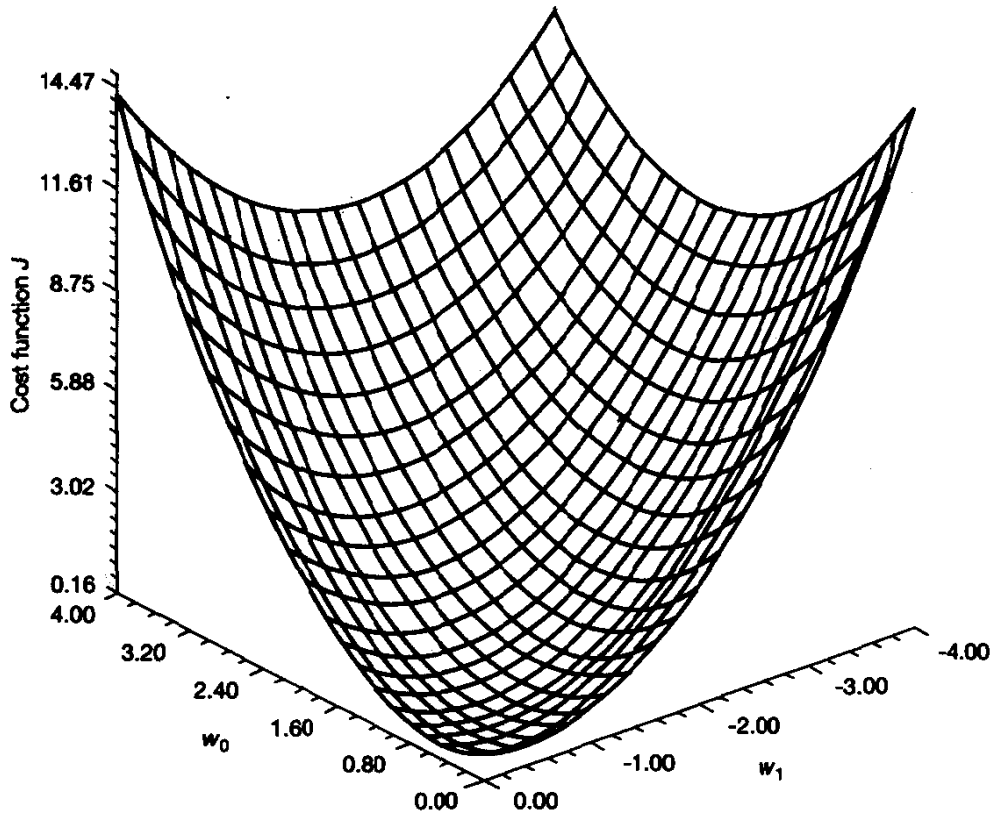


Figure 5.6 Error-performance surface of the two-tap transversal filter described in the numerical example.

$$\begin{aligned}\mathbf{R}^{-1} &= \begin{bmatrix} r(0) & r(1) \\ r(1) & r(0) \end{bmatrix}^{-1} \\ &= \frac{1}{r^2(0) - r^2(1)} \begin{bmatrix} r(0) & -r(1) \\ -r(1) & r(0) \end{bmatrix} \\ &= \begin{bmatrix} 1.1456 & -0.5208 \\ -0.5208 & 1.1456 \end{bmatrix}\end{aligned} \tag{5.69}$$

Hence, substituting Eqs. (5.68) and (5.69) into Eq. (5.36), we get the desired result:

$$\begin{aligned}\mathbf{w}_o &= \begin{bmatrix} 1.1456 & -0.5208 \\ -0.5208 & 1.1456 \end{bmatrix} \begin{bmatrix} 0.5272 \\ -0.4458 \end{bmatrix} \\ &= \begin{bmatrix} 0.8360 \\ -0.7853 \end{bmatrix}\end{aligned} \tag{5.70}$$

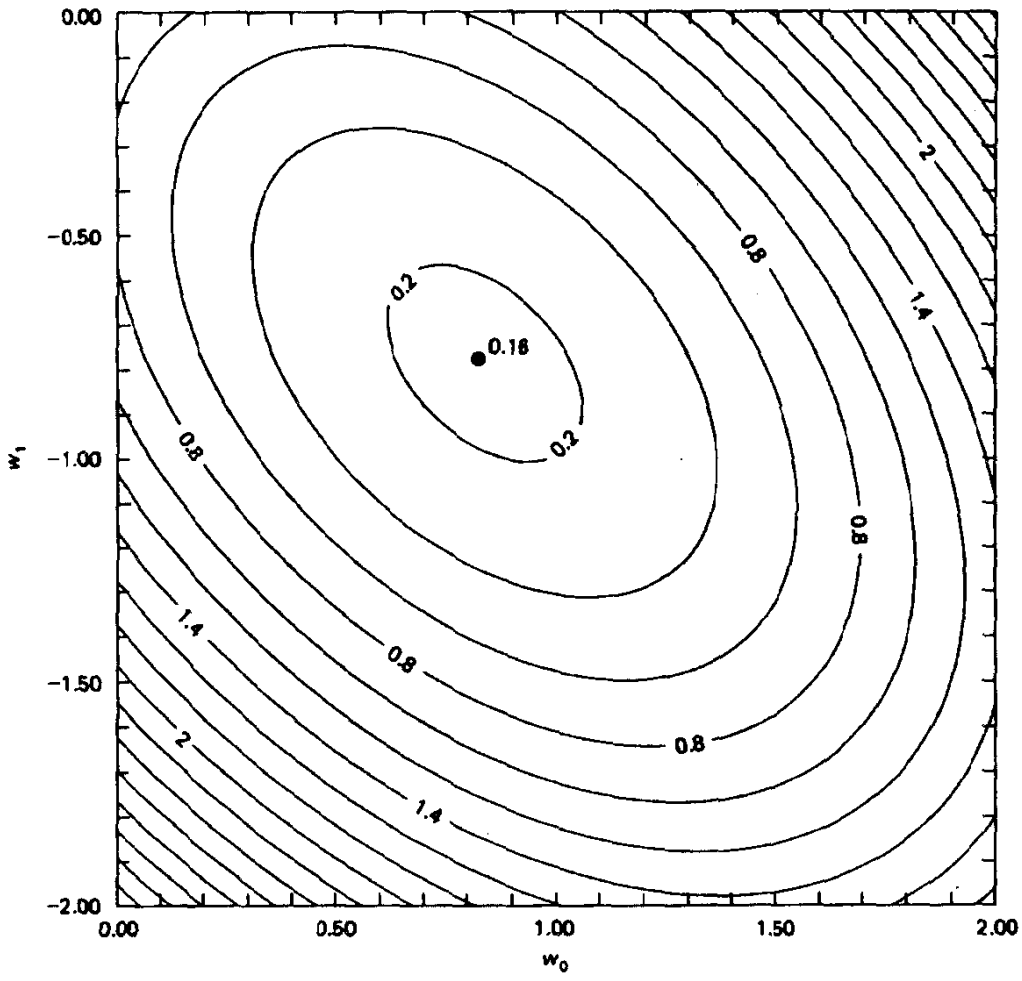


Figure 5.7 Contour plots of the error-performance surface depicted in Fig. 5.6.

Minimum Mean-Squared Error

To evaluate the minimum value of the mean-squared error, $J_{\min}$, which results from the use of the optimum tap-weight vector $\mathbf{w}_o$, we use Eq. (5.49). Hence, substituting Eqs. (5.59), (5.68), and (5.70) into Eq. (5.49), we get

$$\begin{aligned} J_{\min} &= 0.9486 - [0.5272, -0.4458]\begin{bmatrix} 0.8360 \\ -0.7853 \end{bmatrix} \\ &= 0.1579 \end{aligned} \tag{5.71}$$

The point represented jointly by the optimum tap-weight vector $\mathbf{w}_o$ of Eq. (5.70) and the minimum mean-squared error of Eq. (5.71) defines the bottom of the error-performance surface in Fig. 5.6, or the center of the contour plots in Fig. 5.7.

Canonical Error-Performance Surface

The characteristic equation of the 2-by-2 correlation matrix $\mathbf{R}$ of Eq. (5.66) is

$$(1.1 - \lambda)^2 - (0.5)^2 = 0$$

The two eigenvalues of the correlation matrix $\mathbf{R}$ are therefore

$$\lambda_1 = 1.6$$

$$\lambda_2 = 0.6$$

The canonical error-performance surface is therefore defined by [see Eq. (5.57)]

$$J(v_1, v_2) = J_{\min} + 1.6v_1^2 + 0.6v_2^2 \tag{5.72}$$

The locus of v_2 versus v_1, as defined in Eq. (5.72), traces an *ellipse* for a fixed value of $J - J_{\min}$. In particular, the ellipse has a minor axis of $[(J - J_{\min})/\lambda_1]^{1/2}$ along the v_1-coordinate and a major axis of $[(J - J_{\min}/\lambda_2)]^{1/2}$ along the v_2-coordinate; this assumes that $\lambda_1 > \lambda_2$, which is how they are related.

5.7 CHANNEL EQUALIZATION

We turn next to some applications of Wiener filter theory. In this section we consider a temporal signal-processing problem, namely, that of channel equalization. This is followed by a spatial signal processing problem, namely, that of beamforming, which is presented in the next two sections.

A communication channel well suited for the transmission of digital data (e.g., computer data) is the *telephone channel*, which is characterized by a high signal-to-noise ratio. However, a practical shortcoming of the telephone channel is the fact that it is *bandwidth-limited*. Consequently, when data are transmitted over the channel by means of discrete pulse-amplitude modulation combined with a linear modulation scheme (e.g., quadriphase-shift keying), the number of detectable levels that the telephone channel can support is essentially limited by *intersymbol interference* rather than by additive noise. In what follows, we are therefore justified in ignoring channel noise. Intersymbol interference (ISI) arises because of the "spreading" of a transmitted pulse due to the dispersive nature of the channel, which results in an overlap of adjacent pulses. If ISI is left unchecked, it can produce errors in the reconstructed data stream at the receiver output. An effective method for combatting the system degradation due to ISI is to connect an *equalizer* in cascade with the channel, as in Fig. 5.8(a). A structure well suited for this application is the tapped-delay-line filter shown in Fig. 5.8(b). For equalizer *symmetry*, the total number of taps in the equalizer is chosen to be $(2N + 1)$, with the tap-weights themselves denoted by $h_{-N}, \ldots, h_{-1}, h_0, h_1, \ldots, h_N$. The impulse response of the equalizer is, therefore,

$$h(n) = \sum_{k=-N}^{N} h_k \, \delta(n - k) \tag{5.73}$$

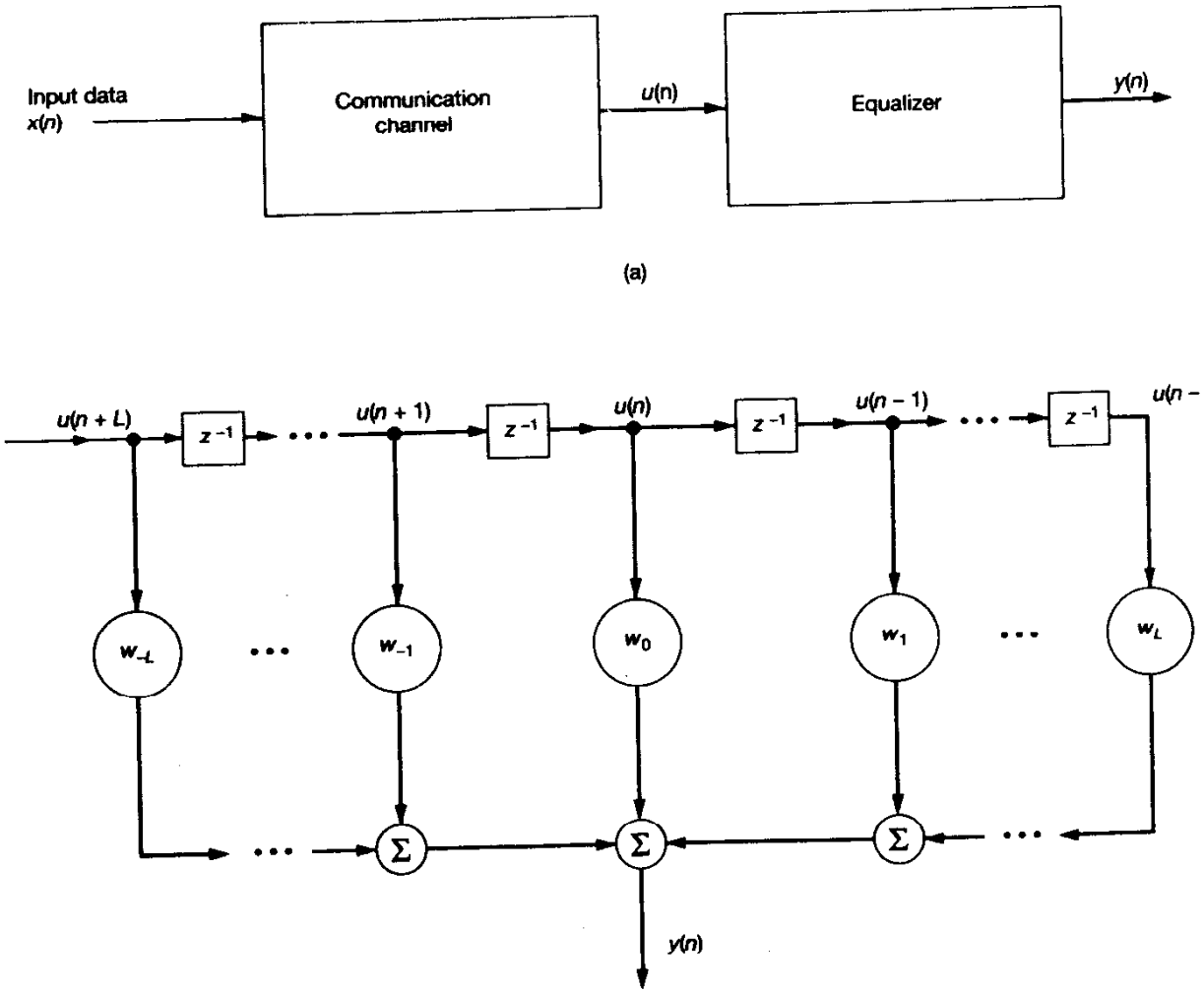


Figure 5.8 (a) Block diagram of equalized channel. (b) Symmetric tapped-delay-line filter implementation of the equalizer.

where $\delta(n)$ is the Dirac delta function. Similarly, we may express the impulse response of the channel as

$$c(n) = \sum_k c_k \delta(n - k) \tag{5.74}$$

In light of what we said earlier, we may ignore the effect of channel noise. Thus, the cascade connection of the channel and the equalizer is equivalent to a single tapped-delay-line filter. Let the impulse response of this equivalent filter be defined by

$$w(n) = \sum_{k=-N}^{N} w_k \delta(n - k) \tag{5.75}$$

where the sequence w_n is equal to the *convolution* of the sequences c_n and h_n. That is, we have

$$w_l = \sum_{k=-N}^{N} h_k c_{l-k}, \qquad l = 0, \pm 1, \ldots, \pm N \tag{5.76}$$

Let the data sequence $u(n)$ applied to the channel input consist of a white-noise sequence of zero mean and unit variance. In practice, such a sequence may be closely approximated by a pseudo-inverse sequence generated using a feedback shift register. Accordingly, we may express the elements of the correlation matrix $\mathbf{R}$ of the channel input as follows:

$$r(l) = \begin{cases} 1, & l = 0 \\ 0, & l \neq 0 \end{cases} \tag{5.77}$$

For the desired response $d(n)$ supplied to the equalizer, we assume the availability of a delayed "replica" of the transmitted sequence. This desired response may be generated by using another feedback shifter of identical design to that used to supply the original data sequence $u(n)$. The two feedback shift registers are synchronized with each other, such that we may set

$$d(n) = u(n)$$

where time n is measured with respect to the center tap of the equalizer. Thus, the cross-correlation between the transmitted sequence $u(n)$ and the desired response $d(n)$ is defined by

$$p(l) = \begin{cases} 1, & l = 0 \\ 0, & l = \pm 1, \pm 2, \ldots, \pm N \end{cases} \tag{5,78}$$

The stage is now set for the application of the Wiener–Hopf equations (5.28). Specifically, in light of Eqs. (5.77) and (5.78), we may write

$$w_l = \begin{cases} 1, & l = 0 \\ 0, & l = \pm 1, \pm 2, \ldots, \pm N \end{cases} \tag{5.79}$$

Equivalently, invoking the convolution sum of Eq. (5.76), we have

$$\sum_{k=-N}^{N} h_k c_{l-k} = \begin{cases} 1, & l = 0 \\ 0, & l = \pm 1, \pm 2, \ldots, \pm N \end{cases} \tag{5.80}$$

This system of simultaneous equations may be rewritten in the expanded matrix form:

$$\begin{bmatrix} c_0 & \cdots & c_{-N+1} & c_{-N} & c_{-N-1} & \cdots & c_{-2N} \\ \vdots & & \vdots & \vdots & \vdots & & \vdots \\ c_{N-1} & \cdots & c_0 & c_{-1} & c_{-2} & \cdots & c_{-N-1} \\ c_N & \cdots & c_1 & c_0 & c_{-1} & \cdots & c_{-N} \\ c_{N+1} & \cdots & c_2 & c_1 & c_0 & \cdots & c_{-N+1} \\ \vdots & & \vdots & \vdots & \vdots & & \vdots \\ c_{2N} & \cdots & c_{N+1} & c_N & c_{N-1} & \cdots & c_0 \end{bmatrix} \begin{bmatrix} h_{-N} \\ \vdots \\ h_{-1} \\ h_0 \\ h_1 \\ \vdots \\ h_N \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{5.81}$$

Thus, given the impulse response of the channel characterized by the coefficients $c_{-N}, \ldots, c_{-1}, c_0, c_1, \ldots, c_N$, we may use Eq. (5.81) to solve for the unknown tap-weights $h_{-N}, \ldots, h_{-1}, h_0, h_1, \ldots, h_N$ of the equalizer.

In the literature on digital communications, an equalizer designed in accordance with Eq. (5.81) is referred to as a *zero-forcing equalizer* (Lucky et al., 1968). The equalizer is so called because, with a single pulse transmitted over the channel, it "forces" the receiver output to be zero at all the sampling instances, except for the time instant that corresponds to the transmitted pulse.

5.8 LINEARLY CONSTRAINED MINIMUM VARIANCE FILTER

The essence of a Wiener filter is that it minimizes the mean-square value of an estimation error, defined as the difference between a desired response and the actual filter output. In solving this optimization (minimization) problem, there are *no* constraints imposed on the solution. In some filtering applications, however, it may be desirable (or even mandatory) to design a filter that minimizes a mean-square criterion, subject to a specific *constraint*. For example, the requirement may be that of minimizing the average output power of a linear filter while the response of the filter measured at some specific frequency of interest is constrained to remain constant. In this section, we consider one such solution.

Consider a linear transversal filter, as in Fig. 5.9. The filter output, in response to the tap inputs $u(n), u(n-1), \ldots, u(n-M+1)$, is given by

$$y(n) = \sum_{k=0}^{M-1} w_k^* u(n-k) \tag{5.82}$$

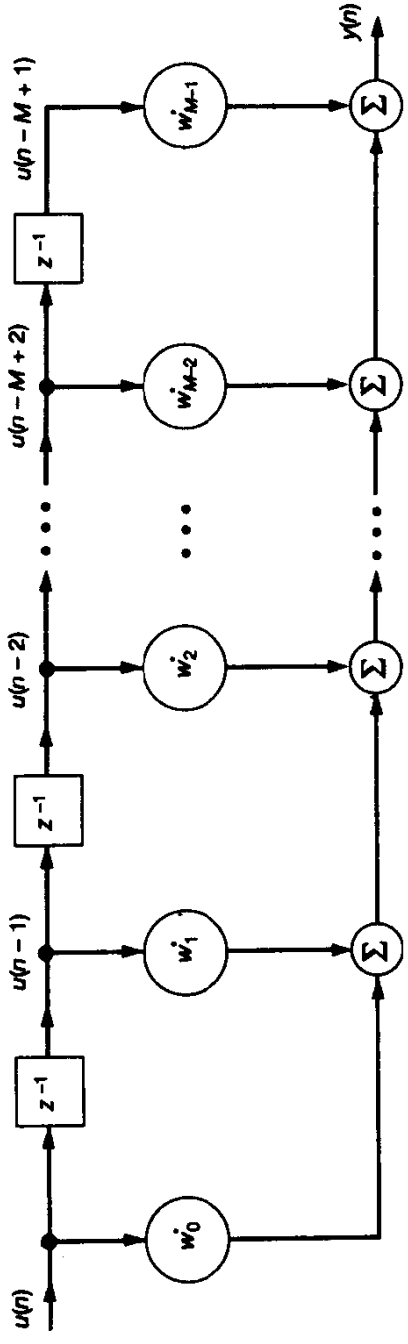


Figure 5.9 Linear transversal filter.

For the special case of a *sinusoidal excitation*

$$u(n) = e^{j\omega n} \tag{5.83}$$

we may rewrite Eq. (5.82) as

$$y(n) = e^{j\omega n} \sum_{k=0}^{M-1} w_k^* e^{-j\omega k} \tag{5.84}$$

where ω is the angular frequency of the excitation, which is normalized with respect to the sampling rate.

The *constrained optimization problem* we wish to solve may now be stated as follows:

Find the optimum set of filter coefficients $w_{o0}, w_{o1}, \ldots, w_{o,M-1}$ that minimizes the mean-square value of the filter output $y(n)$, subject to the linear constraint

$$\sum_{k=0}^{M-1} w_k^* e^{-j\omega_0 k} = g \tag{5.85}$$

where ω_0 is a prescribed value of the normalized angular frequency ω, lying inside the range $-\pi < \omega \leq \pi$, and g is a complex-valued gain.

The constrained optimization filtering problem described by Eqs. (5.82) and (5.85) is *temporal* in nature. We may formulate the *spatial* version of this constrained optimization problem by considering the *beamformer* depicted in Fig. 5.10, which consists of a linear array of uniformly spaced antenna elements. The array is illuminated by an isotropic source located in the *far field*, such that, at time n, a *plane wave* impinges on the array along a direction specified by the angle θ_0 with respect to the perpendicular to the array. It is also assumed that the interelement spacing of the array is less than $\lambda/2$, where λ is the wavelength of the transmitted signal so as to avoid the appearance of grating lobes (Skolnik, 1980). The resulting beamformer output is given by

$$y(n) = u_0(n) \sum_{k=0}^{M-1} w_k^* e^{-jk\phi_0} \tag{5.86}$$

where the *direction of arrival* is defined by the electrical angle ϕ_0 that is related to the angle of incidence θ_0 by Eq. (45) of the introductory chapter, $u_0(n)$ is the electrical signal picked up by the antenna element labeled 0 in Fig. 5.10 that is treated as the point of reference, and the w_k denote the *elemental weights* of the beamformer. The spatial version of the constrained optimization problem may thus be stated as follows:

Find the optimum set of elemental weights $w_{o0}, w_{o1}, \ldots, w_{o,M-1}$ that minimizes the mean-square value of the beamformer output, subject to the linear constraint:

$$\sum_{k=0}^{M-1} w_k^* e^{-jk\phi_0} = g \tag{5.87}$$

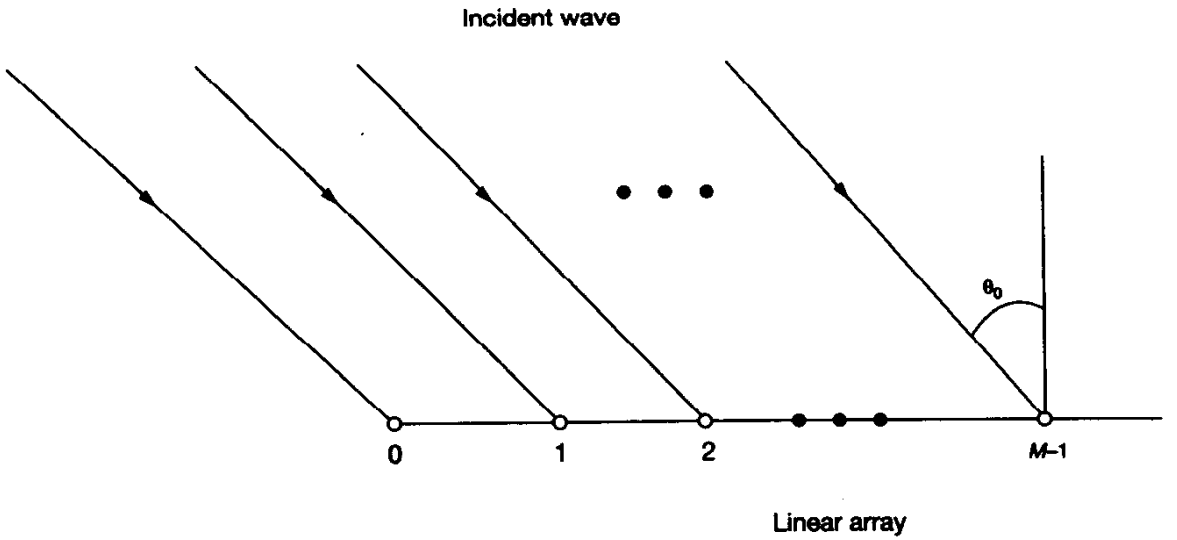


Figure 5.10 Illustrating a plane wave incident on a linear array antenna.

where ϕ_0 is a prescribed value of the electrical angle ϕ, lying inside the range $-\pi < \phi \leq \pi$, and g is a complex-valued gain. The beamformer is narrowband in the sense that its response needs to be constrained only at a single frequency.

Comparing the transversal filter and beamformer described in Figs. 5.9 and 5.10, respectively, we see that although they address entirely different physical situations, their formulations are equivalent in mathematical terms. Indeed, in both cases we have exactly the same constrained optimization problem on our hands.

To solve this constrained optimization problem, we use the *method of Lagrange multipliers*.[5] We begin by defining a *real-valued* cost function J that combines the two parts of the constrained optimization problem. Specifically, we write

$$J = \underbrace{\sum_{k=0}^{M-1} \sum_{i=0}^{M-1} w_k^* w_i r(i-k)}_{\text{output power}} + \underbrace{\text{Re}\left[\lambda^*\left(\sum_{k=0}^{M-1} w_k^* e^{-j\phi_0 k} - g\right)\right]}_{\text{linear constraint}} \tag{5.88}$$

where λ is a *complex Lagrange multiplier*. Note that there is no desired response in the definition of the cost function J; rather, it includes a linear constraint that has to be satisfied for the prescribed electrical angle ϕ_0 in the context of beamforming, or equivalently the angular frequency ω_0 in transversal filtering. In any event, imposition of the linear constraint preserves the signal of interest, and minimization of the cost function J attenuates interference or noise that can be troublesome if left unchecked.

We wish to solve for the optimum values of the elemental weights of the beamformer that minimize J defined in Eq. (5.88). To do so, we may determine the gradient vector ∇J, and then set it equal to zero. Thus, proceeding in a manner similar to that described in Section 5.2, we find that the kth element of the gradient vector ∇J is

$$\nabla_k J = 2 \sum_{i=0}^{M-1} w_i r(i-k) + \lambda^* e^{-j\phi_0 k} \tag{5.89}$$

Let w_{oi} be the ith element of the optimum weight vector $\mathbf{w}_o$. Then the condition for optimality of the beamformer is described by

$$\sum_{i=0}^{M-1} w_{oi} r(i-k) = -\frac{\lambda^*}{2} e^{-j\phi_0 k}, \qquad k = 0, 1, \ldots, M-1 \tag{5.90}$$

This system of M simultaneous equations defines the optimum values of the beamformer's elemental weights. It has a form somewhat similar to that of the Wiener–Hopf equations (5.28).

At this point in the analysis, we find it convenient to switch to matrix notation. In particular, we may rewrite the system of M simultaneous equations given in (5.90) simply as

$$\mathbf{R}\mathbf{w}_o = -\frac{\lambda^*}{2}\mathbf{s}(\phi_0) \tag{5.91}$$

[5] The method of Lagrange multipliers is described in Appendix C.

where $\mathbf{R}$ is the M-by-M correlation matrix, and $\mathbf{w}_o$ is the M-by-1 optimum weight vector of the constrained beamformer. The M-by-1 *steering* vector $\mathbf{s}(\phi_0)$ is defined by

$$\mathbf{s}(\phi_0) = [1, e^{-j\phi_0}, \ldots, e^{-j(M-1)\phi_0}]^T \tag{5.92}$$

Solving Eq. (5.91) for $\mathbf{w}_o$, we thus have

$$\mathbf{w}_o = -\frac{\lambda^*}{2}\mathbf{R}^{-1}\mathbf{s}(\phi_0) \tag{5.93}$$

where $\mathbf{R}^{-1}$ is the inverse of the correlation matrix $\mathbf{R}$, assuming that $\mathbf{R}$ is nonsingular. This assumption is perfectly justified in practice by virtue of the fact that, in the context of a beamformer, the received signal at the output of each antenna element of the beamformer includes a white (thermal) noise component.

The solution for the optimum weight vector $\mathbf{w}_o$ given in Eq. (5.93) is not quite complete, as it involves the unknown Lagrange multiplier λ (or its complex conjugate to be precise). To eliminate λ^* from this expression, we first use the linear constraint of Eq. (5.87) to write

$$\mathbf{w}_o^H \mathbf{s}(\phi_0) = g \tag{5.94}$$

Hence, taking the Hermitian transpose of both sides of Eq. (5.93), postmultiplying by $\mathbf{s}(\phi_0)$, and then using the linear constraint of Eq. (5.94), we get

$$\lambda = -\frac{2g}{\mathbf{s}^H(\phi_0)\mathbf{R}^{-1}\mathbf{s}(\phi_0)} \tag{5.95}$$

where we have used the fact that $\mathbf{R}^{-H} = \mathbf{R}^{-1}$. The quadratic form $\mathbf{s}^H(\phi_0)\mathbf{R}^{-1}\mathbf{s}(\phi_0)$ is real valued. Hence, substituting Eq. (5.95) in (5.93), we get the desired formula for the optimum weight vector

$$\mathbf{w}_o = \frac{g^*\mathbf{R}^{-1}\mathbf{s}(\phi_0)}{\mathbf{s}^H(\phi_0)\mathbf{R}^{-1}\mathbf{s}(\phi_0)} \tag{5.96}$$

Note that by minimizing the output power, subject to the linear constraint of Eq. (5.87), signals incident on the array along directions different from the prescribed value ϕ_0 tend to be attenuated.

For obvious reasons, a beamformer characterized by the weight vector $\mathbf{w}_o$ is referred to as a *linearly constrained minimum variance* (LCMV) *beamformer*. For a zero-mean input and therefore zero-mean output, "minimum variance" and "minimum mean-square value" are indeed synonymous. Also, in light of what we said previously, the solution defined by Eq. (5.96) with ω_0 substituted for ϕ_0 may be referred to as an *LCMV filter*.

Minimum Variance Distortionless Response Beamformer

The complex constant g defines the response of an LCMV beamformer at the electrical angle ϕ_0. For the special case of $g = 1$, the optimum solution given in Eq. (5.96) reduces to

$$\mathbf{w}_o = \frac{\mathbf{R}^{-1}\mathbf{s}(\phi_0)}{\mathbf{s}^H(\phi_0)\mathbf{R}^{-1}\mathbf{s}(\phi_0)} \tag{5.97}$$

The response of the beamformer defined by Eq. (5.97) is constrained to equal unity at the electrical angle ϕ_0. In other words, this beamformer is constrained to produce a distortionless response along the look direction corresponding to ϕ_0.

Now, the minimum mean-square value (average power) of the optimum beamformer output may be expressed as the quadratic form

$$J_{\min} = \mathbf{w}_o^H \mathbf{R} \mathbf{w}_o \tag{5.98}$$

Hence, substituting Eq. (5.97) in (5.98) and simplifying, we get the result

$$J_{\min} = \frac{1}{\mathbf{s}^H(\phi_0)\mathbf{R}^{-1}\mathbf{s}(\phi_0)} \tag{5.99}$$

The optimum beamformer is constrained to pass the target signal with unit response, while at the same time minimizing the total output variance. This variance minimization process attenuates interference and noise not originating at the electrical angle ϕ_0. Hence, $J_{\min}$ represents an estimate of the variance of the signal impinging on the array along the direction corresponding to ϕ_0. We may generalize this result and obtain an estimate of variance as a function of direction by formulating $J_{\min}$ as a function of ϕ. In so doing, we obtain the MVDR (spatial) power spectrum defined as

$$S_{\text{MVDR}}(\phi) = \frac{1}{\mathbf{s}^H(\phi)\mathbf{R}^{-1}\mathbf{s}(\phi)} \tag{5.100}$$

where

$$\mathbf{s}(\phi) = [1, e^{-j\phi}, \ldots, e^{-j\phi(M-1)}]^T \tag{5.101}$$

The M-by-1 vector $\mathbf{s}(\phi)$ is called a *spatial scanning vector* in the context of the beamformer of Fig. 5.10, and a *frequency scanning vector* with ω in place of ϕ for the transversal filter of Fig. 5.9. By definition, $S_{\text{MVDR}}(\phi)$ or $S_{\text{MVDR}}(\omega)$ has the dimension of power. Its dependence on the electrical angle ϕ at the beamformer input or the angular frequency ω at the transversal filter input therefore justifies referring to it as a power spectrum estimate. Indeed, it is commonly referred to as the *minimum variance distortionless response* (MVDR) *spectrum*.[6] Note that at any ω in the temporal context, power due to other angular frequencies is minimized. Accordingly, the MVDR spectrum tends to have sharper peaks and higher resolution, compared to nonparametric (classical) methods based on the definition of power spectrum discussed in Chapter 3.

In Appendix F we present a fast algorithm for computing the MVDR spectrum when the correlation matrix $\mathbf{R}$ is known. Also, it is noteworthy that the MVDR beamformer/spectrum analyzer is an important member of the family of *superresolution algo-*

[6] The formula given in Eq. (5.100) is credited to Capon (1969). It is also referred to in the literature as the *maximum-likelihood method (MLM)*. In reality, however, this formula has no bearing on the classical principle of maximum likelihood. The use of the terminology MLM for this formula is therefore not recommended.

rithms. The term "super-resolution" or "high-resolution" refers to the fact that a frequency estimation or angle-of-arrival estimation algorithm so termed has, under carefully controlled conditions, the ability to surpass the limiting behavior of classical Fourier-based methods, with the limitation being imposed by the finite length of the transversal filter or finite aperture of the linear array.

5.9 GENERALIZED SIDELOBE CANCELERS

Continuing with the discussion of the LCMV narrow-band beamformer defined by the linear constraint of Eq. (5.87), we note that this constraint represents the inner product

$$\mathbf{w}^H\mathbf{s}(\phi_0) = g$$

in which $\mathbf{w}$ is the weight vector and $\mathbf{s}(\phi_0)$ is the steering vector pointing along the electrical angle ϕ_0. The steering vector is an M-by-1 vector, where M is the number of antenna elements in the beamformer. We may generalize the notion of a linear constraint by introducing *multiple linear constraints* defined by

$$\mathbf{C}^H\mathbf{w} = \mathbf{g} \tag{5.102}$$

The matrix $\mathbf{C}$ is termed the *constraint matrix*; and the vector $\mathbf{g}$, termed the *gain vector*, has constant elements. Assuming that there are L linear constraints, the matrix $\mathbf{C}$ is an M-by-L matrix, and $\mathbf{g}$ is an L-by-1 vector; each column of the matrix $\mathbf{C}$ represents a single linear constraint. Furthermore, it is assumed that the constraint matrix $\mathbf{C}$ has linearly independent columns. For example, with

$$[s(\phi_0), s(\phi_1)]^H\,\mathbf{w} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

the narrow-band beamformer is constrained to preserve a signal of interest impinging on the array along the electrical angle ϕ_0 and, at the same time, to suppress an interference known to originate along the electrical angle ϕ_1.

Let the columns of an M-by-$(M-L)$ matrix $\mathbf{C}_a$ be defined as a basis for the *orthogonal complement* of the space spanned by the columns of matrix $\mathbf{C}$. Using the definition of an orthogonal complement, we may thus write

$$\mathbf{C}^H\mathbf{C}_a = \mathbf{O} \tag{5.103}$$

or, just as well, write

$$\mathbf{C}_a^H\,\mathbf{C} = \mathbf{O} \tag{5.104}$$

The null matrix $\mathbf{O}$ in Eq. (5.103) is L-by-$(M-L)$, whereas in Eq. (5.104) it is $(M-L)$-by-L; we naturally have $M > L$. Define the M-by$M$ partitioned matrix

$$\mathbf{U} = [\mathbf{C} \vdots \mathbf{C}_a] \tag{5.105}$$

whose columns span the entire M-dimensional signal space. The inverse matrix $\mathbf{U}^{-1}$ exists by virtue of the fact that the determinant of matrix $\mathbf{U}$ is nonzero.

Next, let the M-by-1 weight vector of the beamformer be written in terms of the matrix $\mathbf{U}$ as

$$\mathbf{w} = \mathbf{U}\mathbf{q} \tag{5.106}$$

Equivalently, the M-by-1 vector $\mathbf{q}$ is defined by

$$\mathbf{q} = \mathbf{U}^{-1}\mathbf{w} \tag{5.107}$$

Let the vector $\mathbf{q}$ be partitioned in a compatible way to that in Eq. (5.105), as shown by

$$\mathbf{q} = \begin{bmatrix} \mathbf{v} \\ \cdots\cdots \\ -\mathbf{w}_a \end{bmatrix} \tag{5.108}$$

where $\mathbf{v}$ is an L-by-1 vector, and the $(M-L)$-by-1 vector $\mathbf{w}_a$ is that portion of the weight vector $\mathbf{w}$ that is not affected by the constraints. We may then use the definitions of Eqs. (5.105) and (5.108) in Eq. (5.106) to write

$$\begin{aligned} \mathbf{w} &= [\mathbf{C} \vdots \mathbf{C}_a] \begin{bmatrix} \mathbf{v} \\ \cdots\cdots \\ -\mathbf{w}_a \end{bmatrix} \\ &= \mathbf{C}\mathbf{v} - \mathbf{C}_a\mathbf{w}_a \end{aligned} \tag{5.109}$$

We may now apply the multiple linear constraints of Eq. (5.102), obtaining

$$\mathbf{C}^H\mathbf{C}\mathbf{v} - \mathbf{C}^H\mathbf{C}_a\mathbf{w}_a = \mathbf{g} \tag{5.110}$$

But, from Eq. (5.103) we know that $\mathbf{C}^H\mathbf{C}_a$ is zero; hence, Eq. (5.110) reduces to

$$\mathbf{C}^H\mathbf{C}\mathbf{v} = \mathbf{g} \tag{5.111}$$

Solving for the vector $\mathbf{v}$, we thus get

$$\mathbf{v} = (\mathbf{C}^H\mathbf{C})^{-1}\mathbf{g} \tag{5.112}$$

which shows that the multiple linear constraints do not affect $\mathbf{w}_a$.

Define a nonadaptive beamformer component represented by

$$\mathbf{w}_q = \mathbf{C}\mathbf{v} = \mathbf{C}(\mathbf{C}^H\mathbf{C})^{-1}\mathbf{g} \tag{5.113}$$

which is orthogonal to the columns of matrix $\mathbf{C}_a$ by virtue of the property described in Eq. (5.104); the rationale for using the subscript q in $\mathbf{w}_q$ will become apparent later. Using this definition, we may use Eq. (5.109) to express the overall weight vector of the beamformer as

$$\mathbf{w} = \mathbf{w}_q - \mathbf{C}_a\mathbf{w}_a \tag{5.114}$$

Substituting Eq. (5.114) in (5.102) yields

$$\mathbf{C}^H\mathbf{w}_q - \mathbf{C}^H\mathbf{C}_a\mathbf{w}_a = \mathbf{g}$$

which, by virtue of Eq. (5.103), reduces to

$$\mathbf{C}^H\mathbf{w}_q = \mathbf{g} \tag{5.115}$$

Equation (5.115) shows that the weight vector $\mathbf{w}_q$ is that part of the weight vector $\mathbf{w}$ that satisfies the constraints. In contrast, the vector $\mathbf{w}_a$ is unaffected by the constraints; it therefore provides the degrees of freedom built into the design of the beamformer. Thus, in light of Eq. (5.114), the beamformer may be represented by the block diagram shown in Fig. 5.11(a). The beamformer described herein is referred to as a *generalized sidelobe canceler* (GSC).[7]

In light of Eq. (5.115), we may now perform an unconstrained minimization of the mean-square value of the beamformer output $y(n)$ with respect to the adjustable weight vector $\mathbf{w}_a$. According to Eq. (5.86), the beamformer output is defined by the inner product

$$y(n) = \mathbf{w}^H\mathbf{u}(n) \tag{5.116}$$

where $\mathbf{u}(n)$ is the input signal vector:

$$\mathbf{u}(n) = u_0(n)\,[1, e^{-j\phi_0}, \ldots, e^{-j(M-1)\phi_0}]^T \tag{5.117}$$

where the electrical angle ϕ_0 is defined by the direction of arrival of the incoming plane wave, and $u_0(n)$ is the electrical signal picked up by antenna element 0 of the linear array in Fig. 5.10 at time n. Hence, substituting Eq. (5.114) in (5.116) yields

$$y(n) = \mathbf{w}_q^H\mathbf{u}(n) - \mathbf{w}_a^H\mathbf{C}_a^H\mathbf{u}(n) \tag{5.118}$$

Define

$$\mathbf{w}_q^H\mathbf{u}(n) = d(n) \tag{5.119}$$

$$\mathbf{C}_a^H\mathbf{u}(n) = \mathbf{x}(n) \tag{5.120}$$

We may then rewrite Eq. (5.118) in a form that resembles the standard Wiener filter exactly, as shown by

$$y(n) = d(n) - \mathbf{w}_a^H\mathbf{x}(n) \tag{5.121}$$

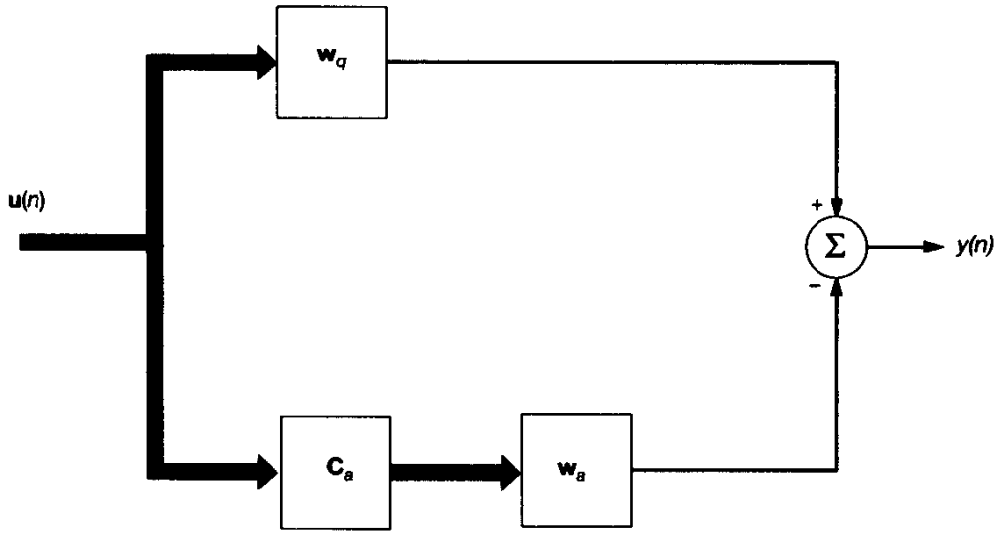
where $d(n)$ plays the role of a "desired response" for the GSC and $\mathbf{x}(n)$ plays the role of input vector, as depicted in Fig. 5.11(b). We thus see that the combined use of vector $\mathbf{w}_q$ and matrix $\mathbf{C}_a$ has converted the linearly constrained optimization problem into a standard optimum filtering problem. In particular, we now have an unconstrained optimization problem involving the adjustable portion $\mathbf{w}_a$ of the weight vector, which may be formally written as

$$\min_{\mathbf{w}_a} E[\,|y(n)|^2] = \min_{\mathbf{w}_a} (\sigma_d^2 - \mathbf{p}_x^H\mathbf{w}_a - \mathbf{w}_a^H\mathbf{p}_x + \mathbf{w}_a^H\mathbf{R}_x\mathbf{w}_a) \tag{5.122}$$

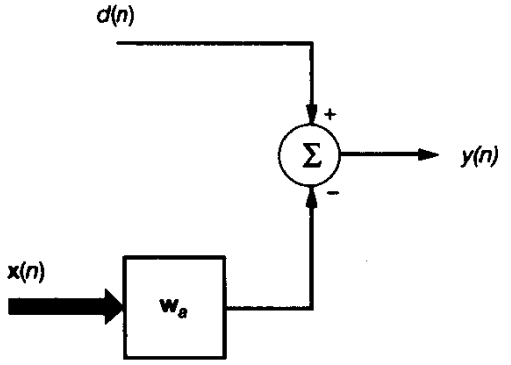
where the $(M-L)$-by-1 vector $\mathbf{p}_x$ is defined by

$$\mathbf{p}_x = E[\mathbf{x}(n)d^*(n)] \tag{5.123}$$

[7] The essence of the generalized sidelobe canceler may be traced back to a method for solving linearly constrained quadratic minimization problems originally proposed by Hanson and Lawson (1969). The term "generalized sidelobe canceler" was coined by Griffiths and Jim (1982). For a discussion of the generalized sidelobe canceler, see Van Veen and Buckley (1988) and Van Veen (1992).



(a)



(b)

Figure 5.11 (a) Block diagram of generalized sidelobe canceler. (b) Reformulation of the generalized sidelobe canceling problem as a standard optimum filtering problem.

and the $(M-L)$-by-$(M-L)$ matrix $\mathbf{R}_x$ is defined by

$$\mathbf{R}_x = E[\mathbf{x}(n)\mathbf{x}^H(n)] \tag{5.124}$$

The cost function of Eq. (5.122) is a quadratic in the unknown vector $\mathbf{w}_a$, which, as previously stated, embodies the available degrees of freedom in the GSC. Most importantly, this cost function has exactly the same mathematical form as that of the standard Wiener filter defined in Eq. (5.50). Accordingly, we may readily use our previous results to obtain the optimum value of $\mathbf{w}_a$ as

$$\mathbf{w}_{ao} = \mathbf{R}_x^{-1}\mathbf{p}_x \tag{5.125}$$

Using the definitions of Eqs. (5.119) and (5.120) in Eq. (5.123), we may express the vector $\mathbf{p}_x$ as

$$\begin{aligned}\mathbf{p}_x &= E[\mathbf{C}_a^H\mathbf{u}(n)\mathbf{u}^H(n)\mathbf{w}_q] \\ &= \mathbf{C}_a^H E[\mathbf{u}(n)\mathbf{u}^H(n)]\mathbf{w}_q \\ &= \mathbf{C}_a^H\mathbf{R}\mathbf{w}_q\end{aligned} \tag{5.126}$$

where $\mathbf{R}$ is the correlation matrix of the incoming data vector $\mathbf{u}(n)$. Similarly, using the definition of Eq. (5.120) in (5.124), we may express the matrix $\mathbf{R}_x$ as

$$\begin{aligned}\mathbf{R}_x &= E[\mathbf{C}_a^H\mathbf{u}(n)\mathbf{u}^H(n)\mathbf{C}_a] \\ &= \mathbf{C}_a^H\mathbf{R}\mathbf{C}_a\end{aligned} \tag{5.127}$$

The matrix $\mathbf{C}_a$ has full rank, and the correlation matrix $\mathbf{R}$ is positive definite since the incoming data always contain some form of additive receiver noise. Accordingly, we may rewrite the optimum solution $\mathbf{w}_{ao}$ of Eq. (5.125) as

$$\mathbf{w}_{ao} = (\mathbf{C}_a^H\mathbf{R}\mathbf{C}_a)^{-1}\mathbf{C}_a^H\mathbf{R}\mathbf{w}_q \tag{5.128}$$

Let P_o denote the minimum output power of the GSC attained by using the optimum solution $\mathbf{w}_{ao}$. Then, adapting the previous result derived in Eq. (5.49) for the standard Wiener filter and proceeding in a manner similar to that described above, we may express P_o as follows:

$$\begin{aligned}P_o &= \sigma_d^2 - \mathbf{p}_x^H\mathbf{R}_x^{-1}\mathbf{p}_x \\ &= \mathbf{w}_q^H\mathbf{R}\mathbf{w}_q - \mathbf{w}_q^H\mathbf{R}\mathbf{C}_a(\mathbf{C}_a^H\mathbf{R}\mathbf{C}_a)^{-1}\mathbf{C}_a^H\mathbf{R}\mathbf{w}_q\end{aligned} \tag{5.129}$$

Consider the special case of a *quiet environment*, for which the received signal consists of white noise acting alone. Let the corresponding value of the correlation matrix $\mathbf{R}$ be written as

$$\mathbf{R} = \sigma^2\mathbf{I} \tag{5.130}$$

where $\mathbf{I}$ is the M-by-M identity matrix, and σ^2 is the noise variance. Under this condition, we readily find from Eq. (5.128) that

$$\mathbf{w}_{ao} = (\mathbf{C}_a^H\mathbf{C}_a)^{-1}\mathbf{C}_a^H\mathbf{w}_q$$

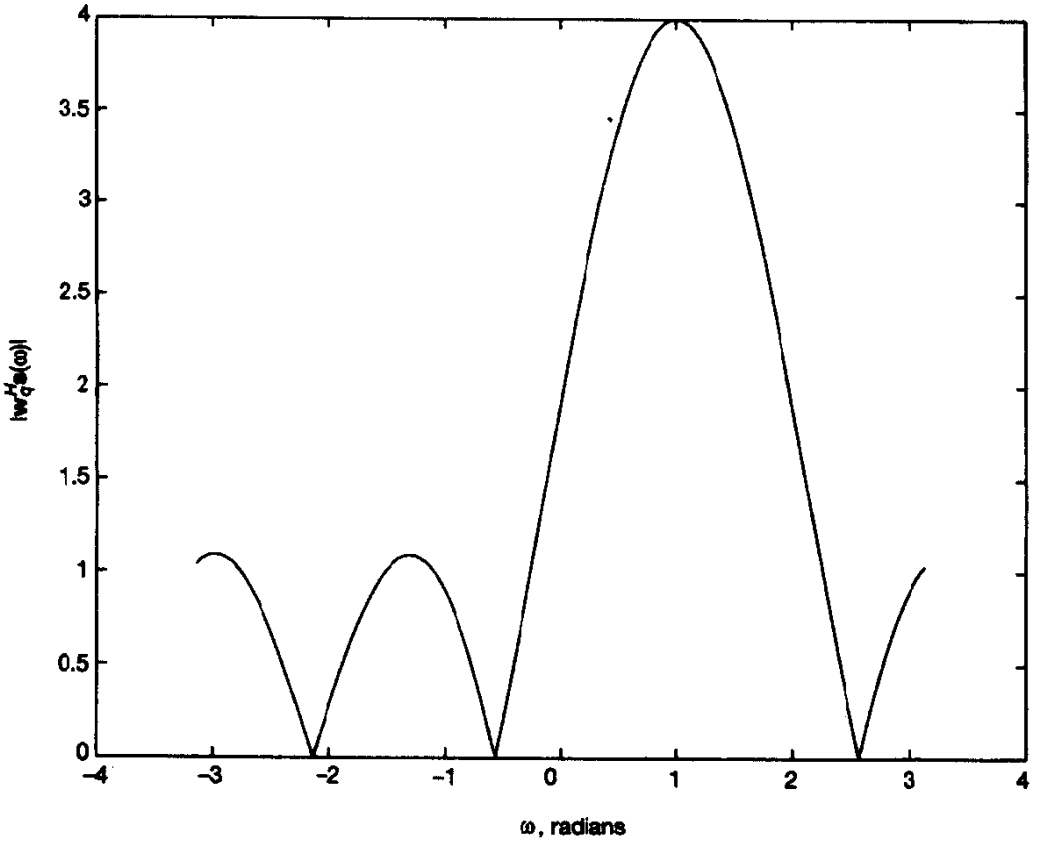


Figure 5.12a Interpretation of $\mathbf{w}_q^H \mathbf{s}(\omega)$ as the response of an FIR filter.

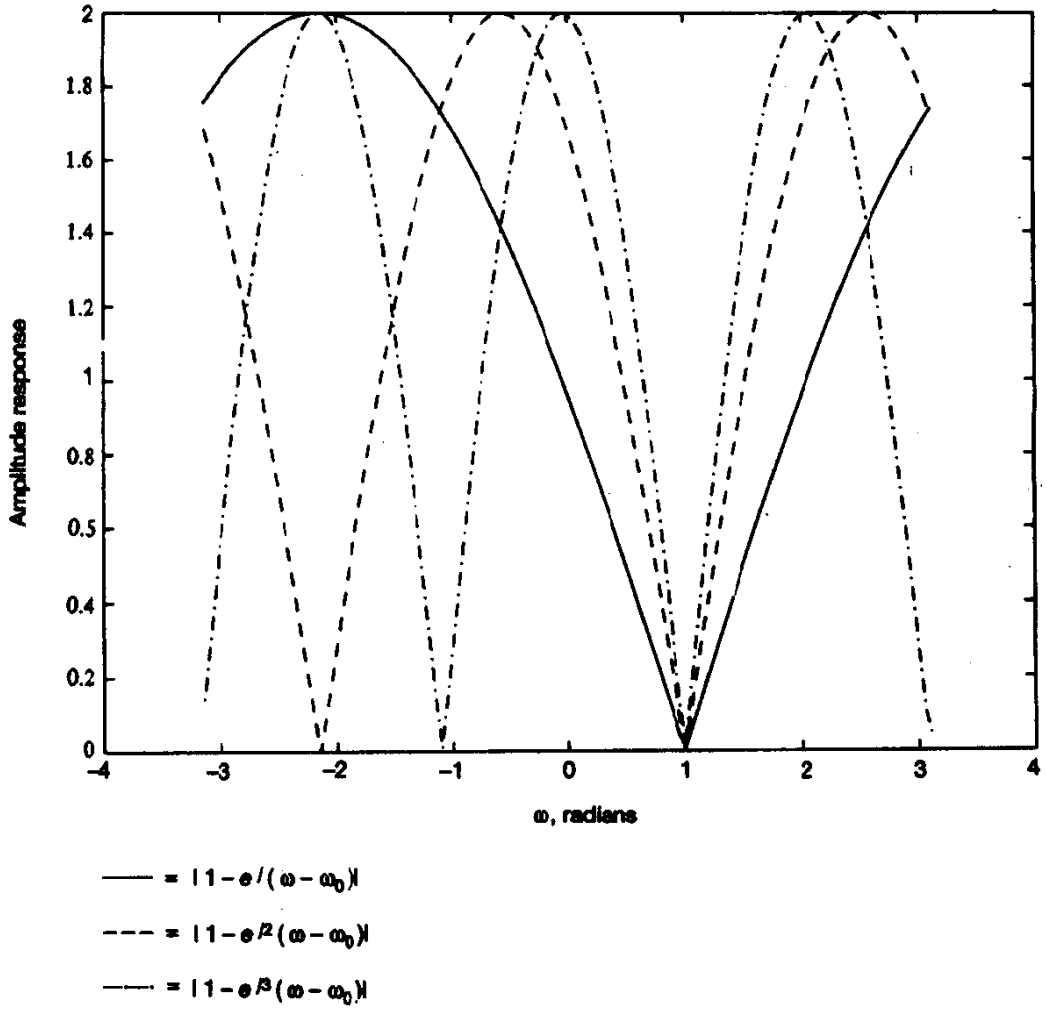


Figure 5.12b Interpretation of each column of matrix $\mathbf{C}_a$ as a band-rejection filter. In both parts of the figure, part a on the previous page and part b on this page, it is assumed that $\omega_0 = 1$.

By definition, the weight vector $\mathbf{w}_q$ is orthogonal to the columns of matrix $\mathbf{C}_a$. It follows therefore that the optimum weight vector $\mathbf{w}_{ao}$ is identically zero for a quiet environment described by Eq. (5.130). Thus, with $\mathbf{w}_{ao}$ equal to zero, we find from Eq. (5.114) that $\mathbf{w} = \mathbf{w}_q$. It is for this reason that $\mathbf{w}_q$ is often referred to as the *quiescent weight vector*, hence the use of subscript q for denoting it.

Filtering Interpretations of $\mathbf{w}_q$ and $\mathbf{C}_a$

The quiescent weight vector $\mathbf{w}_q$ and matrix $\mathbf{C}_a$ play critical roles of their own in the operation of the GSC. To develop physical interpretations of them, consider an MVDR spectrum estimator (formulated in temporal terms) for which we have

$$\begin{aligned} \mathbf{C} &= \mathbf{s}(\omega_0) \\ &= [1, e^{-j\omega_0}, \ldots, e^{-j(M-1)\omega_0}]^T \end{aligned} \tag{5.131}$$

and

$$\mathbf{g} = 1$$

Hence, the use of these values in Eq. (5.113) yields the corresponding value of the quiescent weight vector to be

$$\begin{aligned} \mathbf{w}_q &= \mathbf{C}(\mathbf{C}^H\mathbf{C})^{-1}\mathbf{g} \\ &= \frac{1}{M}[1, e^{-j\omega_0}, \ldots, e^{-j(M-1)\omega_0}]^T \end{aligned} \tag{5.132}$$

which represents an FIR filter of length M. The frequency response of this filter is given by

$$\begin{aligned} \mathbf{w}_q^H\mathbf{s}(\omega) &= \frac{1}{M}\sum_{k=0}^{M-1} e^{jk(\omega_0-\omega)} \\ &= \frac{1 - e^{jM(\omega_0-\omega)}}{1 - e^{j(\omega_0-\omega)}} \\ &= \left(\frac{\sin\left(\frac{M}{2}(\omega_0 - \omega)\right)}{\sin\left(\frac{1}{2}(\omega_0 - \omega)\right)}\right) \exp\left(\frac{j(M-1)}{2}(\omega_0 - \omega)\right) \end{aligned} \tag{5.133}$$

Figure 5.12(a) shows the amplitude response of this filter for $M = 4$ and $\omega_0 = 1$. From this figure we clearly see that the FIR filter representing the quiescent weight vector $\mathbf{w}_q$ acts like a bandpass filter tuned to the angular frequency ω_0, for which the MVDR spectrum estimator is constrained to produce a distortionless response.

Consider next a physical interpretation of the matrix $\mathbf{C}_a$. The use of Eq. (5.131) in (5.103) yields

$$\mathbf{s}^H(\omega_0)\mathbf{C}_a = \mathbf{0} \tag{5.134}$$

According to Eq. (5.134), each of the $(M-L)$ columns of matrix $\mathbf{C}_a$ represents an FIR filter with an amplitude response that is zero at ω_0 as illustrated in Fig. 5.12b for $\omega_0 = 1$, $M = 4$, $L = 1$, and

$$\mathbf{C}_a = \begin{bmatrix} -1 & -1 & -1 \\ e^{-j\omega_0} & 0 & 0 \\ 0 & e^{-j2\omega_0} & 0 \\ 0 & 0 & e^{-j3\omega_0} \end{bmatrix}$$

In other words, the matrix $\mathbf{C}_a$ is represented by a bank of band-rejection filters, each of which is tuned to ω_0. Thus, the matrix $\mathbf{C}_a$ is referred to as a *signal-blocking matrix*, since it blocks (rejects) the received signal at the angular frequency ω_0. The function of the matrix $\mathbf{C}_a$ is to cancel interference that leaks through the sidelobes of the band-pass filter representing the quiescent weight vector $\mathbf{w}_q$.

5.10 SUMMARY AND DISCUSSION

The discrete-time version of the Wiener filter theory, as described in this chapter, has evolved from the pioneering work of Norbert Wiener on linear optimum filters for continuous-time signals. The importance of the Wiener filter lies in the fact that it provides a frame of reference for the linear filtering of stochastic signals, assuming wide-sense stationarity.

The filtering structures that fall under the umbrella of Wiener filter theory are of two different physical types:

- Transversal filters, which are characterized by an impulse response of finite duration
- Narrow-band beamformers, which consist of a set of uniformly spaced antenna elements with adjustable weights.

These two structures share a common feature: they are both examples of a linear device whose output is defined by the inner product of its weight vector and the input vector. The optimum filter involving such a structure is embodied in the Wiener–Hopf equations, the solution of which involves two ensemble-averaged parameters:

- The correlation matrix of the input vector
- The cross-correlation vector between the input vector and desired response

The standard formulation of Wiener filtering requires the availability of a desired response. There are, however, applications where it is not feasible to provide a desired response. In such situations, we may use a class of linear optimum filters known as linearly constrained minimum variance (LCMV) filters or LCMV beamformers, depending

on whether the application is temporal or spatial in nature. The essence of the LCMV approach is that it minimizes the average output power, subject to a set of linear constraints on the weight vector. The constraints are imposed so as to prevent the weight vector from canceling the signal of interest. In a special form of LCMV beamformers known as generalized sidelobe cancelers, the weight vector is separated into two components:

- A quiescent weight vector, which satisfies the prescribed constraints
- An unconstrained weight vector, the optimization of which in accordance with the Wiener filter theory minimizes the effects of receiver noise and interfering signals

PROBLEMS

1. A necessary condition for a function $f(z)$ of the complex variable $z = x + jy$ to be an analytic function is that its real part $u(x, y)$ and imaginary part $v(x, y)$ must satisfy the Cauchy–Riemann equations. For a discussion of complex variables, see Appendix A. Demonstrate that the cost function J defined in Eq. (5.43) is *not* an analytic function of the weights by doing the following:
   (a) Show that the product $p^*(-k)w_k$ is an analytic function of the complex tap-weight (filter coefficient) w_k.
   (b) Show that the second product $w_k^* p(-k)$ is *not* an analytic function.
2. Consider a Wiener filtering problem characterized as follows: The correlation matrix $\mathbf{R}$ of the tap-input vector $\mathbf{u}(n)$ is

$$\mathbf{R} = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}$$

   The cross-correlation vector $\mathbf{p}$ between the tap-input vector $\mathbf{u}(n)$ and the desired response $d(n)$ is

$$\mathbf{p} = \begin{bmatrix} 0.5 \\ 0.25 \end{bmatrix}$$

   (a) Evaluate the tap weights of the Wiener filter.
   (b) What is the minimum mean-squared error produced by this Wiener filter?
   (c) Formulate a representation of the Wiener filter in terms of the eigenvalues of matrix $\mathbf{R}$ and associated eigenvectors.
3. The tap-weight vector of a transversal filter is defined by

$$\mathbf{u}(n) = \alpha(n)\mathbf{s}(n) + \mathbf{v}(n)$$

   where

$$\mathbf{s}(\omega) = [1, e^{-j\omega}, \ldots, e^{-j\omega(M-1)}]^T$$

   and

$$\mathbf{v}(n) = [v(n), v(n-1), \ldots, v(n-M+1)]^T$$

   The complex amplitude of the sinusoidal vector $\mathbf{s}(\omega)$ is a random variable with zero mean and variance $\sigma_\alpha^2 = E[|\alpha(n)|^2]$.

(a) Determine the correlation matrix of the tap-input vector $\mathbf{u}(n)$.

(b) Suppose that the desired response $d(n)$ is uncorrelated with $\mathbf{u}(n)$. What is the value of the tap-weight vector of the corresponding Wiener filter?

(c) Suppose that the variance σ_v^2 is zero, and the desired response is defined by

$$d(n) = v(n - k)$$

where $0 \leq k \leq M - 1$. What is the new value of the tap-weight vector of the Wiener filter?

(d) Determine the tap-weight vector of the Wiener filter for a desired response defined by

$$d(n) = \alpha(n)e^{-j\omega\tau}$$

where τ is a prescribed delay.

4. Show that the Wiener–Hopf equations (5.34), defining the tap-weight vector $\mathbf{w}_o$ of the Wiener filter, and Eq. (5.49), defining the minimum mean-squared error $J_{\min}$, may be combined into a single matrix relation

$$\mathbf{A}\begin{bmatrix} 1 \\ -\mathbf{w}_o \end{bmatrix} = \begin{bmatrix} J_{\min} \\ \mathbf{0} \end{bmatrix}$$

The matrix $\mathbf{A}$ is the correlation matrix of the augmented vector

$$\begin{bmatrix} d(n) \\ \mathbf{u}(n) \end{bmatrix}$$

where $d(n)$ is the desired response and $\mathbf{u}(n)$ is the tap-input vector of the Wiener filter.

5. The minimum mean-squared error $J_{\min}$ is defined by [see Eq. (5.49)]

$$J_{\min} = \sigma_d^2 - \mathbf{p}^H\mathbf{R}^{-1}\mathbf{p}$$

where σ_d^2 is the variance of the desired response $d(n)$, $\mathbf{R}$ is the correlation matrix of the tap-input vector $\mathbf{u}(n)$, and $\mathbf{p}$ is the cross-correlation vector between $\mathbf{u}(n)$ and $d(n)$. By applying the unitary similarity transformation to the inverse of the correlation matrix, that is, $\mathbf{R}^{-1}$, show that

$$J_{\min} = \sigma_d^2 - \sum_{k=1}^{M} \frac{|\mathbf{q}_k^H\mathbf{p}|^2}{\lambda_k}$$

where λ_k is the kth eigenvalue of the correlation matrix $\mathbf{R}$, and $\mathbf{q}_k$ is the corresponding eigenvector. Note that $\mathbf{q}_k^H\mathbf{p}$ is a scalar.

6. In this problem, we explore the extent of the improvement that may result from using a more complex Wiener filter for the environment described in Section 5.6. To be specific, the new formulation of the Wiener filter has three taps.

(a) Find the 3-by-3 correlation matrix of the tap inputs of this filter and the 3-by-1 cross-correlation vector between the desired response and the tap inputs.

(b) Compute the 3-by-1 tap-weight vector of the Wiener filter, and also compute the new value for the minimum mean-squared error.

7. In this problem we explore an application of Wiener filtering to *radar*. The sampled form of the transmitted radar signal is $A_0e^{j\omega_0 n}$ where ω_0 is the transmitted angular frequency, and A_0 is the transmitted complex amplitude. The received signal is

$$\mathbf{u}(n) = A_1e^{-j\omega_1 n} + v(n)$$

where $|A_1| < |A_0|$ and ω_1 differs from ω_0 by virtue of the *Doppler* shift produced by the motion of a target of interest, and $v(n)$ is a sample of white noise.

(a) Show that the correlation matrix of the time series $u(n)$, made up of M elements, may be written as

$$\mathbf{R} = \sigma_v^2\mathbf{I} + \sigma_1^2\mathbf{s}(\omega_1)\mathbf{s}^H(\omega_1)$$

where σ_v^2 is the variance of the zero-mean white noise $v(n)$, and

$$\sigma_1^2 = E[|A_1|^2]$$

and

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

(b) The time series $u(n)$ is applied to an M-tap Wiener filter with the cross-correlation vector $\mathbf{p}$ between $u(n)$ and the desired response $d(n)$ preset to

$$\mathbf{p} = \sigma_0^2\mathbf{s}(\omega_0)$$

where

$$\sigma_0^2 = E[|A_0|^2]$$

and

$$\mathbf{s}(\omega_0) = [1, e^{-j\omega_0}, \ldots, e^{-j\omega_0(M-1)}]^T$$

Derive an expression for the tap-weight vector of the Wiener filter.

8. An array processor consists of a primary sensor and a reference sensor interconnected with each other. The output of the reference sensor is weighted by w and then subtracted from the output of the primary sensor. Show that the mean-square value of the output of the array processor is minimized when the weight w attains the optimum value

$$w_o = \frac{E[u_1(n)u_2^*(n)]}{E[|u_2(n)|^2]}$$

where $u_1(n)$ and $u_2(n)$ are the primary- and reference-sensor outputs at time n, respectively.

9. Consider a discrete-time stochastic process $u(n)$ that consists of K (uncorrelated) complex sinusoids plus additive white noise of zero mean and variance σ^2. That is,

$$u(n) = \sum_{k=1}^{K} A_k e^{j\omega_k n} + v(n)$$

where the terms $A_k\exp(j\omega_k n)$ and $v(n)$ refer to the kth sinusoid and noise, respectively. The process $u(n)$ is applied to a transversal filter with M taps, producing the output

$$e(n) = \mathbf{w}^H\mathbf{u}(n)$$

Assume that $M > K$. The requirement is to choose the tap-weight vector $\mathbf{w}$ so as to minimize the mean-square value of $e(n)$, subject to the multiple signal-protection constraint

$$\mathbf{S}^H\mathbf{w} = \mathbf{D}^{1/2}\mathbf{1}$$

where $\mathbf{S}$ is the M-by-K signal matrix whose kth column has 1, $\exp(j\omega_k), \ldots, \exp[j\omega_k(M-1)]$ for its elements, $\mathbf{D}$ is the K-by-K diagonal matrix whose nonzero elements equal the average powers of the individual sinusoids, and the K-by-1 vector $\mathbf{1}$ has 1's for all its K elements. Using

the method of Lagrange multipliers, show that the value of the optimum weight vector that results from this constrained optimization equals

$$\mathbf{w}_o = \mathbf{R}^{-1}\mathbf{S}(\mathbf{S}^H\mathbf{R}^{-1}\mathbf{S})^{-1}\mathbf{D}^{1/2}\mathbf{1}$$

where $\mathbf{R}$ is the correlation matrix of the M-by-1 tap-input vector $\mathbf{u}(n)$. This formula represents a temporal generalization of the MVDR formula.

10. The weight vector $\mathbf{w}$ of the LCMV beamformer is defined by Eq. (5.96). In general, the LCMV beamformer so defined does *not* maximize the output signal-to-noise ratio. To be specific, let the input vector $\mathbf{u}(n)$ be written as

$$\mathbf{u}(n) = \mathbf{s}(n) + \mathbf{v}(n)$$

where the vector $\mathbf{s}(n)$ represents the signal component, and the vector $\mathbf{v}(n)$ represents the additive noise component. Show that the weight vector $\mathbf{w}$ does *not* satisfy the condition

$$\max_{\mathbf{w}} \frac{\mathbf{w}^H\mathbf{R}_s\mathbf{w}}{\mathbf{w}^H\mathbf{R}_v\mathbf{w}}$$

where $\mathbf{R}_s$ is the correlation matrix of $\mathbf{s}(n)$, and $\mathbf{R}_v$ is the correlation matrix of $\mathbf{v}(n)$.

11. In this problem, we explore the design of constraints for a beamformer using a *nonuniformly spaced array* of antenna elements. Let t_i denote the propagation delay to the ith element for a plane wave impinging on the array from look direction θ; the delay t_i is measured with respect to the zero-time reference.

(a) Find the response of the beamformer with elemental weight vector $\mathbf{w}$ to a signal of angular frequency ω that originates from the look direction θ.

(b) Hence, specify the linear constraint imposed on the array to produce a response equal to g along the direction θ.

12. Consider the problem of detecting a known signal in the presence of additive noise. The noise is assumed to be Gaussian, to be independent of the signal, and to have zero mean and a positive definite correlation matrix $\mathbf{R}_v$. The aim of the problem is to show that under these conditions the three criteria: minimum mean-squared error, maximum signal-to-noise ratio, and the likelihood ratio test yield identical designs for the transversal filter.

Let $u(n)$, $n = 1, 2, \ldots, M$, denote a set of M complex-valued data samples. Let $v(n)$, $n = 1, 2, \ldots, M$, denote a set of samples taken from a Gaussian noise process of zero mean. Finally, let $s(n)$, $n = 1, 2, \ldots, M$, denote samples of the signal. The detection problem is to determine whether the input consists of signal plus noise or noise alone. That is, the two hypotheses to be tested for are

$$\text{hypothesis } H_2: \quad u(n) = s(n) + v(n), \qquad n = 1, 2, \ldots, M$$

$$\text{hypothesis } H_1: \quad u(n) = v(n), \qquad n = 1, 2, \ldots, M$$

(a) The *Wiener filter* minimizes the mean-squared error. Show that this criterion yields an optimum tap-weight vector for estimating s_k, the kth component of signal vector $\mathbf{s}$, that equals

$$\mathbf{w}_o = \frac{s_k}{1 + \mathbf{s}^H\mathbf{R}_v^{-1}\mathbf{s}}\mathbf{R}_v^{-1}\mathbf{s}$$

Hint: To evaluate the inverse of the correlation matrix of $u(n)$ under hypothesis H_2, you may use the matrix inversion lemma. Let

$$\mathbf{A} = \mathbf{B}^{-1} + \mathbf{C}\mathbf{D}^{-1}\mathbf{C}^H$$

where **A**, **B** and **D** are positive-definite matrices. Then

$$\mathbf{A}^{-1} = \mathbf{B} - \mathbf{BC}(\mathbf{D} + \mathbf{C}^H \mathbf{BC})^{-1}\mathbf{C}^H\mathbf{B}$$

(b) The *maximum signal-to-noise ratio filter* maximizes the ratio

$$\rho = \frac{\text{average power of filter output due to signal}}{\text{average power of filter output due to noise}}$$

$$= \frac{E[(\mathbf{w}^H\mathbf{s})^2]}{E[(\mathbf{w}^H\mathbf{v})^2]}$$

Show that the tap-weight vector for which the output signal-to-noise ratio ρ is at maximum equals

$$\mathbf{w}_{SN} = \mathbf{R}_v^{-1}\mathbf{s}$$

Hint: Since $\mathbf{R}_v$ is positive definite, you may use $\mathbf{R}_v = \mathbf{R}_v^{1/2}\mathbf{R}_v^{1/2}$.

(c) The *likelihood ratio processor* computes the log-likelihood ratio and compares it to a threshold. If the threshold is exceeded, it decides in favor of hypothesis H_2; otherwise, it decides in favor of hypothesis H_1. The likehood ratio is defined by

$$\Lambda = \frac{\mathbf{f}_U(\mathbf{u}|H_2)}{\mathbf{f}_U(\mathbf{u}|H_1)}$$

where $\mathbf{f}_U(\mathbf{u}|H_i)$ is the conditional joint probability density function of the observation vector $\mathbf{u}$, given that hypothesis H_i is true, where $i = 1, 2$. Show that the likelihood ratio test is equivalent to the test

$$\mathbf{w}_{ml}^H\mathbf{u} \underset{H_1}{\overset{H_2}{\lesseqgtr}} \eta$$

where η is the threshold and

$$\mathbf{w}_{ml} = \mathbf{R}_v^{-1}\mathbf{s}$$

Hint: Refer to Section 2.11 for the joint probability function of the M-by-1 Gaussian noise vector $\mathbf{v}$ with zero mean and correlation matrix $\mathbf{R}_v$.

CHAPTER

6

Linear Prediction

One of the most celebrated problems in time-series analysis is that of *predicting* a future value of a stationary discrete-time stochastic process, given a set of past samples of the process. To be specific, consider the time series $u(n), u(n-1), \ldots, u(n-M)$, representing $(M+1)$ samples of such a process up to and including time n. The operation of prediction may, for example, involve using the samples $u(n-1), u(n-2), \ldots, u(n-M)$ to make an estimate of $u(n)$. Let $\mathcal{U}_{n-1}$ denote the M-dimensional space spanned by the samples $u(n-1), u(n-2), \ldots, u(n-M)$, and use $\hat{u}(n|\mathcal{U}_{n-1})$ to denote the *predicted value* of $u(n)$ given this set of samples. In *linear prediction*, we express this predicted value as a linear combination of the samples $u(n-1), u(n-2), \ldots, u(n-M)$. This operation corresponds to one-step prediction into the future, measured with respect to time $n-1$. Accordingly, we refer to this form of prediction as *one-step linear prediction in the forward direction* or simply *forward linear prediction*. In another form of prediction, we use the samples $u(n), u(n-1), \ldots, u(n-M+1)$ to make a prediction of the past sample $u(n-M)$. We refer to this second form of prediction as *backward linear prediction*.[1]

In this chapter, we study forward linear prediction (FLP) as well as backward linear prediction (BLP). In particular, we use the Wiener filter theory of Chapter 5 to optimize

[1] The term "backward prediction" is somewhat of a misnomer. A more appropriate description for this operation is "hindsight." Correspondingly, the use of "forward" in the associated operation of forward prediction is superfluous. Nevertheless, the terms "forward prediction" and "backward prediction" have become deeply embedded in the literature on linear prediction.

the design of a forward or backward *predictor* in the mean-square sense for the case of a wide-sense stationary discrete-time stochastic process. As explained in Chapter 2, the correlation matrix of such a process has a Toeplitz structure. We will put this Toeplitz structure to good use in developing algorithms that are computationally efficient.

6.1 FORWARD LINEAR PREDICTION

Figure 6.1(a) shows a *forward predictor* that consists of a linear transversal filter with M tap weights $w_{f,1}, w_{f,2}, \ldots, w_{f,M}$ and tap inputs $u(n-1), u(n-2), \ldots, u(n-M)$, respectively. We assume that these tap inputs are drawn from a wide-sense stationary stochastic process of zero mean. We further assume that the tap weights are optimized in the mean-square sense in accordance with the Wiener filter theory. The predicted value $\hat{u}(n|\mathcal{U}_{n-1})$ is defined by

$$\hat{u}(n|\mathcal{U}_{n-1}) = \sum_{k=1}^{M} w_{f,k}^{*} u(n-k) \tag{6.1}$$

For the situation described herein, the desired response $d(n)$ equals $u(n)$, representing the actual sample of the input process at time n. We may thus write

$$d(n) = u(n) \tag{6.2}$$

The *forward prediction error* equals the difference between the input sample $u(n)$ and its predicted value $\hat{u}(n|\mathcal{U}_{n-1})$. We denote the forward prediction error by $f_M(n)$ and thus write

$$f_M(n) = u(n) - \hat{u}(n|\mathcal{U}_{n-1}) \tag{6.3}$$

The subscript M in the symbol for the forward prediction error signifies *order* of the predictor, defined as *the number of unit-delay elements needed to store the given set of samples used to make the prediction*. The reason for using the subscript will become apparent later in the chapter.

Let P_M denote the *minimum mean-squared prediction error*:

$$P_M = E[|f_M(n)|^2] \tag{6.4}$$

With the tap inputs assumed to have zero mean, the forward prediction error $f_M(n)$ will likewise have zero mean. Under this condition, P_M will also equal the variance of the forward prediction error. Yet another interpretation for P_M is that it may be viewed as the ensemble-averaged *forward prediction error power*, assuming that $f_M(n)$ is developed across a 1-Ω load. We will use the latter description to refer to P_M.

Let $\mathbf{w}_f$ denote the M-by-1 optimum tap-weight vector of the forward predictor in Fig. 6.1(a). We write it in expanded form as

$$\mathbf{w}_f = [w_{f,1}, w_{f,2}, \ldots, w_{f,M}]^T \tag{6.5}$$

To solve the Wiener–Hopf equations for the weight vector $\mathbf{w}_f$, we require knowledge of two quantities: (1) the M-by-M correlation matrix of the tap inputs $u(n-1), u(n-2),$

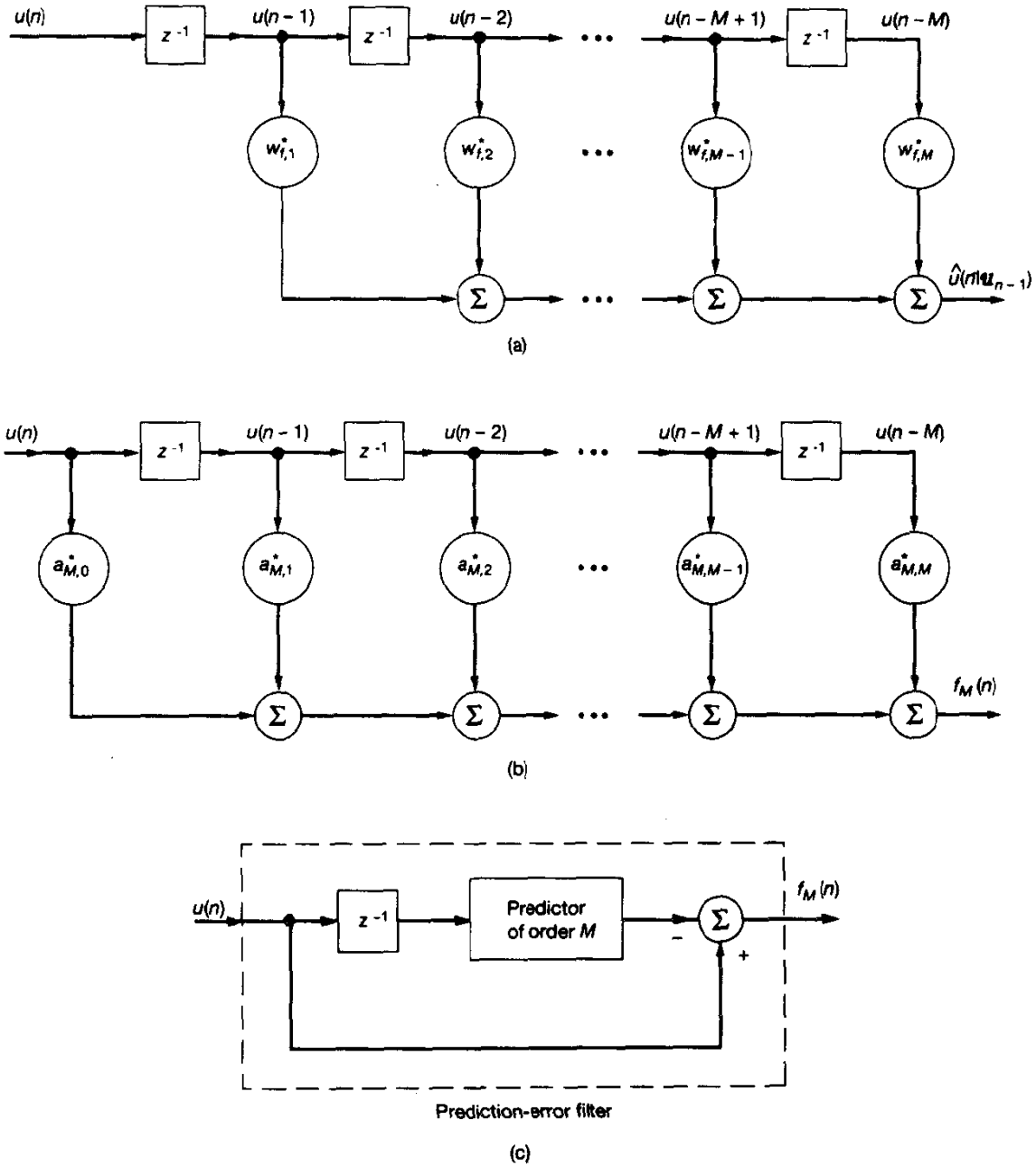


Figure 6.1 (a) One-step predictor; (b) prediction-error filter; (c) relationship between the predictor and the prediction-error filter.

$\ldots, u(n-M)$, and (2) the M-by-1 cross-correlation vector between these tap inputs and the desired response $u(n)$. To evaluate P_M, we require a third quantity, the variance of $u(n)$. We now consider these three quantities, one by one:

1. The tap inputs $u(n-1), u(n-2), \ldots, u(n-M)$ define the M-by-1 tap-input vector, $\mathbf{u}(n-1)$, as shown by

$$\mathbf{u}(n-1) = [u(n-1), u(n-2), \ldots, u(n-M)]^T \tag{6.6}$$

Hence, the correlation matrix of the tap inputs equals

$$\begin{aligned}\mathbf{R} &= E[\mathbf{u}(n-1)\mathbf{u}^H(n-1)] \\ &= \begin{bmatrix} r(0) & r(1) & \cdots & r(M-1) \\ r^*(1) & r(0) & \cdots & r(M-2) \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ r^*(M-1) & r^*(M-2) & \cdots & r(0) \end{bmatrix}\end{aligned} \tag{6.7}$$

where $r(k)$ is the autocorrelation function of the input process for lag k, where $k = 0, 1, \ldots, M-1$. Note that the symbol used for the correlation matrix of the tap inputs in Fig. 6.1(a) is the same as that for the correlation matrix of the tap inputs in the transversal filter of Fig. 5.4. We are justified to do this since the input process in both cases is assumed to be wide-sense stationary, so the correlation matrix of the process is invariant to a time shift.

2. The cross-correlation vector between the tap inputs $u(n-1), \ldots, u(n-M)$ and the desired response $u(n)$ equals

$$\begin{aligned}\mathbf{r} &= E[\mathbf{u}(n-1)u^*(n)] \\ &= \begin{bmatrix} r^*(1) \\ r^*(2) \\ \cdot \\ \cdot \\ \cdot \\ r^*(M) \end{bmatrix} = \begin{bmatrix} r(-1) \\ r(-2) \\ \cdot \\ \cdot \\ \cdot \\ r(-M) \end{bmatrix}\end{aligned} \tag{6.8}$$

3. The variance of $u(n)$ equals $r(0)$, since $u(n)$ has zero mean.

In Table 6.1, we summarize the various quantities pertaining to the Wiener filter of Fig. 5.4 and the corresponding quantities pertaining to the forward predictor of Fig. 6.1(a). The last column of this table pertains to the backward predictor, on which more will be said later.

TABLE 6.1 SUMMARY OF WIENER FILTER VARIABLES

Quantity	Wiener filter of Fig. 5.4	Forward predictor of Fig. 6.1(a)	Backward predictor of Fig. 6.2(a)
Tap-input vector	$\mathbf{u}(n)$	$\mathbf{u}(n-1)$	$\mathbf{u}(n)$
Desired response	$d(n)$	$u(n)$	$u(n-M)$
Tap-weight vector	$\mathbf{w}_o$	$\mathbf{w}_f$	$\mathbf{w}_b$
Estimation error	$e(n)$	$f_M(n)$	$b_M(n)$
Correlation matrix of tap inputs	$\mathbf{R}$	$\mathbf{R}$	$\mathbf{R}$
Cross-correlation vector between tap inputs and desired response	$\mathbf{p}$	$\mathbf{r}$	$\mathbf{r}^{B*}$
Minimum mean-squared error	$J_{\min}$	P_M	P_M

Thus, using the correspondences of this table, we may adapt the Wiener–Hopf equations (5.45) to solve the forward linear prediction (FLP) problem for stationary inputs and so write

$$\mathbf{R}\mathbf{w}_f = \mathbf{r} \tag{6.9}$$

Similarly, the use of Eq. (5.49), together with Eq. (6.8), yields the following expression for the forward prediction-error power:

$$P_M = r(0) - \mathbf{r}^H\mathbf{w}_f \tag{6.10}$$

From Eqs. (6.8) and (6.9), we see that the M-by-1 tap-weight vector of the forward predictor and the forward prediction-error power are determined solely by the set of $(M + 1)$ autocorrelation function values of the input process for lags $0, 1, \ldots, M$.

Relation between Linear Prediction and Autoregressive Modeling

It is highly informative to compare the Wiener-Hopf equations (6.9) for linear prediction with the Yule–Walker equations (2.66) for an autoregressive (AR) model. We see that these two systems of simultaneous equations are of exactly the same mathematical form. Furthermore, Eq. (6.10) defining the average power (i.e., variance) of the forward prediction error is also of the same mathematical form as Eq. (2.71) defining the variance of the white-noise process used to excite the autoregressive model. For the case of an AR process for which we know the model order M, we may thus state that when a forward predictor is optimized in the mean-square sense, in theory, its tap weights take on the same values as the corresponding parameters of the process. This relationship should not be surprising since the equation defining the forward prediction error and the difference equation defining the autoregressive model have the same mathematical form. When the process is not autoregressive, however, the use of a predictor provides an approximation to the process.

Forward Prediction-Error Filter

The forward predictor of Fig. 6.1(a) consists of M unit-delay elements and M tap weights $w_{f,1}, w_{f,2}, \ldots, w_{f,M}$ that are fed with the respective samples $u(n-1), u(n-2), \ldots, u(n-M)$ as inputs. The resultant output is the predicted value of $u(n)$, which is defined by Eq. (6.1). Hence, substituting Eq. (6.1) in (6.3), we may express the forward prediction error as

$$f_M(n) = u(n) - \sum_{k=1}^{M} w_{f,k}^* u(n-k) \tag{6.11}$$

Let $a_{M,k}$, $k = 0, 1, \ldots, M$, denote the tap weights of a new transversal filter, which are related to the tap weights of the forward predictor as follows:

$$a_{M,k} = \begin{cases} 1, & k = 0 \\ -w_{f,k}, & k = 1, 2, \ldots, M \end{cases} \tag{6.12}$$

Then we may combine the two terms on the right-hand side of Eq. (6.11) into a single summation as follows:

$$f_M(n) = \sum_{k=0}^{M} a_{M,k}^* u(n-k) \tag{6.13}$$

This input–output relation is represented by the transversal filter shown in Fig. 6.1(b). A filter that operates on the set of samples $u(n), u(n-1), \ldots, u(n-M)$ to produce the forward prediction error $f_M(n)$ at its output is called a forward *prediction-error filter* (PEF).

The relationship between the forward prediction-error filter and the forward predictor is illustrated in block diagram form in Fig. 6.1(c). Note that the length of the prediction-error filter exceeds the length of the one-step prediction filter by 1. However, both filters have the same order, M, as they both involve the same number of delay elements for the storage of past data.

Augmented Wiener–Hopf Equations for Forward Prediction

The Wiener–Hopf equations (6.9) define the tap-weight vector of the forward predictor, while Eq. (6.10) defines the resulting forward prediction-error power P_M. We may combine these two equations into a single matrix relation as follows:

$$\begin{bmatrix} r(0) & \mathbf{r}^H \\ \mathbf{r} & \mathbf{R} \end{bmatrix} \begin{bmatrix} 1 \\ -\mathbf{w}_f \end{bmatrix} = \begin{bmatrix} P_M \\ \mathbf{0} \end{bmatrix} \tag{6.14}$$

where $\mathbf{0}$ is the M-by-1 null vector. The M-by-M correlation matrix $\mathbf{R}$ is defined in Eq. (6.7), and the M-by-1 correlation vector $\mathbf{r}$ is defined in Eq. (6.8). The partitioning of the $(M+1)$-by-$(M+1)$ correlation matrix on the left-hand side of Eq. (6.14) into the form shown therein was discussed in Section 2.3. Note that this $(M+1)$-by-$(M+1)$ matrix equals the correlation matrix of the tap inputs $u(n), u(n-1), \ldots, u(n-M)$ in the pre-

diction-error filter of Fig. 6.1(b). Moreover, the $(M + 1)$-by-1 coefficient vector on the left-hand side of Eq. (6.14) equals the *forward prediction-error filter vector*:

$$\mathbf{a}_M = \begin{bmatrix} 1 \\ -\mathbf{w}_f \end{bmatrix} \tag{6.15}$$

We may also express the matrix relation of Eq. (6.14) as a system of $(M + 1)$ simultaneous equations as follows:

$$\sum_{i=0}^{M} a_{M,i} r(l - i) = \begin{cases} P_M, & i = 0 \\ 0, & i = 1, 2, \ldots, M \end{cases} \tag{6.16}$$

We refer to Eq. (6.14) or (6.16) as the *augmented Wiener–Hopf equations* of a forward prediction-error filter of order M.

Example 1

For the case of a prediction-error filter of order $M = 1$, Eq. (6.14) yields a pair of simultaneous equations described by

$$\begin{bmatrix} r(0) & r(1) \\ r^*(1) & r(0) \end{bmatrix} \begin{bmatrix} a_{1,0} \\ a_{1,1} \end{bmatrix} = \begin{bmatrix} P_1 \\ 0 \end{bmatrix}$$

Solving for $a_{1,0}$, and $a_{1,1}$, we get

$$a_{1,0} = \frac{P_1}{\Delta_r} r(0)$$

$$a_{1,1} = -\frac{P_1}{\Delta_r} r^*(1)$$

where Δ_r is the determinant of the correlation matrix; thus

$$\begin{aligned} \Delta_r &= \begin{vmatrix} r(0) & r(1) \\ r^*(1) & r(0) \end{vmatrix} \\ &= r^2(0) - |r(1)|^2 \end{aligned}$$

But $a_{1,0}$ equals 1. Hence,

$$P_1 = \frac{\Delta_r}{r(0)}$$

$$a_{1,1} = -\frac{r^*(1)}{r(0)}$$

Consider next the case of a prediction-error filter of order $M = 2$. Equation (6.14) yields a system of three simultaneous equations, as shown by

$$\begin{bmatrix} r(0) & r(1) & r(2) \\ r^*(1) & r(0) & r(1) \\ r^*(2) & r^*(1) & r(0) \end{bmatrix} \begin{bmatrix} a_{2,0} \\ a_{2,1} \\ a_{2,2} \end{bmatrix} = \begin{bmatrix} P_2 \\ 0 \\ 0 \end{bmatrix}$$

Solving for $a_{2,0}$, $a_{2,1}$, and $a_{2,2}$, we get

$$a_{2,0} = \frac{P_2}{\Delta_r}[r^2(0) - |r(1)|^2]$$

$$a_{2,1} = -\frac{P_2}{\Delta_r}[r^*(1)r(0) - r(1)r^*(2)]$$

$$a_{2,2} = \frac{P_2}{\Delta_r}[(r^*(1))^2 - r(0)r^*(2)]$$

where Δ_r is the determinant of the correlatin matrix:

$$\Delta_r = \begin{vmatrix} r(0) & r(1) & r(2) \\ r^*(1) & r(0) & r(1) \\ r^*(2) & r^*(1) & r(0) \end{vmatrix}$$

The coefficient $a_{2,0}$ equals 1. Accordingly, we may express the prediction-error power P_2 as

$$P_2 = \frac{\Delta_r}{r^2(0) - |r(1)|^2}$$

and the prediction-error filter coefficients $a_{2,1}$ and $a_{2,2}$ as

$$a_{2,1} = -\frac{r^*(1)(r(0) - r(1)r^*(2))}{r^2(0) - |r(1)|^2}$$

$$a_{2,2} = \frac{(r^*(1))^2 - r(0)r^*(2)}{r^2(0) - |r(1)|^2}$$

6.2 BACKWARD LINEAR PREDICTION

The form of linear prediction considered in Section 6.1 is said to be in the *forward* direction. That is, given the time series $u(n)$, $u(n-1), \ldots, u(n-M)$, we use the subset of M samples $u(n-1), u(n-2), \ldots, u(n-M)$ to make a prediction of the sample $u(n)$. This operation corresponds to *one-step linear prediction into the future*, measured with respect to time $n-1$. Naturally, we may also operate on this time series in the *backward* direction. That is, we may use the subset of M samples $u(n), u(n-1), \ldots, u(n-M+1)$ to make a prediction of the sample $u(n-M)$. This second operation corresponds to *backward linear prediction by one step*, measured with respect to time $n-M+1$.

Let $\mathcal{U}_n$ denote the M-dimensional space spanned by $u(n), u(n-1), \ldots, u(n-M+1)$ that are used in making the backward prediction. Then, using this set of samples as tap inputs, we make a linear prediction of the sample $u(n-M)$, as shown by

$$\hat{u}(n-M|\mathcal{U}_n) = \sum_{k=1}^{M} w_{b,k}^* u(n-k+1) \tag{6.17}$$

where $w_{b,1}, w_{b,2}, \ldots, w_{b,M}$ are the tap weights. Figure 6.2(a) shows a representation of the

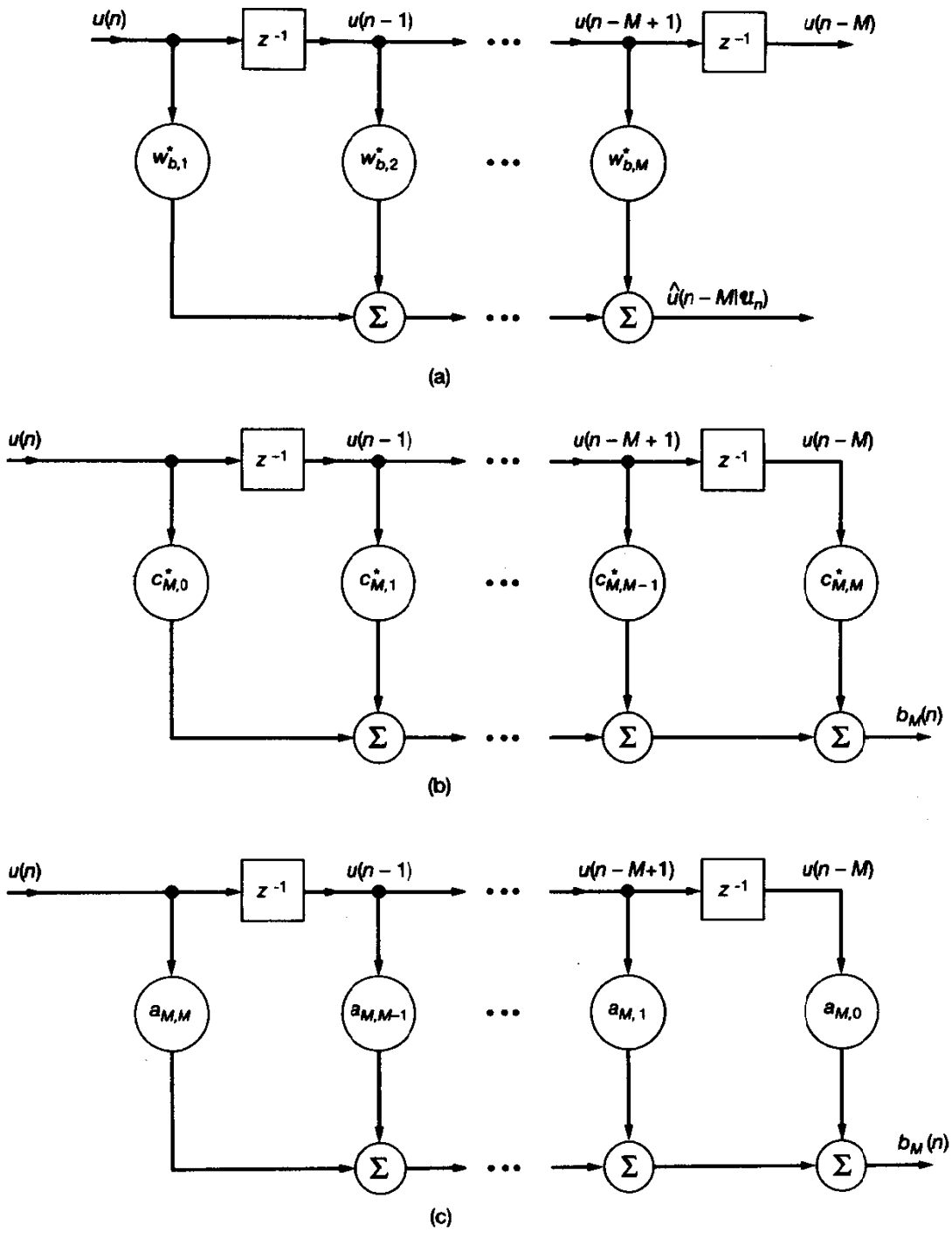


Figure 6.2 (a) Backward one-step predictor; (b) backward prediction-error filter; (c) backward prediction-error filter defined in terms of the tap weights of the corresponding forward prediction-error filter.

backward predictor as described by Eq. (6.17). We assume that these tap weights are optimized in the mean-square sense in accordance with the Wiener filter theory.

In the case of backward prediction, the desired response equals

$$d(n) = u(n - M) \tag{6.18}$$

The *backward prediction error* equals the difference between the actual sample value $u(n - M)$ and its predicted value $\hat{u}(n - M|\mathcal{U}_n)$. We denote the backward prediction error by $b_M(n)$ and thus write

$$b_M(n) = u(n - M) - \hat{u}(n - M|\mathcal{U}_n) \tag{6.19}$$

Here, again, the subscript M in the symbol for the backward prediction error $b_M(n)$ signifies the number of unit-delay elements needed to store the given set of samples used to make the prediction; that is, M is the order of the predictor.

Let P_M denote the *minimum mean-squared prediction error*

$$P_M = E[|b_M(n)|^2] \tag{6.20}$$

We may also view P_M as the ensemble-averaged *backward prediction-error power*, assuming that $b_M(n)$ is developed across a 1-Ω load.

Let $\mathbf{w}_b$ denote the M-by-1 optimum tap-weight vector of the backward predictor in Fig. 6.2(a). We express it in the expanded form

$$\mathbf{w}_b = [w_{b,1}, w_{b,2}, \ldots, w_{b,M}]^T \tag{6.21}$$

To solve the Wiener–Hopf equations for the weight vector $\mathbf{w}_b$, we require knowledge of two quantities: (1) the M-by-M correlation matrix of the tap inputs $u(n)$, $u(n - 1)$, . . . , $u(n - M + 1)$, and (2) the M-by-1 cross-correlation vector between the desired response $u(n - M)$ and these tap inputs. To evaluate P_M, we need a third quantity, the variance of $u(n - M)$. We consider these three quantities in turn:

1. Let $\mathbf{u}(n)$ denote the M-by-1 tap-input vector in the backward predictor of Fig. 6.2(a). We write it in expanded form as

$$\mathbf{u}(n) = [u(n), u(n - 1), \ldots, u(n - M + 1)]^T \tag{6.22}$$

The M-by-M correlation matrix of the tap inputs in Fig. 6.2(a) thus equals

$$\mathbf{R} = E[\mathbf{u}(n)\mathbf{u}^H(n)]$$

The expanded form of the correlation matrix $\mathbf{R}$ is given in Eq. (6.7).

2. The M-by-1 cross-correlation vector between the tap inputs $u(n)$, $u(n - 1)$, . . . , $u(n - M + 1)$ and the desired response $u(n - M)$ equals

$$\mathbf{r}^{B*} = E[\mathbf{u}(n)u^*(n - M)]$$

$$= \begin{bmatrix} r(M) \\ r(M - 1) \\ \cdot \\ \cdot \\ \cdot \\ r(1) \end{bmatrix} \tag{6.23}$$

The expanded form of the correlation vector $\mathbf{r}$ is given in Eq. (6.8). As usual, the superscript B denotes backward arrangement and the asterisk denotes complex conjugation.

3. The variance of the desired response $u(n - M)$ equals $r(0)$.

In the last column of Table 6.1, we summarize the various quantities pertaining to the backward predictor of Fig. 6.2(a).

Accordingly, using the correspondences of Table 6.1, we may adapt the Wiener–Hopf equations (5.34) to solve the backward linear prediction (BLP) problem for stationary inputs and so write

$$\mathbf{R}\mathbf{w}_b = \mathbf{r}^{B*} \tag{6.24}$$

Similarly, the use of Eq. (5.49), together with Eq. (6.24), yields the following expression for the backward prediction-error power:

$$P_M = r(0) - \mathbf{r}^{BT}\mathbf{w}_b \tag{6.25}$$

Here again we see that the M-by-1 tap-weight vector $\mathbf{w}_b$ of a backward predictor and the backward prediction-error power P_M are uniquely defined by knowledge of the set of autocorrelation function values of the process for lags $0, 1, \ldots, M$.

Relations between Backward and Forward Predictors

In comparing the two sets of Wiener–Hopf equations (6.9) and (6.24), pertaining to forward prediction and backward prediction, respectively, we see that the vector on the right-hand side of Eq. (6.24) differs from that of Eq. (6.9) in two respects: (1) its elements are arranged backward, and (2) they are complex conjugated. To correct for the first difference, we reverse the order in which the elements of the vector on the right-hand side of Eq. (6.24) are arranged. This operation has the effect of replacing the left-hand side of Eq. (6.24) by $\mathbf{R}^T\mathbf{w}_b^B$, where $\mathbf{R}^T$ is the transpose of the correlation matrix $\mathbf{R}$ and $\mathbf{w}_b^B$ is the backward version of the tap-weight vector $\mathbf{w}_b$ (see Problem 3). We may thus write

$$\mathbf{R}^T\mathbf{w}_b^B = \mathbf{r}^* \tag{6.26}$$

To correct for the remaining difference, we complex-conjugate both sides of Eq. (6.26), obtaining

$$\mathbf{R}^H\mathbf{w}_b^{B*} = \mathbf{r}$$

Since the correlation matrix $\mathbf{R}$ is Hermitian, that is, $\mathbf{R}^H = \mathbf{R}$, we may thus reformulate the Wiener–Hopf equations for backward prediction as

$$\mathbf{R}\mathbf{w}_b^{B*} = \mathbf{r} \tag{6.27}$$

Now we may compare Eq. (6.27) with Eq. (6.9) and thus deduce the following fundamental relationship between the tap-weight vectors of a backward predictor and the corresponding forward predictor:

$$\mathbf{w}_b^{B*} = \mathbf{w}_f \tag{6.28}$$

Equation (6.28) states that *we may modify a backward predictor into a forward predictor by reversing the sequence in which its tap weights are positioned and also complex-conjugating them.*

Next we wish to show that the ensemble-averaged error powers for backward prediction and forward prediction have exactly the same value. To do this, we first observe that the product $\mathbf{r}^{BT}\mathbf{w}_b$ equals $\mathbf{r}^T\mathbf{w}_b^B$, so we may rewrite Eq. (6.25) as

$$P_M = r(0) - \mathbf{r}^T\mathbf{w}_b^B \tag{6.29}$$

Taking the complex conjugate of both sides of Eq. (6.29), and recognizing that both P_M and $r(0)$ are unaffected by this operation since they are real-valued scalars, we get

$$P_M = r(0) - \mathbf{r}^H\mathbf{w}_b^{B*} \tag{6.30}$$

Comparing this result with Eq. (6.10) and using the equivalence of Eq. (6.28), we find that the backward prediction-error power has exactly the same value as the forward prediction-error power. Indeed, it is in anticipation of this equality that we have used the same symbol P_M to denote both the forward prediction-error power and the backward prediction-error power.

Backward Prediction-Error Filter

The backward prediction error $b_M(n)$ equals the difference between the desired response $u(n - M)$ and the linear prediction of it, given the samples $u(n)$, $u(n - 1)$, . . . , $u(n - M + 1)$. This prediction is defined by Eq. (6.17). Therefore, substituting Eq. (6.17) in (6.19), we get

$$b_M(n) = u(n - M) - \sum_{k=1}^{M} w_{bk}^* u(n - k + 1) \tag{6.31}$$

Define the tap weights of the backward prediction-error filter in terms of the corresponding backward predictor as follows:

$$c_{M,k} = \begin{cases} -w_{b,k+1} & k = 0, 1, \ldots, M - 1 \\ 1, & k = M \end{cases} \tag{6.32}$$

Hence, we may rewrite Eq. (6.31) as [see Fig. 6.2(b)]

$$b_M(n) = \sum_{k=0}^{M} c_{M,k}^* u(n - k) \tag{6.33}$$

Equation (6.28) defines the tap-weight vector of the backward predictor in terms of that of the forward predictor. We may express the scalar version of this relation as

$$w_{b,M-k+1}^* = w_{f,k}, \qquad k = 1, 2, \ldots, M$$

or, equivalently

$$w_{b,k} = w_{f,M-k+1}^* \qquad k = 1, 2, \ldots, M \tag{6.34}$$

Hence, substituting Eq. (6.34) in (6.32), we get

$$c_{M,k} = \begin{cases} -w^*_{f,M-k}, & k = 0, 1, \ldots, M-1 \\ 1, & k = M \end{cases} \tag{6.35}$$

Thus, using the relationship between the tap weights of the forward prediction-error filter and those of the forward predictor as given in Eq. (6.12), we may write

$$c_{M,k} = a^*_{M,M-k}, \qquad k = 0, 1, \ldots, M \tag{6.36}$$

Accordingly, we may express the input–output relation of the backward prediction-error filter in the equivalent form

$$b_M(n) = \sum_{k=0}^{M} a_{M,M-k} u(n-k) \tag{6.37}$$

The input–output relation of Eq. (6.37) is depicted in Fig. 6.2(c). Comparison of this representation for a backward prediction-error filter with that of Fig. 6.1(b) for the corresponding forward prediction-error filter reveals that these two forms of a prediction-error filter for *stationary inputs* are uniquely related to each other. In particular, *we may modify a forward prediction-error filter into the corresponding backward prediction-error filter by reversing the sequence in which the tap weights are positioned and complex-conjugating them.* Note that in both figures, the respective tap inputs have the same values.

Augmented Wiener–Hopf Equations for Backward Prediction

The set of Wiener–Hopf equations for backward prediction is defined by Eq. (6.24), and the resultant backward prediction-error power is defined by Eq. (6.25). We may combine these two equations into a single relation as follows:

$$\begin{bmatrix} \mathbf{R} & \mathbf{r}^{B*} \\ \mathbf{r}^{BT} & r(0) \end{bmatrix} \begin{bmatrix} -\mathbf{w}_b \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ P_M \end{bmatrix} \tag{6.38}$$

where $\mathbf{0}$ is the M-by-1 null vector. The M-by-M matrix $\mathbf{R}$ is the correlation matrix of the M-by-1 tap-input vector $\mathbf{u}(n)$; it has the expanded form shown in the second line of Eq. (6.7) by virtue of the assumed wide-sense stationarity of the input process. The M-by-1 vector $\mathbf{r}^{B*}$ is the cross-correlation vector between the input vector $\mathbf{u}(n)$ and the desired response $u(n-M)$; here again, the assumed wide-sense stationarity of the input process means that the vector $\mathbf{r}$ has the expanded form shown in the second line of Eq. (6.8). The $(M+1)$-by-$(M+1)$ matrix on the left-hand side of Eq. (6.38) equals the correlation matrix of the tap inputs in the backward prediction-error filter of Fig. 6.2(c). The partitioning of this $(M+1)$-by-$(M+1)$ into the form shown in Eq. (6.38) was discussed in Section 2.3.

We may also express the matrix relation of Eq. (6.38) as a system of $(M+1)$ simultaneous equations:

$$\sum_{l=0}^{M} a^*_{M,M-l} r(l-i) = \begin{cases} 0, & i = 0, \ldots, M-1 \\ P_M, & i = M \end{cases} \tag{6.39}$$

We refer to Eq. (6.38) or (6.39) as the *augmented Wiener–Hopf equations* of a backward prediction-error filter of order M.

Note that in the matrix form of the augmented Wiener–Hopf equations for backward prediction defined by Eq. (6.38), the correlation matrix of the tap inputs is equivalent to that in the corresponding equation (6.14). This is merely a restatement of the fact that the tap inputs in the backward prediction-error filter of Fig. 6.2(c) are exactly the same as those in the forward prediction-error filter of Fig. 6.1(b).

6.3 LEVINSON–DURBIN ALGORITHM

We now describe a direct method for computing the prediction-error filter coefficients and prediction-error power by solving the augmented Wiener–Hopf equations. The method is recursive in nature and makes particular use of the Toeplitz structure of the correlation matrix of the tap inputs of the filter. It is known as the *Levinson–Durbin algorithm*, so named in recognition of its use first by Levinson (1947) and then its independent reformulation at a later date by Durbin (1960). Basically, the procedure utilizes the solution of the augmented Wiener–Hopf equations for a prediction-error filter of order $m - 1$ to compute the corresponding solution for a prediction-error filter of order m (i.e., one order higher). The order $m = 1, 2, \ldots, M$, where M is the *final* order of the filter. The important virtue of the Levinson–Durbin algorithm is its computational efficiency, in that its use results in a big saving in the number of operations (multiplications or divisions) and storage locations compared to standard methods such as the Gauss elimination method (Makhoul, 1975). To derive the Levinson–Durbin recursive procedure, we will use the matrix formulation of both forward and backward predictions in an elegant way (Burg, 1968, 1975).

Let the $(m + 1)$-by-1 vector $\mathbf{a}_m$ denote the tap-weight vector of a forward prediction-error filter of order m. The $(m + 1)$-by-1 tap-weight vector of the corresponding backward prediction-error filter is obtained by backward rearrangement of the elements of vector $\mathbf{a}_m$ and their complex conjugation. We denote the combined effect of these two operations by $\mathbf{a}_m^{B*}$. Let the m-by-1 vectors $\mathbf{a}_{m-1}$ and $\mathbf{a}_{m-1}^{B*}$ denote the tap-weight vectors of the corresponding forward and backward prediction-error filters of order $m - 1$, respectively. The Levinson–Durbin recursion may be stated in one of two equivalent ways:

1. The tap-weight vector of a *forward* prediction-error filter may be order-updated as follows:

$$\mathbf{a}_m = \begin{bmatrix} \mathbf{a}_{m-1} \\ 0 \end{bmatrix} + \kappa_m \begin{bmatrix} 0 \\ \mathbf{a}_{m-1}^{B*} \end{bmatrix} \tag{6.40}$$

where κ_m is a constant. The scalar version of this *order update* is

$$a_{m,l} = a_{m-1,l} + \kappa_m a_{m-1,m-l}^*, \qquad l = 0, 1, \ldots, m \tag{6.41}$$

where $a_{m,l}$ is the lth tap weight of a backward prediction-error filter of order m, and likewise for $a_{m-1,l}^*$. The element $a_{m-1,m-l}^*$ is the lth tap weight of a backward

prediction-error filter of order $m - 1$. In Eq. (6.41), note that $a_{m-1,0} = 1$ and $a_{m-1,m} = 0$.

2. The tap-weight vector of a *backward* prediction-error filter may be order-updated as follows:

$$\mathbf{a}_m^{B*} = \begin{bmatrix} 0 \\ \mathbf{a}_{m-1}^{B*} \end{bmatrix} + \kappa_m^* \begin{bmatrix} \mathbf{a}_{m-1} \\ 0 \end{bmatrix} \tag{6.42}$$

The scalar version of this order update is

$$a_{m,m-l}^* = a_{m-1,m-l}^* + \kappa_m^* a_{m-1,l}, \qquad l = 0, 1, \ldots, m \tag{6.43}$$

where $a_{m,m-l}^*$ is the lth tap weight of the backward prediction-error filter of order m, and the other elements are as defined previously.

The Levinson–Durbin recursion is usually formulated in the context of forward prediction, in vector form as in Eq. (6.40) or scalar form as in Eq. (6.41). The formulation of the recursion in the context of backward prediction, in vector form as in Eq. (6.42) or scalar form as in Eq. (6.43), follows directly from that of Eq. (6.40) or (6.41), respectively, through a combination of backward rearrangement and complex conjugation (see Problem 8).

To establish the condition that the constant κ_m has to satisfy in order to justify the validity of the Levinson–Durbin algorithm, we proceed in four stages as follows:

1. We premultiply both sides of Eq. (6.40) by $\mathbf{R}_{m+1}$, the $(m + 1)$-by-$(m + 1)$ correlation matrix of the tap inputs $u(n), u(n - 1), \ldots, u(n - m)$ in the forward prediction-error filter of order m. For the left-hand side of Eq. (6.40), we thus get [see Eq. (6.14)]

$$\mathbf{R}_{m+1}\mathbf{a}_m = \begin{bmatrix} P_m \\ \mathbf{0}_m \end{bmatrix} \tag{6.44}$$

where P_m is the forward prediction-error power, and $\mathbf{0}_m$ is the m-by-1 null vector. The subscripts in the matrix $\mathbf{R}_{m+1}$ and the vector $\mathbf{0}_m$ refer to their dimensions, whereas the subscripts in the vector $\mathbf{a}_m$ and the scalar P_m refer to prediction order.

2. For the first term of the right-hand side of Eq. (6.40), we use the following partitioned form of the correlation matrix $\mathbf{R}_{m+1}$ (see Section 2.3).

$$\mathbf{R}_{m+1} = \begin{bmatrix} \mathbf{R}_m & \mathbf{r}_m^{B*} \\ \mathbf{r}_m^{BT} & r(0) \end{bmatrix}$$

where $\mathbf{R}_m$ is the m-by-m correlation matrix of the tap inputs $u(n)$, $u(n - 1)$, . . . , $u(n - m + 1)$, and $\mathbf{r}_m^{B*}$ is the cross-correlation vector between these tap inputs and $u(n - m)$. We may thus write

$$\begin{aligned} \mathbf{R}_{m+1} \begin{bmatrix} \mathbf{a}_{m-1} \\ 0 \end{bmatrix} &= \begin{bmatrix} \mathbf{R}_m & \mathbf{r}_m^{B*} \\ \mathbf{r}_m^{BT} & r(0) \end{bmatrix} \begin{bmatrix} \mathbf{a}_{m-1} \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{R}_m \mathbf{a}_{m-1} \\ \mathbf{r}_m^{BT} \mathbf{a}_{m-1} \end{bmatrix} \end{aligned} \tag{6.45}$$

The set of augmented Wiener–Hopf equations for the forward prediction-error filter of order $m - 1$ is

$$\mathbf{R}_m \mathbf{a}_{m-1} = \begin{bmatrix} P_{m-1} \\ \mathbf{0}_{m-1} \end{bmatrix} \tag{6.46}$$

where P_{m-1} is the prediction-error power for this filter, and $\mathbf{0}_{m-1}$ is the $(m-1)$-by-1 null vector. Define the scalar

$$\begin{aligned} \Delta_{m-1} &= \mathbf{r}_m^{BT} \mathbf{a}_{m-1} \\ &= \sum_{l=0}^{m-1} r(l-m) a_{m-1,l} \end{aligned} \tag{6.47}$$

Substituting Eqs. (6.46) and (6.47) in Eq. (6.45), we may therefore write

$$\mathbf{R}_{m+1} \begin{bmatrix} \mathbf{a}_{m-1} \\ 0 \end{bmatrix} = \begin{bmatrix} P_{m-1} \\ \mathbf{0}_{m-1} \\ \Delta_{m-1} \end{bmatrix} \tag{6.48}$$

3. For the second term on the right-hand side of Eq. (6.40), we use the following partitioned form of the correlation matrix $\mathbf{R}_{m+1}$ (see Section 2.3):

$$\mathbf{R}_{m+1} = \begin{bmatrix} r(0) & \mathbf{r}_m^H \\ \mathbf{r}_m & \mathbf{R}_m \end{bmatrix}$$

where $\mathbf{R}_m$ is the m-by-m correlation matrix of the tap inputs $u(n-1), u(n-2), \ldots, u(n-m)$, and $\mathbf{r}_m$ is the m-by-1 cross-correlation vector between these tap inputs and $u(n)$. We may thus write

$$\begin{aligned} \mathbf{R}_{m+1} \begin{bmatrix} 0 \\ \mathbf{a}_{m-1}^{B*} \end{bmatrix} &= \begin{bmatrix} r(0) & \mathbf{r}_m^H \\ \mathbf{r}_m & \mathbf{R}_m \end{bmatrix} \begin{bmatrix} 0 \\ \mathbf{a}_{m-1}^{B*} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{r}_m^H \mathbf{a}_{m-1}^{B*} \\ \mathbf{R}_m \mathbf{a}_{m-1}^{B*} \end{bmatrix} \end{aligned} \tag{6.49}$$

The scalar $\mathbf{r}_m^H \mathbf{a}_{m-1}^{B*}$ equals

$$\begin{aligned} \mathbf{r}_m^H \mathbf{a}_{m-1}^{B*} &= \sum_{k=1}^{m} r^*(-k) a_{m-1,m-k}^* \\ &= \sum_{l=0}^{m-1} r^*(l-m) a_{m-1,l}^* \\ &= \Delta_{m-1}^* \end{aligned} \tag{6.50}$$

Also, the set of augmented Wiener–Hopf equations for the backward prediction-error filter of order $m - 1$ is

$$\mathbf{R}_m \mathbf{a}_{m-1}^{B*} = \begin{bmatrix} \mathbf{0}_{m-1} \\ P_{m-1} \end{bmatrix} \tag{6.51}$$

Substituting Eqs. (6.50) and (6.51) in (6.49), we may therefore write

$$\mathbf{R}_{m+1} \begin{bmatrix} 0 \\ \mathbf{a}_{m-1}^{B*} \end{bmatrix} = \begin{bmatrix} \Delta_{m-1}^* \\ \mathbf{0}_{m-1} \\ P_{m-1} \end{bmatrix} \tag{6.52}$$

4. Summarizing the results obtained in stages 1, 2, and 3 and, in particular, using Eqs. (6.44), (6.48), and (6.52), we may now state that the premultiplication of both sides of Eq. (6.40) by the correlation matrix $\mathbf{R}_{m+1}$ yields

$$\begin{bmatrix} P_m \\ \mathbf{0}_m \end{bmatrix} = \begin{bmatrix} P_{m-1} \\ \mathbf{0}_{m-1} \\ \Delta_{m-1} \end{bmatrix} + \kappa_m \begin{bmatrix} \Delta_{m-1}^* \\ \mathbf{0}_{m-1} \\ P_{m-1} \end{bmatrix} \tag{6.53}$$

We conclude therefore that, if the order-update recursion of Eq. (6.40) holds, the results described by Eq. (6.53) are direct consequences of this recursion. Conversely, we may state that, if the conditions described by Eq. (6.53) apply, the tap-weight vector of a forward prediction-error filter may be order-updated as in Eq. (6.40).

From Eq. (6.53), we may make two important deductions:

1. By considering the first elements of the vectors on the left-hand and right-hand sides of Eq. (6.53), we have

$$P_m = P_{m-1} + \kappa_m \Delta_{m-1}^* \tag{6.54}$$

2. By considering the last elements of the vectors on the left-hand and right-hand sides of Eq. (6.53), we have

$$0 = \Delta_{m-1} + \kappa_m P_{m-1} \tag{6.55}$$

From Eq. (6.55), we see that the constant κ_m has the value

$$\kappa_m = -\frac{\Delta_{m-1}}{P_{m-1}} \tag{6.56}$$

where Δ_{m-1} is itself defined by Eq. (6.47). Furthermore, eliminating Δ_{m-1} between Eqs. (6.54) and (6.55), we get the following relation for the order update of the prediction-error power:

$$P_m = P_{m-1}(1 - |\kappa_m|^2) \tag{6.57}$$

As the order m of the prediction-error filter increases, the corresponding value of the prediction-error power P_m normally decreases or else remains the same. Of course, P_m can never be negative. Hence, we must always have

$$0 \leq P_m \leq P_{m-1}, \qquad m \geq 1 \tag{6.58}$$

For the elementary case of a prediction-error filter of order zero, we naturally have

$$P_0 = r(0)$$

where $r(0)$ is the autocorrelation function of the input process for zero lag.

Starting with $m = 0$, and increasing the filter order by 1 at a time, we find that through the repeated application of Eq. (6.57) the prediction-error power for a prediction-error filter of *final* order M equals

$$P_M = P_0 \prod_{m=1}^{M} (1 - |\kappa_m|^2) \tag{6.59}$$

Interpretations of the Parameters κ_m and Δ_{m-1}

The parameters κ_m, $1 \leq m \leq M$, resulting from the application of the Levinson–Durbin recursion to a prediction-error filter of final order M, are called *reflection coefficients*. The use of this term comes from the analogy of Eq. (6.57) with transmission line theory, where (in the latter context) κ_m may be considered as the reflection coefficient at the boundary between two sections with different characteristic impedances. Note that the condition on the reflection coefficient corresponding to that of Eq. (6.58) is

$$|\kappa_m| \leq 1 \qquad \text{for all } m \tag{6.60}$$

From Eq. (6.41), we see that for a prediction-error filter of order m, the reflection coefficient κ_m equals the *last* tap-weight $a_{m,m}$ of the filter. That is,

$$\kappa_m = a_{m,m}$$

As for the parameter Δ_{m-1}, it may be interpreted as a cross-correlation between the forward prediction error $f_{m-1}(n)$ and the delayed backward prediction error $b_{m-1}(n-1)$. Specifically, we may write (see Problem 9)

$$\Delta_{m-1} = E[b_{m-1}(n-1)f^*_{m-1}(n)] \tag{6.61}$$

where $f_{m-1}(n)$ is produced at the output of a forward prediction-error filter of order $m-1$ in response to the tap inputs $u(n), u(n-1), \ldots, u(n-m+1)$, and $b_{m-1}(n-1)$ is produced at the output of a backward prediction-error filter of order $m-1$ in response to the tap inputs $u(n-1), u(n-2), \ldots, u(n-m)$.

Note that

$$f_0(n) = b_0(n) = u(n)$$

where $u(n)$ is the prediction-error filter input at time n. Accordingly, from Eq. (6.61) we find that this cross-correlation parameter has the *zero-order value*

$$\begin{aligned}\Delta_0 &= E[b_0(n-1)f^*_0(n)] \\ &= E[u(n-1)u^*(n)] \\ &= r^*(1)\end{aligned}$$

where $r(1)$ is the autocorrelation function of the input for a lag of 1.

We may also use Eqs. (6.56) and (6.61) to develop a second interpretation for the parameter κ_m. In particular, since P_{m-1} may be viewed as the mean-square value of the forward prediction error $f_{m-1}(n)$, we may write

$$\kappa_m = \frac{E[b_{m-1}(n-1)f_{m-1}^*(n)]}{E[|f_{m-1}(n)|^2]} \tag{6.62}$$

The right-hand side of Eq. (6.62), except for the minus sign, is referred to as a *partial correlation (PARCOR) coefficient.* This terminology is widely used in the statistics literature (Box and Jenkins, 1976). Hence, the reflection coefficient, as defined here, is the negative of the PARCOR coefficient.

Application of the Levinson–Durbin Algorithm

There are two possible ways of applying the Levinson–Durbin algorithm to compute the prediction-error filter coefficients $a_{M,k}$, $k = 0, 1, \ldots, M$, and the prediction-error power P_M for a final prediction order M:

1. We have explicit knowledge of the autocorrelation function of the input process; in particular, we have $r(0), r(1), \ldots, r(M)$, denoting the values of the autocorrelation function for lags $0, 1, \ldots, M$, respectively. For example, we may compute *biased* estimates of these parameters by means of the *time-average formula*

$$\hat{r}(k) = \frac{1}{N}\sum_{n=1+k}^{N} u(n)u^*(n-k), \qquad k = 0, 1, \ldots, M \tag{6.63}$$

   where N is the total length of the input time series, with $N >> M$. There are, of course, other estimators that we may use.[2] In any event, given $r(0), r(1), \ldots, r(M)$, the computation proceeds by using Eq. (6.47) for Δ_{m-1} and Eq. (6.57) for P_m. The recursion is initiated with $m = 0$, for which we have $P_0 = r(0)$ and $\Delta_0 = r^*(1)$. Note also that $a_{m,0}$ equals 1 for all m, and $a_{m,k}$ is zero for all $k > m$. The computation is terminated when $m = M$. The resulting estimates of the prediction-error filter coefficients and prediction error power obtained by using this procedure are known as the *Yule–Walker estimates.*

2. We have explicit knowledge of the reflection coefficients $\kappa_1, \kappa_2, \ldots, \kappa_M$ and the autocorrelation function $r(0)$ for a lag of zero. Later in the chapter we describe a procedure for estimating these reflection coefficients directly from the given data. In this second application of the Levinson–Durbin recursion, we only need the pair of relations:

[2] In practice, the biased estimate of Eq. (6.63) is preferred over an unbiased estimate because it yields a much lower variance for the estimate $\hat{r}(k)$ for values of the lag k close to the data length N; for more details, see Box and Jenkins (1976). For a more refined estimate of the autocorrelation function $r(k)$, we may use the multiple-window method described in McWhorter and Scharf (1995). This method uses a multiplicity of special windows, resulting in the most general Hermitian, nonnegative-definite, and modulation-invariant estimate. The Hermitian and nonnegative-definite properties were described in Chapter 2. To define the modulation-invariant property, let $\hat{\mathbf{R}}$ denote an estimate of the correlation matrix, given the input vector $\mathbf{u}$. The estimate is said to be *modulation-invariant* if $\mathbf{D}(e^{j\Phi})\mathbf{u}$ has a correlation matrix equal to $\mathbf{D}(e^{j\Phi})\hat{\mathbf{R}}\mathbf{D}(e^{-j\Phi})$, where $\mathbf{D}(e^{j\Phi}) = \text{diag}(\psi(e^{j\Phi}))$ is a modulation matrix, and $\psi(e^{j\Phi}) = [1, e^{j\Phi}, \ldots, e^{j\Phi(M-1)}]^T$.

$$a_{m,k} = a_{m-1,k} + \kappa_m a^*_{m-1,m-k}, \qquad k = 0, 1, \ldots, m$$
$$P_m = P_{m-1}(1 - |\kappa_m|^2)$$

Here, again, the recursion is initiated with $m = 0$ and stopped when the order m reaches the final value M.

Example 2

To illustrate the second method for the application of the Levinson–Durbin recursion, suppose we are given the reflection coefficients κ_1, κ_2, κ_3 and average power P_0. The problem we wish to solve is to use these parameters to determine the corresponding tap weights $a_{3,1}$, $a_{3,2}$, $a_{3,3}$ and the prediction-error power P_3 for a prediction-error filter of order 3. The application of the Levinson–Durbin recursion, described by Eqs. (6.41) and (6.57), yields the following results for $m = 1, 2, 3$:

1. Prediction-error filter order $m = 1$:

$$a_{1,0} = 1$$
$$a_{1,1} = \kappa_1$$
$$P_1 = P_0(1 - |\kappa_1|^2)$$

2. Prediction-error filter order $m = 2$:

$$a_{2,0} = 1$$
$$a_{2,1} = \kappa_1 + \kappa_2 \kappa_1^*$$
$$a_{2,2} = \kappa_2$$
$$P_2 = P_1(1 - |\kappa_2|^2)$$

where P_1 is as defined above.

3. Prediction-error filter order $m = 3$:

$$a_{3,0} = 1$$
$$a_{3,1} = a_{2,1} + \kappa_3 \kappa_2^*$$
$$a_{3,2} = \kappa_2 + \kappa_3 a^*_{2,1}$$
$$a_{3,3} = \kappa_3$$
$$P_3 = P_2(1 - |\kappa_3|^2)$$

where $a_{2,1}$ and P_2 are as defined above.

The interesting point to observe from this example is that the Levinson–Durbin recursion yields not only the values of the tap weights and prediction-error power for the prediction-error filter of final order M, but also the corresponding values of these parameters for the prediction-error filters of intermediate orders $M - 1, \ldots, 1$.

Inverse Levinson–Durbin Algorithm

In the normal application of the Levinson–Durbin recursion, as illustrated in Example 2, we are given the set of reflection coefficients $\kappa_1, \kappa_2, \ldots \kappa_M$ and the requirement is to compute the corresponding set of tap weights $a_{M,1}, a_{M,2}, \ldots, a_{M,M}$ for a prediction-error filter of final order M. Of course, the remaining coefficient of the filter, $a_{M,0} = 1$. Frequently, however, the need arises to solve the following *inverse* problem: Given the set of tap weights $a_{M,1}, a_{M,2}, \ldots, a_{M,M}$, solve for the corresponding set of reflection coefficients κ_1, $\kappa_2, \ldots, \kappa_M$. We may solve this problem by applying the inverse form of Levinson–Durbin recursion, which we refer to simply as the *inverse recursion*.

To derive the inverse recursion, we first combine Eqs. (6.41) and (6.43), representing the scalar versions of the Levinson–Durbin recursion for forward and backward prediction-error filters, respectively, in matrix form as follows:

$$\begin{bmatrix} a_{m,k} \\ a^*_{m,m-k} \end{bmatrix} = \begin{bmatrix} 1 & \kappa_m \\ \kappa^*_m & 1 \end{bmatrix} \begin{bmatrix} a_{m-1,k} \\ a^*_{m-1,m-k} \end{bmatrix}, \qquad k = 0, 1, \ldots, m \tag{6.64}$$

where the order $m = 1, 2, \ldots, M$. Then, assuming that $|\kappa_m| < 1$ and solving Eq. (6.64) for the tap weight $a_{m-1,k}$, we get

$$a_{m-1,k} = \frac{a_{m,k} - a_{m,m}a^*_{m,m-k}}{1 - |a_{m,m}|^2}, \qquad k = 0, 1, \ldots, m \tag{6.65}$$

where we have used the fact that $\kappa_m = a_{m,m}$. We may now describe the procedure. Starting with the set of tap weights $\{a_{M,k}\}$ for which the prediction-error filter order equals M, we use the inverse recursion, Eq. (6.65), with decreasing filter order $m = M, M-1, \ldots, 2$ to compute the tap weights of the corresponding prediction-error filters of order $M-1$, $M-2, \ldots, 1$, respectively. Finally, knowing the tap weights of all the prediction-error filters of interest (whose order ranges all the way from M down to 1), we use the fact that

$$\kappa_m = a_{m,m}, \qquad m = M, M-1, \ldots, 1$$

to determine the desired set of reflection coefficients $\kappa_M, \kappa_{M-1}, \ldots, \kappa_1$. Example 3 illustrates the application of the inverse recursion.

Example 3

Suppose we are given the tap weights $a_{3,1}, a_{3,2}, a_{3,3}$ of a prediction-error filter of order 3, and the requirement is to determine the corresponding reflection coefficients $\kappa_1, \kappa_2, \kappa_3$. Application of the inverse recursion, described by Eq. (6.65), for filter order $m = 3, 2$ yields the following set of tap weights:

1. Prediction-error filter of order 2 [corresponding to $m = 3$ in Eq. (6.65)]:

$$a_{2,1} = \frac{a_{3,1} - a_{3,3}a^*_{3,2}}{1 - |a_{3,3}|^2}$$

$$a_{2,2} = \frac{a_{3,2} - a_{3,3}a^*_{3,1}}{1 - |a_{3,3}|^2}$$

2. Prediction-error filter of order 1 [corresponding to $m = 2$ in Eq. (6.65)]:

$$a_{1,1} = \frac{a_{2,1} - a_{2,2}a_{2,1}^*}{1 - |a_{2,2}|^2}$$

where $a_{2,1}$ and $a_{2,2}$ are as defined above. Thus, the required reflection coefficients are given by

$$\kappa_3 = a_{3,3}$$

$$\kappa_2 = a_{2,2}$$

$$\kappa_1 = a_{1,1}$$

where $a_{3,3}$ is given, and $a_{2,2}$ and $a_{1,1}$ are computed as shown above.

6.4 PROPERTIES OF PREDICTION-ERROR FILTERS

Property 1. ***Relations between the autocorrelation function and the reflection coefficients*** It is customary to represent the second-order statistics of a stationary time series in terms of its autocorrelation function or, equivalently, the power spectrum. The autocorrelation function and power spectrum form a discrete-time Fourier-transform pair (see Chapter 3). Another way of describing the second-order statistics of a stationary time series is to use the set of numbers $P_0, \kappa_1, \kappa_2, \ldots, \kappa_M$, where $P_0 = r(0)$ is the value of the autocorrelation function of the process for a lag of zero, and $\kappa_1, \kappa_2, \ldots, \kappa_M$ are the reflection coefficients for a prediction-error filter of final order M. This is a consequence of the fact that the set of numbers $P_0, \kappa_1, \kappa_2, \ldots, \kappa_M$ uniquely determines the corresponding set of autocorrelation function values $r(0), r(1), \ldots, r(M)$, and vice versa.

To prove this relationship, we first eliminate Δ_{m-1} between Eqs. (6.47) and (6.55), obtaining

$$\sum_{k=0}^{m-1} a_{m-1,k}\, r(k - m) = -\kappa_m P_{m-1} \tag{6.66}$$

Solving Eq. (6.66) for $r(m) = r^*(-m)$ and recognizing that $a_{m-1,0}$ equals 1, we get

$$r(m) = -\kappa_m^* P_{m-1} - \sum_{k=1}^{m-1} a_{m-1,k}^*\, r(m - k) \tag{6.67}$$

This is the desired recursive relation. If we are given the set of numbers $r(0), \kappa_1, \kappa_2, \ldots, \kappa_M$, then by using Eq. (6.67), together with the Levinson–Durbin recursive equations (6.41) and (6.57), we may recursively generate the corresponding set of numbers $r(0), r(1), \ldots, r(M)$.

For $|\kappa_m| \leq 1$, we find from Eq. (6.67) that the permissible region for $r(m)$, the value of the autocorrelation function of the input signal for a lag of m, is the interior (including circumference) of a circle of radius P_{m-1} and center at the complex-valued quantity

$$-\sum_{k=1}^{m-1} a_{m-1,k}^* r(m - k)$$

This is illustrated in Fig. 6.3.

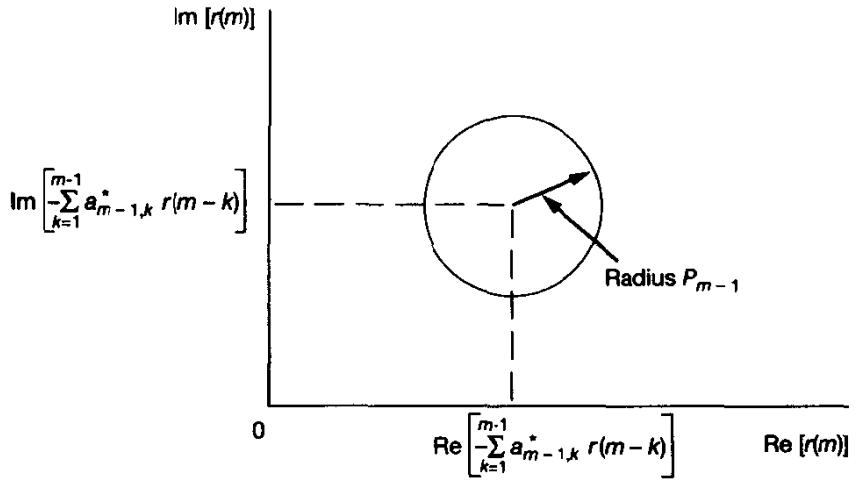


Figure 6.3 Permissible region for $r(m)$ for the case when $|\kappa_m| \leq 1$.

Suppose now that we are given the set of autocorrelation function values $r(1), \ldots, r(M)$. Then we may recursively generate the corresponding set of numbers $\kappa_1, \kappa_2, \ldots, \kappa_M$ by using the relation:

$$\kappa_m = -\frac{1}{P_{m-1}} \sum_{k=0}^{m-1} a_{m-1,k} r(k-m) \tag{6.68}$$

which is obtained by solving Eq. (6.66) for κ_m. In Eq. (6.68), it is assumed that P_{m-1} is nonzero. If P_{m-1} is zero, this would have been the result of $|\kappa_{m-1}| = 1$, and the sequence of reflection coefficients $\kappa_1, \kappa_2, \ldots, \kappa_{m-1}$ is terminated.

We may therefore make the statement: *There is a one-to-one correspondence between the two sets of quantities* $\{P_0, \kappa_1, \kappa_2, \ldots, \kappa_M\}$ *and* $\{r(0), r(1), r(2), \ldots, r(M)\}$, *in that if we are given the one we may uniquely determine the other in a recursive manner.*

Example 4

Suppose that we are given P_0, and κ_1, κ_2, κ_3 and the requirement is to compute $r(0)$, $r(1)$, $r(2)$, and $r(3)$. We start with $m = 1$, for which Eq. (6.67) yields

$$r(1) = -P_0 \kappa_1^*$$

where

$$P_0 = r(0)$$

For $m = 2$, the use of Eq. (6.67) yields

$$r(2) = -P_1 \kappa_2^* - r(1)\kappa_1^*$$

where

$$P_1 = P_0(1 - |\kappa_1|^2)$$

Finally, for $m = 3$, the use of Eq. (6.67) yields

$$r(3) = -P_2\kappa_3^* - [a_{2,1}^* r(2) + \kappa_2^* r(1)]$$

where

$$P_2 = P_1(1 - |\kappa_2|^2)$$
$$a_{2,1} = \kappa_1 + \kappa_2\kappa_1^*$$

Property 2. ***Transfer function of a forward prediction-error filter*** Let $H_{f,m}(z)$ denote the *transfer function* of a forward prediction-error filter of order m, and whose impulse response is defined by the sequence of numbers $a_{m,k}^*$, $k = 0, 1, \ldots, m$, as illustrated in Fig. 6.1(b) for $m = M$. From Chapter 1 we recall that the transfer function of a discrete-time filter equals the z-transform of its impulse response. We may therefore write

$$H_{f,m}(z) = \sum_{k=0}^{m} a_{m,k}^* z^{-k} \tag{6.69}$$

where z is a complex variable. Based on the Levinson–Durbin recursion, in particular Eq. (6.41), we may relate the coefficients of this filter of order m to those of a corresponding prediction-error filter of order $m - 1$ (i.e., one order smaller). In particular, substituting Eq. (6.41) into (6.69), we get

$$\begin{aligned} H_{f,m}(z) &= \sum_{k=0}^{m} a_{m-1,k}^* z^{-k} + \kappa_m^* \sum_{k=0}^{m} a_{m-1,m-k} z^{-k} \\ &= \sum_{k=0}^{m-1} a_{m-1,k}^* z^{-k} + \kappa_m^* z^{-1} \sum_{k=0}^{m-1} a_{m-1,m-1-k} z^{-k} \end{aligned} \tag{6.70}$$

where, in the second line, we have used the fact that $a_{m-1,m} = 0$. The sequence of numbers $a_{m-1,k}^*$, $k = 0, 1, \ldots, m - 1$, defines the impulse response of a forward prediction-error filter of order $m - 1$. Hence, we may write

$$H_{f,m-1}(z) = \sum_{k=0}^{m-1} a_{m-1,k}^* z^{-k} \tag{6.71}$$

The sequence of numbers $a_{m-1,m-1-k}$, $k = 0, 1, \ldots, m - 1$, defines the impulse response of a backward prediction-error filter of order $m - 1$; this is illustrated in Fig. 6.2(c) for the case of prediction order $m = M$. Hence, the second summation on the right-hand side of Eq. (6.70) represents the transfer function of this backward prediction-error filter. Let $H_{b,m-1}(z)$ denote this transfer function, as shown by

$$H_{b,m-1}(z) = \sum_{k=0}^{m-1} a_{m-1,m-1-k} z^{-k} \tag{6.72}$$

Hence, substituting Eqs. (6.71) and (6.72) in (6.70), we may write

$$H_{f,m}(z) = H_{f,m-1}(z) + \kappa_m^* z^{-1} H_{b,m-1}(z) \tag{6.73}$$

On the basis of the order update recursion of Eq. (6.73), we may now state: *Given the reflection coefficient κ_m and the transfer functions of the forward and backward predic-*

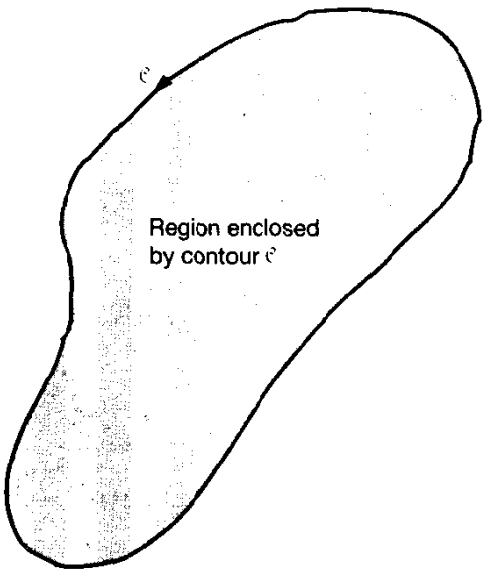


Figure 6.4 Contour $\mathscr{C}$ (traversed in the counterclockwise direction) in the z-plane and the region enclosed by it.

tion-error filters of order $m - 1$, the transfer function of the corresponding forward prediction-error filter of order m *is uniquely determined.*

Property 3. ***A forward prediction-error filter is minimum phase*** On the unit circle in the z-plane (i.e., for $|z| = 1$), we find that

$$|H_{f,m-1}(z)| = |H_{b,m-1}(z)|, \qquad |z| = 1$$

This is readily proved by substituting $z = \exp(j\omega)$, $-\pi < \omega \leq \pi$, in Eqs. (6.71) and (6.72). Suppose that the reflection coefficient κ_m satisfies the requirement $|\kappa_m| < 1$, for all m. Then we find that on the unit circle in the z-plane the magnitude of the second term in the right-hand side of Eq. (6.73) satisfies the conditions

$$|\kappa_m^* z^{-1} H_{b,m-1}(z)| < |H_{b,m-1}(z)| = |H_{f,m-1}(z)|, \qquad |z| = 1 \tag{6.74}$$

At this stage in our discussion, it is useful to recall *Rouché's theorem* from the theory of complex variables.[3] Rouché's theorem states:

If a function $F(z)$ is analytic upon a contour $\mathscr{C}$ in the z-plane and within the region enclosed by this contour, and if a second function $G(z)$, in addition to satisfying the same analyticity conditions, also fulfills the condition $|G(z)| < F(z)$ on the contour $\mathscr{C}$, then the function $F(z) + G(z)$ has the same number of zeros within the region enclosed by the contour $\mathscr{C}$ as does the function $F(z)$.

Ordinarily, the enclosed contour $\mathscr{C}$ is transversed in the *counterclockwise* direction, and the region enclosed by the contour lies to the *left* of it, as illustrated in Fig. 6.4. We

[3] For a review of complex variable theory, including Rouché's theorem, see Appendix A.

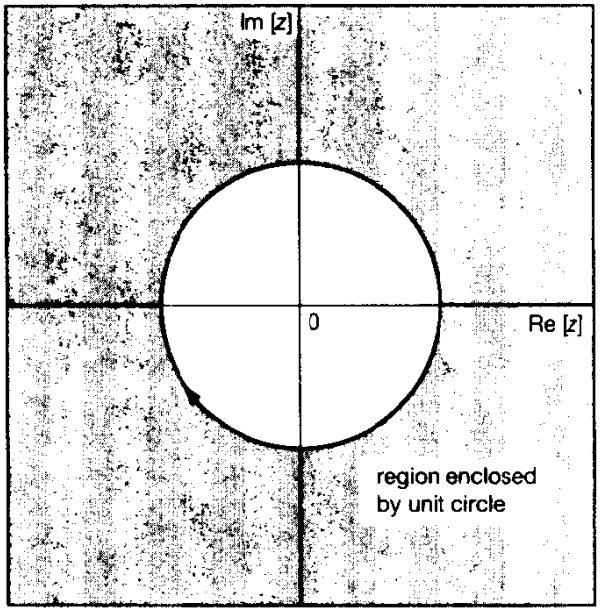


Figure 6.5 Unit circle (traversed in the clockwise direction) used as contour $\mathscr{C}$.

say that a function is analytic upon the contour $\mathscr{C}$ and within the region enclosed by it if the function has a continuous derivative everywhere upon the contour $\mathscr{C}$ and within the region enclosed by this contour. For this requirement to be satisfied, the function must have no poles upon the contour $\mathscr{C}$ or inside the region enclosed by the contour.

Let the contour $\mathscr{C}$ be the unit circle in the z-plane, which is traversed in the *clockwise* direction, as in Fig. 6.5. According to the convention just described, this assumption implies that the region enclosed by the contour $\mathscr{C}$ is represented by the entire part of the z-plane that lies *outside* the unit circle.

Let the functions $F(z)$ and $G(z)$ be identified with the two terms in the right-hand side of Eq. (6.73), as shown by

$$F(z) = H_{f,m-1}(z) \tag{6.75}$$

$$G(z) = \kappa_m^* z^{-1} H_{b,m-1}(z) \tag{6.76}$$

We observe that:

- The functions $F(z)$ and $G(z)$ have no poles inside the contour $\mathscr{C}$ defined in Fig. 6.5. Indeed, their derivatives are continuous throughout the region enclosed by this contour. Therefore, they are both analytic everywhere upon the unit circle and the region outside it.
- In view of Eq. (6.74), we have $|G(z)| < |F(z)|$ on the unit circle.

Accordingly, the functions $F(z)$ and $G(z)$ defined by Eqs. (6.75) and (6.76), respectively, satisfy all the conditions required by Rouché's theorem with respect to the contour $\mathscr{C}$ defined as the unit circle in Fig. 6.5.

Suppose that $H_{f,m-1}(z)$ and therefore $F(z)$ are known to have no zeros outside the unit circle in the z-plane. Then, by applying Rouché's theorem, we find that $F(z) + G(z)$, or, equivalently, $H_{f,m}(z)$ also has no zeros on or outside the unit circle in the z-plane.

In particular, for $m = 0$, the transfer function $H_{f,0}(z)$ is a constant equal to 1; therefore, it has no zeros at all. Using the result just derived, we may state that since $H_{f,0}(z)$ has no zeros outside the unit circle, then $H_{f,1}(z)$ will also have no zeros in this region of the z-plane, provided that $|\kappa_1| < 1$. Indeed, we can easily prove this latter result by noting that

$$\begin{aligned} H_{f,1}(z) &= a^{\dagger}_{1,0} + a^{\dagger}_{1,1}z^{-1} \\ &= 1 + \kappa^{\dagger}z^{-1} \end{aligned}$$

Hence, $H_{f,1}(z)$ has a single zero located at $z = -\kappa^{\dagger}$ and a pole at $z = 0$. With the reflection coefficient κ_1 constrained by the condition $|\kappa_1| < 1$, it follows that this zero must lie inside the unit circle. In other words, $H_{f,1}(z)$ has no zeros on or outside the unit circle. If $H_{f,1}(z)$ has no zeros on or outside the unit circle, then $H_{f,2}(z)$ will also have no zeros on or outside the unit circle provided that $|\kappa_2| < 1$, and so on.

We may thus state that the transfer function $H_{f,m}(z)$ of a forward prediction-error filter of order m has no zeros on or outside the unit circle in the z-plane for all values of m, if and only if the reflection coefficients satisfy the condition $|\kappa_m| < 1$ for all m. Such a filter is said to be minimum phase in the sense that, for a specified amplitude response, it has the minimum phase response possible for all values of z on the unit circle (see Chapter 1). Moreover, the amplitude response and phase response of the filter are uniquely related to each other. Based on these findings we may now make the statement: The transfer function $H_{f,m}(z)$ of a forward prediction-error filter of order m has no zeros on or outside the unit circle in the z-plane for all values of m, if and only if the reflection coefficients satisfy the condition $|\kappa_m| < 1$ for all m. In other words, *a forward prediction-error filter is minimum phase in the sense that, for a specified amplitude response, it has the minimum phase response possible for all values of z on the unit circle.*

Property 4. ***A backward prediction-error filter is maximum phase*** The transfer functions of backward and forward prediction-error filters of the same order are related, in that if we are given one we may uniquely determine the order. To find this relationship, we first evaluate $H_{f,m}^{*}(z)$, the complex conjugate of the transfer function of a forward prediction-error filter of order m, and so write [see Eq. 6.69)]

$$H_{f,m}^{*}(z) = \sum_{k=0}^{m} a_{m,k}(z^{*})^{-k} \tag{6.77}$$

Replacing z by the reciprocal of its complex conjugate z^{*}, we may rewrite Eq. (6.77) as

$$H_{f,m}^{*}\left(\frac{1}{z^{*}}\right) = \sum_{k=0}^{m} a_{m,k}z^{k}$$

Next, replacing k by $m - k$, we get

$$H_{f,m}^{*}\left(\frac{1}{z^{*}}\right) = z^{m}\sum_{k=0}^{m} a_{m,m-k}z^{-k} \tag{6.78}$$

The summation on the right-hand side of Eq. (6.78) constitutes the transfer function of a backward prediction-error filter of order m, as shown by

$$H_{b,m}(z) = \sum_{k=0}^{m} a_{m,m-k} z^{-k} \tag{6.79}$$

We thus find that $H_{b,m}(z)$ and $H_{f,m}(z)$ are related as follows:

$$H_{b,m}(z) = z^{-m} H_{f,m}^*\left(\frac{1}{z^*}\right) \tag{6.80}$$

where $H_{f,m}^*(1/z^*)$ is obtained by complex-conjugating $H_{f,m}(z)$, the transfer function of a forward prediction-error filter of order m, and replacing z by the reciprocal of z^*. Equation (6.80) states that multiplication of the new function obtained in this way by z^{-m} yields $H_{b,m}(z)$, the transfer function of the corresponding backward prediction-error filter.

Let the transfer function $H_{f,m}(z)$ be expressed in its factored form as follows:

$$H_{f,m}(z) = \prod_{i=1}^{m} (1 - z_i z^{-1}) \tag{6.81}$$

where z_i, $i = 1, 2, \ldots, m$, denote the zeros of the forward prediction-error filter. Hence, substituting Eq. (6.81) into (6.80), we may express the transfer function of the corresponding backward prediction-error filter in the factored form

$$\begin{aligned} H_{b,m}(z) &= z^{-m} \prod_{i=1}^{m} (1 - z_i^* z) \\ &= \prod_{i=1}^{m} (z^{-1} - z_i^*) \end{aligned} \tag{6.82}$$

The zeros of this transfer function are located at $1/z_i^*$, $i = 1, 2, \ldots m$. That is, the zeros of the backward and forward prediction-error filters are the *inverse* of each other with respect to the unit circle in the z-plane. The geometric nature of this relationship is illustrated for $m = 1$ in Fig. 6.6. The forward prediction-error filter has a zero at $z = -\kappa_1^*$, as in Fig. 6.6(a), and the backward prediction-error filter has a zero at $z = -1/\kappa_1$, as in Fig. 6.6(b). In this figure, it is assumed that the reflection coefficient κ_1 has a complex value. Consequently, a backward prediction-error filter has all of its zeros located outside the unit circle in the z-plane, because $|\kappa_m| < 1$ for all m.

We may therefore formally make the statement: *A backward prediction-error filter is maximum-phase in the sense that, for a specified amplitude response, it has the maximum phase responsible possible for all values of z on the unit circle.*

Property 5. ***A forward prediction-error filter is a whitening filter*** By definition, a *white-noise* process consists of a sequence of uncorrelated random variables. Thus, assuming that such a process, denoted by $v(n)$, has zero mean and variance σ_v^2, we may write (see Section 2.5)

$$E[v(k)v^*(n)] = \begin{cases} \sigma_v^2, & k = n \\ 0, & k \neq n \end{cases} \tag{6.83}$$

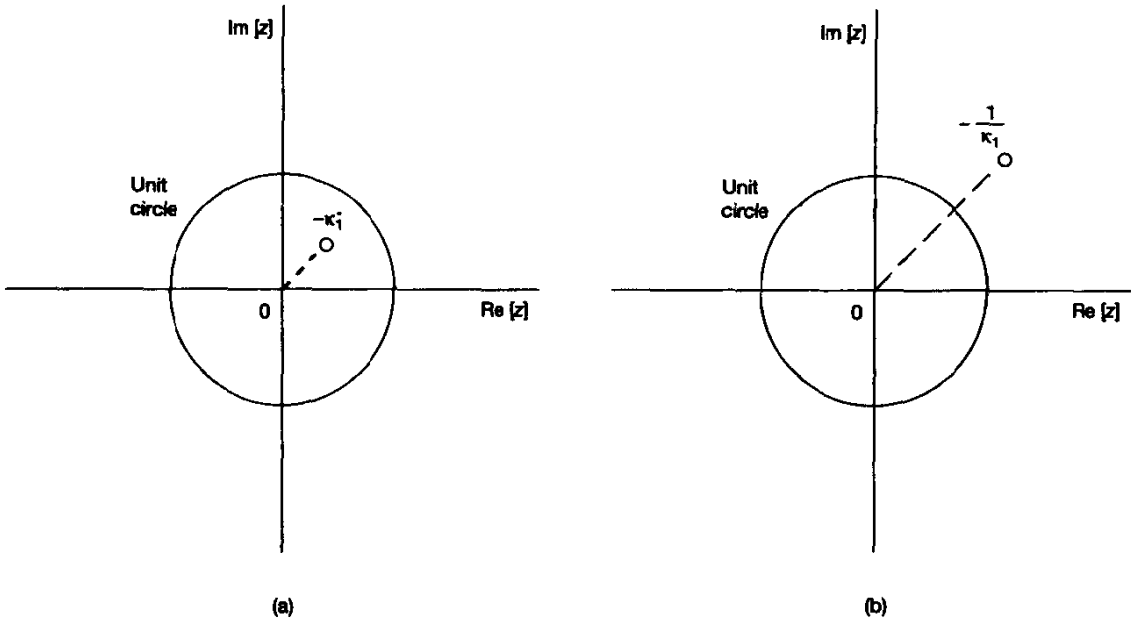


Figure 6.6 (a) Zero of forward prediction-error filter at $z = -\kappa_1^*$; (b) corresponding zero of backward prediction-error filter at $z = -1/\kappa_1$.

Accordingly, we say that white noise is purely unpredictable in the sense that the value of the process at time n is uncorrelated with all past values of the process up to and including time $n - 1$ (and, indeed, with all future values of the process too).

We may now state another important property of prediction-error filters: *A prediction-error filter is capable of whitening a stationary discrete-time stochastic process applied to its input, provided that the order of the filter is high enough.* Basically, prediction relies on the presence of correlation between adjacent samples of the input process. The implication of this is that, as we increase the order of the prediction-error filter, we successively reduce the correlation between adjacent samples of the input process, until ultimately we reach a point at which the filter has a high enough order to produce an output process that consists of a sequence of uncorrelated samples. The whitening of the original process applied to the filter input will have thereby been accomplished.

Property 6. ***Eigenvector representation of forward prediction-error filters*** The representation of a forward prediction-error filter is naturally related to eigenvalues (and associated eigenvectors) of the correlation matrix of the tap inputs in the filter. To develop such a representation, we first rewrite the augmented Wiener–Hopf equations (6.14), pertaining to a forward prediction-error filter of order M, in the compact matrix form

$$\mathbf{R}_{M+1}\mathbf{a}_M = P_M\mathbf{i}_{M+1} \tag{6.84}$$

where $\mathbf{R}_{M+1}$ is the $(M + 1)$-by-$(M + 1)$ correlation matrix of the tap inputs $u(n)$, $u(n - 1), \ldots, u(n - M)$ in the filter of Fig. 6.1(b), $\mathbf{a}_M$ is the $(M + 1)$-by-1 tap-weight vec-

tor of the filter, and the scalar P_M is the prediction error power. The $(M + 1)$-by-1 vector $\mathbf{i}_{M+1}$ is called the *first coordinate vector*; it has unity for its first element and zero for all the others. We illustrate this by writing

$$\mathbf{i}_{M+1} = [1, 0, \ldots, 0]^T \tag{6.85}$$

Solving Eq. (6.84) for $\mathbf{a}_M$, we get

$$\mathbf{a}_M = P_M \mathbf{R}_{M+1}^{-1} \mathbf{i}_{M+1} \tag{6.86}$$

where $\mathbf{R}_{M+1}^{-1}$ is the inverse of the correlation matrix $\mathbf{R}_{M+1}$; it is assumed that the matrix $\mathbf{R}_{M+1}$ is nonsingular, so that its inverse exists. Using the eigenvalue–eigenvector representation of a correlation matrix, which was discussed in Section 4.2, we may express the inverse matrix $\mathbf{R}_{M+1}^{-1}$ as follows:

$$\mathbf{R}_{M+1}^{-1} = \mathbf{Q}\mathbf{\Lambda}^{-1}\mathbf{Q}^H \tag{6.87}$$

where $\mathbf{\Lambda}$ is an $(M + 1)$-by-$(M + 1)$ diagonal matrix consisting of the eigenvalues of the correlation matrix $\mathbf{R}_{M+1}$, and $\mathbf{Q}$ is an $(M + 1)$-by-$(M + 1)$ matrix whose columns are the associated eigenvectors. That is,

$$\mathbf{\Lambda} = \text{diag}[\lambda_0, \lambda_1, \ldots, \lambda_M] \tag{6.88}$$

and

$$\mathbf{Q} = [\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_M] \tag{6.89}$$

where $\lambda_0, \lambda_1, \ldots, \lambda_M$ are the real-valued eigenvalues of the correlation matrix $\mathbf{R}_{M+1}$, and $\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_M$ are the respective eigenvectors. Thus, substituting Eqs. (6.87), (6.88), and (6.89) in Eq. (6.86), we get

$$\begin{aligned}
\mathbf{a}_M &= P_M \mathbf{Q}\mathbf{\Lambda}^{-1}\mathbf{Q}^H \mathbf{i}_{M+1} \\
&= P_M [\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_M] \text{diag}[\lambda_0^{-1}, \lambda_1^{-1}, \ldots, \lambda_M^{-1}] \begin{bmatrix} \mathbf{q}_0^H \\ \mathbf{q}_1^H \\ \cdot \\ \cdot \\ \cdot \\ \mathbf{q}_M^H \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \end{bmatrix} \\
&= P_M \sum_{k=0}^{M} \left(\frac{q_{k0}^*}{\lambda_k} \right) \mathbf{q}_k
\end{aligned} \tag{6.90}$$

where q_{k0} is the first element of the kth eigenvector of the correlation matrix $\mathbf{R}_{M+1}$. We note that the first element of the forward prediction-error filter vector $\mathbf{a}_M$ equals 1. Therefore, using this fact, we find from Eq. (6.90) that the prediction-error power equals

$$P_M = \frac{1}{\sum_{k=0}^{M} |q_{k0}|^2 \lambda_k^{-1}} \tag{6.91}$$

Thus, on the basis of Eqs. (6.90) and (6.91) we may make the statement: *The tap-weight vector of a forward prediction-error filter of order M and the resultant prediction-error power are uniquely defined by specifying the (M + 1) eigenvalues and the corresponding (M + 1) eigenvectors of the correlation matrix of the tap inputs of the filter.*

6.5 SCHUR–COHN TEST

The test described under Property 3 in Section 6.4 for the minimum-phase condition of a forward prediction-error of order M is relatively simple to apply if we know the associated set of reflection coefficients $\kappa_1, \kappa_2, \ldots, \kappa_M$. For the filter to be minimum phase [i.e., for all the zeros of the transfer function of the filter, $H_{f,m}(z)$, to lie inside the unit circle], we simply require that $|\kappa_m| < 1$ for all m. Suppose, however, that instead of these reflection coefficients we are given the tap weights of the filter, $a_{M,1}, a_{M,2}, \ldots, a_{M,M}$. In this case we first apply the inverse recursion [described by Eq. (6.65)] to compute the corresponding set of reflection coefficients $\kappa_1, \kappa_2, \ldots, \kappa_M$. Then, as before, we check whether or not $|\kappa_m| < 1$ for all m.

The method just described for determining whether or not $H_{f,m}(z)$ has zeros inside the unit circle, given the coefficients $a_{M,1}, a_{M,2}, \ldots, a_{M,M}$, is essentially the same as the *Schur–Cohn test.*[4]

To formulate the Schur–Cohn test, let

$$x(z) = a_{M,M} z^M + a_{M,M-1} z^{M-1} + \cdots + a_{M,0} \tag{6.92}$$

which is a polynomial in z, with $x(0) = a_{M,0} = 1$. Define

$$\begin{aligned} x'(z) &= z^M x^*(1/z^*) \\ &= a^*_{M,M} + a^*_{M,M-1} z + \cdots + a^*_{M,0} z^M \end{aligned} \tag{6.93}$$

which is the *reciprocal polynomial* associated with $x(z)$. The polynomial $x'(z)$ is so called since its zeros are the reciprocals of the zeros of $x(z)$. For $z = 0$, we have $x'(0) = a^*_{M,M}$. Next, define the linear combination

$$T[x(z)] = a^*_{M,0} x(z) - a_{M,M} x'(z) \tag{6.94}$$

so that, in particular, the value

$$\begin{aligned} T[x(0)] &= a^*_{M,0} x(0) - a_{M,M} x'(0) \\ &= 1 - |a_{M,M}|^2 \end{aligned} \tag{6.95}$$

[4] The classical Schur–Cohn test is discussed in Marden (1949) and Tretter (1976). The origin of the test can be traced back to Schur (1917) and Cohn (1922), hence the name. The test is also referred to as the Lehmer–Schur method (Ralston, 1965); this is in recognition of the application of Schur's theorem by Lehmer (1961).

is real, as it should be. Note also that $T[x(z)]$ has no term in z^M. Repeat this operation as far as possible, so that if we define

$$T^i[x,(z)] = T\{T^{i-1}[x(z)]\} \tag{6.96}$$

we generate a finite sequence of polynomials in z of *decreasing* order. The coefficient $a_{M,0}$ is equal to 1. Let it also be assumed that:

- The polynomial $x(z)$ has no zeros on the unit circle.
- The integer m is the smallest for which

$$T^m[x(z)] = 0, \text{ where } m \leq M + 1$$

Then, we may state the Schur–Cohn theorem as follows (Lehmer, 1961):

If for some i such that $1 \leq i \leq m$, we have $T^i[x(0)] < 0$, then $x(z)$ has at least one zero inside the unit circle. If, on the other hand, $T^i[x(0)] > 0$ for $1 \leq i < m$, and $T^{m-1}[x(z)]$ is a constant, then no zero of $x(z)$ lies inside the unit circle.

To apply this theorem to determine whether or not the polynomial $x(z)$ of Eq. (6.92), with $a_{M,0} \neq 0$, has a zero inside the unit circle, we proceed as follows (Ralston, 1965):

1. Calculate $T[x(z)]$. Is $T[x(0)]$ negative? If so, there is a zero inside the unit circle; if not, proceed to step 2.
2. Calculate $T^i[x(z)]$, $i = 1, 2, \ldots,$ until $T^i[x(0)] < 0$ for $i < m$, or $T^i[x(0)] > 0$ for $i < m$. If the former occurs, there is a zero inside the unit circle. If the latter occurs, and if $T^{m-1}[x(z)]$ is a constant, then there is no zero inside the unit circle.

Note that when the polynomial $x(z)$ has zeros inside the unit circle, this algorithm does not tell us how many; rather, it only confirms their existence.

The connection between the Schur–Cohn method and the inverse recursion is readily established by observing that (see Problem 10):

1. The polynomial $x(z)$ is related to the transfer function of a backward prediction-error filter of order M as follows:

$$x(z) = z^M H_{b,M}(z) \tag{6.97}$$

   Accordingly, if the Schur–Cohn test indicates that $x(z)$ has zero(s) inside the unit circle, we may conclude that the transfer function $H_{b,M}(z)$ is *not* maximum phase.
2. The reciprocal polynomial $x'(z)$ is related to the transfer function of the corresponding forward prediction-error filter of order M as follows:

$$x'(z) = z^M H_{f,M}(z) \tag{6.98}$$

Accordingly, if the Schur–Cohn test indicates that the original polynomial $x(z)$, with which $x'(z)$ is associated, has no zero(s) inside the unit circle, we may then conclude that the transfer function $H_{f,M}(z)$ is *not* minimum phase.

3. In general, we have

$$T^i[x(0)] = \prod_{j=0}^{i-1} (1 - |a_{M-j,M-j}|^2), \qquad 1 \le i \le M \tag{6.99}$$

and

$$H_{b,M-i}(z) = \frac{z^{i-M} T^i[x(z)]}{T^i[x(0)]} \tag{6.100}$$

where $H_{b,M-i}(z)$ is the transfer function of the backward prediction-error filter of order $M - i$.

6.6 AUTOREGRESSIVE MODELING OF A STATIONARY STOCHASTIC PROCESS

The whitening property of a forward prediction-error filter, operating on a stationary discrete-time stochastic process, is intimately related to the *autoregressive modeling* of the process. Indeed, we may view these two operations as *complementary*, as illustrated in Fig. 6.7. Part (a) of the figure depicts a forward prediction-error filter of order M, whereas part (b) depicts the corresponding autoregressive model. We may make the following observations:

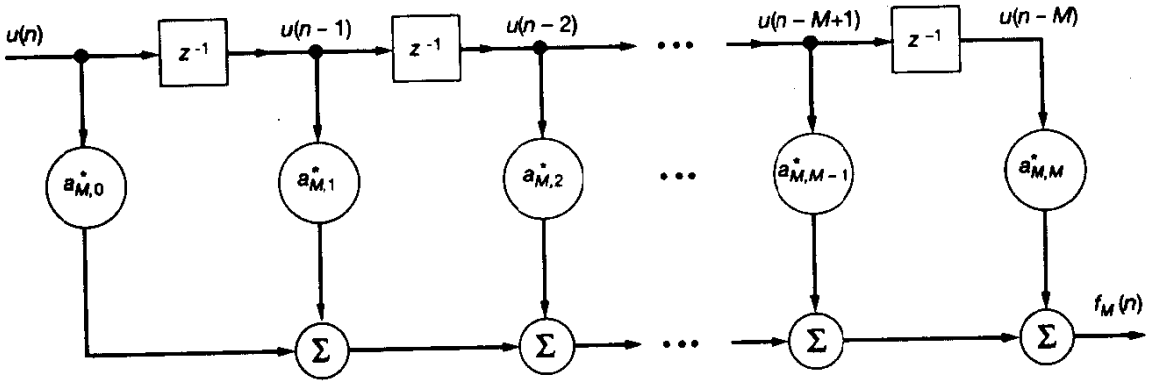
1. We may view the operation of prediction-error filtering applied to a stationary process $u(n)$ as one of *analysis*. In particular, we may use such an operation to whiten the process $u(n)$ by choosing the prediction-error filter order M sufficiently large, in which case the prediction error process $f_M(n)$ at the filter output consists of uncorrelated samples. When this unique condition has been established, the original stochastic process $u(n)$ is represented by the set of tap weights of the filter, $\{a_{M,k}\}$, and the prediction error power, P_M.
2. We may view the autoregressive (AR) modeling of the stationary process $u(n)$ as one of *synthesis*. In particular, we may generate the AR process $u(n)$ by applying a white-noise process $v(n)$ of zero mean and variance σ_v^2 to the input of an *inverse* filter whose parameters are set equal to the AR parameters w_{ok}, $k = 1, 2, \ldots, M$.

The two filter structures of Fig. 6.7 constitute a *matched pair*, with their parameters related as follows:

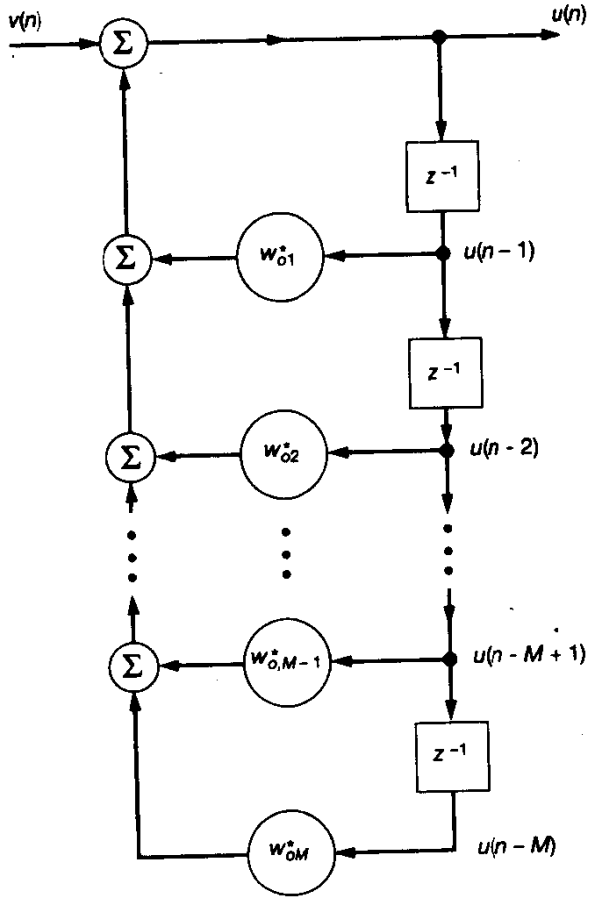
$$a_{M,k} = -w_{ok}, \qquad k = 1, 2, \ldots, M$$

and

$$P_M = \sigma_v^2$$



(a)



(b)

Figure 6.7 (a) Prediction-error (all-zero) filter; (b) autoregressive (all-pole) model with $w_{ok} = -a_{M,K}$, where $k = 1, 2, \ldots, M$; the input $v(n)$ is white noise.

The prediction-error filter of Fig. 6.7(a) is an *all-zero filter with an impulse response of finite duration*. On the other hand, the inverse filter in the AR model of Fig. 6.7(b) is an *all-pole filter with an impulse response of infinite duration*. The prediction-error filter is minimum phase, with the zeros of its transfer function located at exactly the same positions (inside the units circle in the z-plane) as the poles of the transfer function of the inverse filter in part (b) of the figure. This assures the stability of the inverse filter in the bounded input-bounded output sense or, equivalently, the asymptotic stationarity of the AR process generated at the output of this filter.

Equivalence of the Autoregressive and Maximum Entropy Power Spectra

Consider the AR model of Fig. 6.7(b). The process $v(n)$ applied to the model input consists of a white noise of zero mean and variance σ_v^2. To find the power spectral density of the AR process $u(n)$ produced at the model output, we multiply the power spectral density of the model input $v(n)$ by the squared amplitude response of the model (see Chapter 2). Let $S_{AR}(\omega)$ denote the power spectral density of the AR process $u(n)$. We may therefore write (Priestley, 1981)

$$S_{AR}(\omega) = \frac{\sigma_v^2}{\left|1 - \sum_{k=1}^{M} w_{ok}^* e^{-j\omega k}\right|^2} \tag{6.101}$$

The formula of Eq. (6.101) is called the *autoregressive power spectrum* or simply the *AR spectrum*.

A power spectrum of closely related interest is obtained using the *maximum entropy method (MEM)*. Suppose that we are given $2M + 1$ values of the autocorrelation function of a wide-sense stationary process $u(n)$. The essence of the maximum entropy method is to determine the particular power spectrum of the process that corresponds to the most random time series whose autocorrelation function is consistent with the set of $2M + 1$ known values (Burg, 1968, 1975). The result so obtained is referred to as the *maximum entropy power spectrum* or simply the *MEM spectrum*; see Appendix E. Let $S_{MEM}(\omega)$ denote the MEM spectrum. The determination of $S_{MEM}(\omega)$ is linked with the characterization of a prediction-error filter of order M, as shown in Appendix E. There it is shown that

$$S_{MEM}(\omega) = \frac{P_M}{\left|1 + \sum_{k=1}^{M} a_{M,k}^* e^{-jk\omega}\right|^2} \tag{6.102}$$

where the $a_{M,k}$ are the prediction-error filter coefficients, and P_M is the prediction-error power, all of which correspond to a prediction order M.

In view of the one-to-one correspondence that exists between the prediction-error filter of Fig. 6.7(a) and the AR model of Fig. 6.7(b), we have

$$a_{M,k} = -w_{ok}, \qquad k = 1, 2, \ldots, M \tag{6.103}$$

and

$$P_M = \sigma_v^2 \tag{6.104}$$

Accordingly, the formulas given in Eqs. (6.101) and (6.102) are one and the same. In other words, for the case of a wide-sense stationary process, the AR spectrum (for model order M) and the MEM spectrum (for prediction order M) are indeed *equivalent* (Van den Bos, 1971).

6.7 CHOLESKY FACTORIZATION

Consider a stack of backward prediction-error filters of orders 0 to M that are connected in parallel as in Fig. 6.8. The filters are all fed at time n, with the same input denoted by $u(n)$. Note that for the case of zero prediction order, we simply have a direct connection as shown at the top end of Fig. 6.8. Let $b_0(n), b_1(n), \ldots, b_M(n)$ denote the sequence of backward prediction errors produced by these filters. We may express these errors in terms of the respective filter inputs and filter coefficients as follows [see Fig. 6.2(c)]:

$$\begin{aligned}
b_0(n) &= u(n) \\
b_1(n) &= a_{1,1}u(n) + a_{1,0}u(n-1) \\
b_2(n) &= a_{2,2}u(n) + a_{2,1}u(n-1) + a_{2,0}u(n-2) \\
&\vdots \\
b_M(n) &= a_{M,M}u(n) + a_{M,M-1}u(n-1) + \ldots + a_{M,0}u(n-M)
\end{aligned}$$

Combining this system of $(M+1)$ simultaneous linear equations into a compact matrix form, we have

$$\mathbf{b}(n) = \mathbf{L}\mathbf{u}(n) \tag{6.105}$$

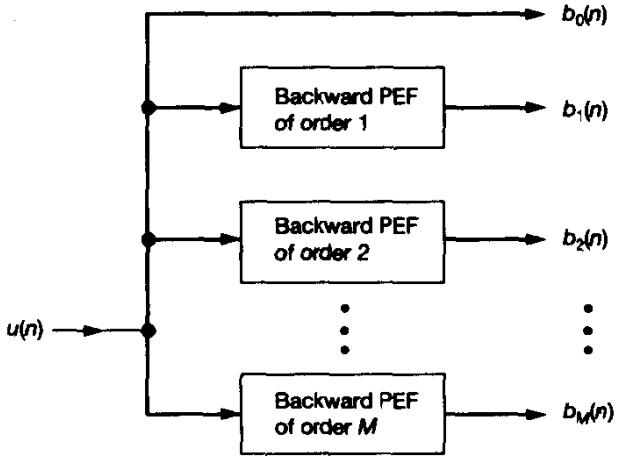


Figure 6.8 Parallel connection of a stack of backward prediction-error filters of orders 0 to M. *Note*: The direct connection represents a PEF with a single coefficient equal to unity.

where $\mathbf{u}(n)$ is the $(M + 1)$-by-1 input vector:

$$\mathbf{u}(n) = [u(n), u(n-1), \ldots, u(n-M)]^T$$

and $\mathbf{b}(n)$ is the $(M + 1)$-by-1 output vector of backward prediction errors:

$$\mathbf{b}(n) = [b_0(n), b_1(n), \ldots, b_M(n)]^T$$

The $(M + 1)$-by-$(M + 1)$ coefficient matrix on the right-hand side of Eq. (6.105) is defined in terms of the backward prediction-error filter coefficients of orders 0 to M as follows

$$\mathbf{L} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ a_{1,1} & 1 & \cdots & 0 \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ a_{M,M} & a_{M,M-1} & \cdots & 1 \end{bmatrix} \tag{6.106}$$

The matrix $\mathbf{L}$ has three useful properties:

1. The matrix $\mathbf{L}$ is a *lower triangular matrix* with 1's along its main diagonal; all of its elements above the main diagonal are zero.
2. The determinant of matrix $\mathbf{L}$ is unity; hence, it is nonsingular (i.e., its inverse exists).
3. The nonzero elements of each row of the matrix $\mathbf{L}$, except for complex conjugation, equal the weights of a backward prediction-error filter whose order corresponds to the position of that row in the matrix.

The transformation of Eq. (6.105) is known as the *Gram–Schmidt orthogonalization algorithm*.[5] According to this algorithm, there is a *one-to-one correspondence between* the input vector $\mathbf{u}(n)$ and the backward prediction-error vector $\mathbf{b}(n)$. In particular, given $\mathbf{u}(n)$ we may obtain $\mathbf{b}(n)$ by using Eq. (6.105). Conversely, given $\mathbf{b}(n)$, we may obtain the corresponding vector $\mathbf{u}(n)$ by using the inverse of Eq. (6.105), as shown by

$$\mathbf{u}(n) = \mathbf{L}^{-1}\mathbf{b}(n) \tag{6.107}$$

where $\mathbf{L}^{-1}$ is the inverse of the matrix $\mathbf{L}$.

Orthogonality of Backward Prediction Errors

The backward prediction errors $b_0(n), b_1(n), \ldots, b_M(n)$, constituting the elements of the vector $\mathbf{b}(n)$, have an important property: *They are all orthogonal to each other, as shown by*

$$E[b_m(n)b_i^*(n)] = \begin{cases} P_m, & i = m \\ 0, & i \neq m \end{cases} \tag{6.108}$$

[5] For a full discussion of the Gram–Schmidt algorithm and the various methods for its implementation, see Haykin (1989).

To prove this property, we may proceed as follows. First of all, without loss of generality, we may assume that $m \geq i$. To prove Eq. (6.108), we express the backward prediction error $b_i(n)$ in terms of the input $u(n)$ as the linear convolution sum

$$b_i(n) = \sum_{k=0}^{i} a_{i,i-k} u(n-k) \tag{6.109}$$

which is a rewrite of Eq. (6.37) with prediction order i used in place of M. Hence, using this relation to evaluate the expectation of $b_m(n)b_i^*(n)$, we get

$$E[b_m(n)b_i^*(n)] = E[b_m(n) \sum_{k=0}^{i} a_{i,i-k}^* u^*(n-k)] \tag{6.110}$$

From the principle of orthogonality, we have

$$E[b_m(n)u^*(n-k)] = 0, \quad 0 \leq k \leq m-1 \tag{6.111}$$

For $m > i$ and $0 \leq k \leq i$, we therefore find that all the expectation terms inside the summation on the right-hand side of Eq. (6.110) are zero. Correspondingly,

$$E[b_m(n)b_i^*(n)] = 0, \qquad m \neq i$$

When $m = i$, Eq. (6.110) reduces to

$$\begin{aligned} E[b_m(n)b_i^*(n)] &= E[b_m(n)b_m^*(n)] \\ &= P_m, \qquad m = i \end{aligned}$$

This completes the proof of Eq. (6.108). It is important, however, to note that *this property holds only for wide-sense stationary input data.*

We thus see that the Gram–Schmidt orthogonalization algorithm of Eq. (6.105) transforms the input vector $\mathbf{u}(n)$ consisting of correlated samples into an equivalent vector $\mathbf{b}(n)$ of uncorrelated backward prediction errors.[6]

Factorization of the Inverse of Correlation Matrix R

Having equipped ourselves with the property that backward prediction errors are indeed orthogonal to each other, we may return to the transformation described by the Gram–Schmidt algorithm in Eq. (6.105). Specifically, using this transformation, we may

[6] Two random variables X and Y are said to be *orthogonal* to each other if

$$E[XY^*] = 0$$

They are said to be *uncorrelated* with each other if

$$E[X - E[X])(Y - E[Y])^*] = 0$$

If one or both of X and Y have zero mean, then these two conditions become one and the same. For the discussion presented herein, the input data, and therefore the backward predicted errors, are assumed to have zero mean. Under this assumption, we may interchange orthogonality with uncorrelatedness when we refer to backward prediction errors.

express the correlation matrix of the backward prediction-error vector $\mathbf{b}(n)$ in terms of the correlation matrix of the input vector $\mathbf{u}(n)$ as follows:

$$\begin{aligned} E[\mathbf{b}(n)\mathbf{b}^H(n)] &= E[\mathbf{L}\mathbf{u}(n)\mathbf{u}^H(n)\mathbf{L}^H] \\ &= \mathbf{L}E[\mathbf{u}(n)\mathbf{u}^H(n)]\mathbf{L}^H \end{aligned} \tag{6.112}$$

Let $\mathbf{D}$ denote the correlation matrix of the backward prediction-error vector $\mathbf{b}(n)$, as shown by

$$\mathbf{D} = E[\mathbf{b}(n)\mathbf{b}^H(n)] \tag{6.113}$$

As before, the correlation matrix of the input vector $\mathbf{u}(n)$ is denoted by $\mathbf{R}$. We may therefore rewrite Eq. (6.112) as

$$\mathbf{D} = \mathbf{L}\mathbf{R}\mathbf{L}^H \tag{6.114}$$

We now make two observations:

1. When the correlation matrix $\mathbf{R}$ of the input vector $\mathbf{u}(n)$ is positive definite and therefore its inverse exists, the correlation matrix $\mathbf{D}$ of the backward prediction-error vector $\mathbf{b}(n)$ is also positive definite and likewise its inverse exists.
2. The correlation matrix $\mathbf{D}$ is a diagonal matrix, because $\mathbf{b}(n)$ consists of elements that are all orthogonal to each other. In particular, we may express the correlation matrix $\mathbf{D}$ in the expanded form:

$$\mathbf{D} = \text{diag}(P_0, P_1, \ldots, P_M) \tag{6.115}$$

where P_i is the average power of the ith backward prediction error $b_i(n)$; that is,

$$P_i = E[|b_i(n)|^2], \qquad i = 0, 1, \ldots, M \tag{6.116}$$

The inverse of matrix $\mathbf{D}$ is also a diagonal matrix, as shown by

$$\mathbf{D}^{-1} = \text{diag}(P_0^{-1}, P_1^{-1}, \ldots, P_M^{-1}) \tag{6.117}$$

Accordingly, we may use Eq. (6.114) to express the inverse of the correlation matrix $\mathbf{R}$ as follows:

$$\begin{aligned} \mathbf{R}^{-1} &= \mathbf{L}^H\mathbf{D}^{-1}\mathbf{L} \\ &= (\mathbf{D}^{-1/2}\mathbf{L})^H(\mathbf{D}^{-1/2}\mathbf{L}) \end{aligned} \tag{6.118}$$

which is the desired result.

The inverse matrix $\mathbf{D}^{-1}$, in the first line of Eq. (6.118), is a diagonal matrix defined by Eq. (6.117). The matrix $\mathbf{D}^{-1/2}$, the *square root* of $\mathbf{D}^{-1}$, in the second line of Eq. (6.118) is also a diagonal matrix defined by

$$\mathbf{D}^{-1/2} = \text{diag}(P_0^{-1/2}, P_1^{-1/2}, \ldots, P_M^{-1/2})$$

The transformation of Eq. (6.118) is called the *Cholesky factorization of the inverse matrix* $\mathbf{R}^{-1}$ (Stewart, 1973). Note that the matrix $\mathbf{D}^{-1/2}\mathbf{L}$ is a lower triangular matrix; however, it

differs from the lower triangular matrix $\mathbf{L}$ of Eq. (6.106) in that its diagonal elements are different from 1. Note also that the Hermitian-transposed matrix product $(\mathbf{D}^{-1/2}\mathbf{L})^H$ is an upper triangular matrix whose diagonal elements are different from 1. Thus, *according to the Cholesky factorization, the inverse correlation matrix* $\mathbf{R}^{-1}$ *may be factored into the product of an upper triangular matrix and a lower triangular matrix that are the Hermitian transpose of each other.*

6.8 LATTICE PREDICTORS

To implement the Gram–Schmidt algorithm of Eq. (6.105) for transforming an input vector $\mathbf{u}(n)$ consisting of correlated samples into an equivalent vector $\mathbf{b}(n)$ consisting of uncorrelated backward prediction errors, we may use the parallel connection of a direct path and an appropriate number of backward prediction-error filters, as illustrated in Fig. 6.8. The vectors $\mathbf{b}(n)$ and $\mathbf{u}(n)$ are said to be "equivalent" in the sense that they contain the same amount of information (see Problem 21). A much more efficient method of implementing the Gram–Schmidt orthogonalization algorithm, however, is to use an order-recursive structure in the form of a ladder, known as a *lattice predictor*. This device combines several forward and backward prediction-error filtering operations into a single structure. Specifically, a lattice predictor consists of a cascade connection of elementary units (stages), all of which have a structure similar to that of a lattice, hence the name. The number of stages in a lattice predictor equals the prediction order. Thus, for a prediction-error filter of order m, there are m stages in the lattice realization of the filter.

Order-update Recursions for the Prediction Errors

The input–output relations that characterize a lattice predictor may be derived in various ways, depending on the particular form in which the Levinson–Durbin algorithm is utilized. For the derivation presented here, we start with the matrix formulations of this algorithm given by Eqs. (6.40) and (6.42) that pertain to the forward and backward operations of a prediction-error filter, respectively. For convenience of presentation, we reproduce these two relations here:

$$\mathbf{a}_m = \begin{bmatrix} \mathbf{a}_{m-1} \\ 0 \end{bmatrix} + \kappa_m \begin{bmatrix} 0 \\ \mathbf{a}_{m-1}^{B*} \end{bmatrix} \tag{6.119}$$

$$\mathbf{a}_m^{B*} = \begin{bmatrix} 0 \\ \mathbf{a}_{m-1}^{B*} \end{bmatrix} + \kappa_m^* \begin{bmatrix} \mathbf{a}_{m-1} \\ 0 \end{bmatrix} \tag{6.120}$$

The $(m + 1)$-by-1 vector $\mathbf{a}_m$ and the m-by-1 vector $\mathbf{a}_{m-1}$ refer to forward prediction-error filters of order m and $m - 1$, respectively. The $(m + 1)$-by-1 vector $\mathbf{a}_m^{B*}$ and the m-by-1 vector $\mathbf{a}_{m-1}^{B*}$ refer to the corresponding backward prediction-error filters of order m and $m - 1$, respectively. The scalar κ_m is the associated reflection coefficient.

Consider first the forward prediction-error filter of order m, with its tap inputs denoted by $u(n), u(n-1), \ldots, u(n-m)$. We may partition $\mathbf{u}_{m+1}(n)$, the $(m+1)$-by-1 tap-input vector of this filter, in the form

$$\mathbf{u}_{m+1}(n) = \begin{bmatrix} \mathbf{u}_m(n) \\ \hline u(n-m) \end{bmatrix} \tag{6.121}$$

or in the equivalent form

$$\mathbf{u}_{m+1}(n) = \begin{bmatrix} u(n) \\ \hline \mathbf{u}_m(n-1) \end{bmatrix} \tag{6.122}$$

Next, we form the inner product of the $(m+1)$-by-1 vectors $\mathbf{a}_m$ and $\mathbf{u}_{m+1}(n)$. This is done by premultiplying $\mathbf{u}_{m+1}(n)$ by the Hermitian transpose of $\mathbf{a}_m$. Thus, using Eq. (6.119) for $\mathbf{a}_m$, we may treat the terms resulting from this multiplication as follows:

1. For the left-hand side of Eq. (6.119), premultiplication of $\mathbf{u}_{m+1}(n)$ by $\mathbf{a}_m^H$ yields

$$f_m(n) = \mathbf{a}_m^H \mathbf{u}_{m+1}(n) \tag{6.123}$$

where $f_m(n)$ is the forward prediction error produced at the output of the forward prediction-error filter of order m.

2. For the first term on the right-hand side of Eq. (6.119), we use the partitioned form of $\mathbf{u}_{m+1}(n)$ given in Eq. (6.121). We may therefore write

$$\begin{aligned} [\mathbf{a}_{m-1}^H \mid 0]\mathbf{u}_{m+1}(n) &= [\mathbf{a}_{m-1}^H \mid 0] \begin{bmatrix} \mathbf{u}_m(n) \\ \hline u(n-m) \end{bmatrix} \\ &= \mathbf{a}_{m-1}^H \mathbf{u}_m(n) \\ &= f_{m-1}(n) \end{aligned} \tag{6.124}$$

where $f_{m-1}(n)$ is the forward prediction error produced at the output of the forward prediction-error filter of order $m-1$.

3. For the second matrix term on the right-hand side of Eq. (6.119), we use the partitioned form of $\mathbf{u}_{m+1}(n)$ given in Eq. (6.122). We may therefore write

$$\begin{aligned} [0 \mid \mathbf{a}_{m-1}^{BT}]\, \mathbf{u}_{m+1}(n) &= [0 \mid \mathbf{a}_{m-1}^{BT}] \begin{bmatrix} u(n) \\ \hline \mathbf{u}_m(n-1) \end{bmatrix} \\ &= \mathbf{a}_{m-1}^{BT} \mathbf{u}_m(n-1) \\ &= b_{m-1}(n-1) \end{aligned} \tag{6.125}$$

where $b_{m-1}(n-1)$ is the *delayed* backward prediction error produced at the output of the backward prediction-error filter of order $m-1$.

Combining the results of the multiplications, described by Eqs. (6.123), (6.124), and (6.125), we may thus write

$$f_m(n) = f_{m-1}(n) + \kappa_m^* b_{m-1}(n-1) \tag{6.126}$$

Consider next the backward prediction-error filter of order m, with its tap inputs denoted by $u(n), u(n-1), \ldots, u(n-m)$. Here gain we may express $\mathbf{u}_{m+1}(n)$, the $(m+1)$-by-1 tap-input vector of this filter, in the partitioned form of Eq. (6.121) or that of Eq. (6.122). In this case, the terms resulting from the formation of the inner product of the vectors $\mathbf{a}_m^{B*}$ and $\mathbf{u}_{m+1}(n)$ are treated as follows:

1. For the left-hand side of Eq. (6.120), premultiplication of $\mathbf{u}_{m+1}(n)$ by the Hermitian transpose of $\mathbf{a}_m^{B*}$ yields

$$b_m(n) = \mathbf{a}_m^{BT}\,\mathbf{u}_{m+1}(n) \tag{6.127}$$

where $b_m(n)$ is the backward prediction error produced at the output of the backward prediction-error filter order m.

2. For the first term on the right-hand side of Eq. (6.120), we use the partitioned form of the tap-input vector $\mathbf{u}_{m+1}(n)$ given in Eq. (6.122). Thus, multiplying the Hermitian transpose of this first term by $\mathbf{u}_{m+1}(n)$, we get

$$\begin{aligned}[0 \mid \mathbf{a}_{m-1}^{BT}]\mathbf{u}_{m+1}(n) &= [0 \mid \mathbf{a}_{m-1}^{BT}]\begin{bmatrix} u(n) \\ \hline \mathbf{u}_m(n-1) \end{bmatrix} \\ &= \mathbf{a}_{m-1}^{BT}\mathbf{u}_m(n-1) \\ &= b_{m-1}(n-1)\end{aligned} \tag{6.128}$$

3. For the second matrix term on the right-hand side of Eq. (6.120), we use the partitioned form of the tap-input vector $\mathbf{u}_{m+1}(n)$ given in Eq. (6.121). Thus, multiplying the Hermitian transpose of this second term by $\mathbf{u}_{m+1}(n)$, we get

$$\begin{aligned}[\mathbf{a}_{m-1}^{H} \mid 0]\mathbf{u}_{m+1}(n) &= [\mathbf{a}_{m-1}^{H} \mid 0]\begin{bmatrix} u(n) \\ \hline \mathbf{u}_m(n-m) \end{bmatrix} \\ &= \mathbf{a}_{m-1}^{H}\mathbf{u}_m(n) \\ &= f_{m-1}(n)\end{aligned} \tag{6.129}$$

Combining the results of Eqs. (6.127), (6.128), and (6.129), we thus find that the inner product of $\mathbf{a}_m^{B*}$ and $\mathbf{u}_{m+1}(n)$ yields

$$b_m(n) = b_{m-1}(n-1) + \kappa_m f_{m-1}(n) \tag{6.130}$$

Equations (6.126) and (6.130) are the sought-after pair of *order-update recursions* that characterize stage m of the lattice predictor. They are reproduced here in matrix form as follows:

$$\begin{bmatrix} f_m(n) \\ b_m(n) \end{bmatrix} = \begin{bmatrix} 1 & \kappa_m^* \\ \kappa_m & 1 \end{bmatrix}\begin{bmatrix} f_{m-1}(n) \\ b_{m-1}(n-1) \end{bmatrix}, \qquad m = 1, 2, \ldots, M \tag{6.131}$$

We may view $b_{m-1}(n-1)$ as the result of applying the unit-delay operator z^{-1} to the backward prediction error $b_{m-1}(n)$; that is,

$$b_{m-1}(n-1) = z^{-1}[b_{m-1}(n)] \tag{6.132}$$

Thus, using Eqs. (6.131) and (6.132), we may represent stage m of the lattice predictor by the signal-flow graph shown in Fig. 6.9(a). Except for the branch pertaining to the block labeled z^{-1}, this signal-flow graph has the appearance of a lattice, hence the name "lattice predictor."[7] Note also that the parameterization of stage m of the lattice predictor is uniquely defined by the reflection coefficient κ_m.

For the elementary case of $m = 0$, we get the *initial conditions*:

$$f_0(n) = b_0(n) = u(n) \tag{6.133}$$

where $u(n)$ is the input signal at time n. Therefore, starting with $m = 0$, and progressively increasing the order of the filter by 1, we obtain the *lattice equivalent model* shown in Fig. 6.9(b) for a prediction-error filter of final order M. In this figure we merely require knowledge of the complete set of reflection coefficients $\kappa_1, \kappa_2, \ldots, \kappa_M$, one for each stage of the filter.

The lattice filter, depicted in Fig. 6.9(b), offers the following attractive features:

1. A lattice filter is a highly efficient structure for generating the sequence of forward prediction errors and the corresponding sequence of backward prediction errors simultaneously.
2. The various stages of a lattice predictor are "decoupled" from each other. This decoupling property was indeed derived in Section 6.7, where it was shown that the backward prediction errors produced by the various stages of a lattice predictor are "orthogonal" to each other for wide-sense stationary input data.
3. The lattice filter is *modular* in structure; hence, if the requirement calls for increasing the order of the predictor, we simply add one or more stages (as desired) without affecting earlier computations.
4. All the stages of a lattice predictor have a similar structure; hence, it lends itself to the use of *very large scale integration* (VLSI) technology if the use of this technology is considered beneficial to the application of interest.

Inverse Filtering Using the Lattice Structure

The multistage lattice predictor of Fig. 6.9(b) may be viewed as an *analyzer*. That is, it enables us to represent an autoregressive (AR) process $u(n)$ by a corresponding sequence of reflection coefficients $\{\kappa_m\}$. By rewiring this multistage lattice predictor in the manner depicted in Fig. 6.10, we may use this new structure as a *synthesizer* or *inverse filter*. That is, given the sequence of reflection coefficients $\{\kappa_m\}$, we may reproduce the original AR process by applying a white-noise process $v(n)$ to the input of the structure in Fig. 6.10.

[7] The first application of lattice filters in on-line adaptive signal processing was apparently made by Itakura and Saito (1971) in the field of speech analysis. Equivalent lattice-filter models, however, were familiar in geophysical signal processing as "layered earth models" (Robinson, 1967; Burg, 1968). It is also of interest to note that such lattice filters have been well studied in network theory, especially in the cascade synthesis of multiport networks (Dewilde, 1969).

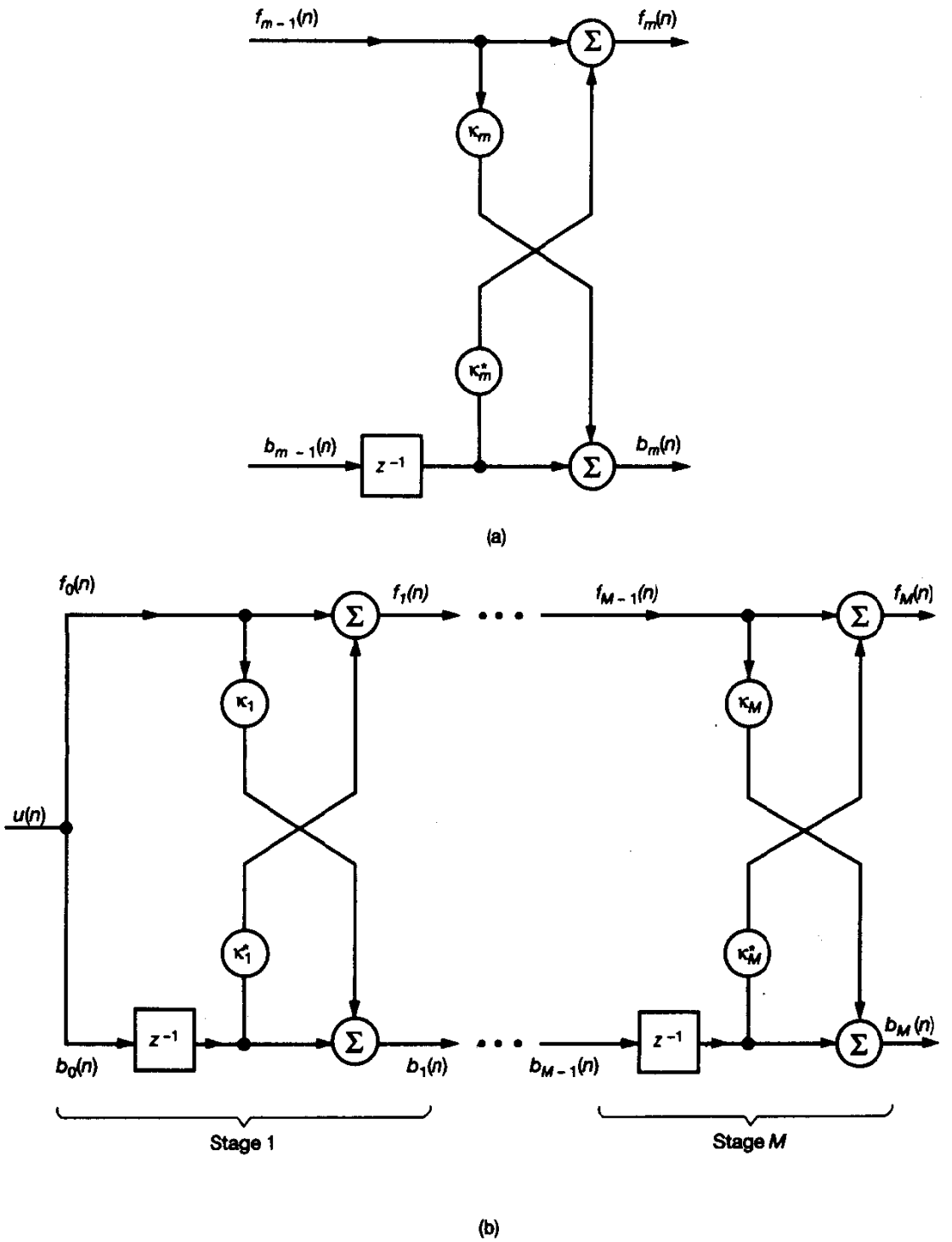


Figure 6.9 Signal-flow graph for stage m of a lattice predictor; (b) lattice equivalent model of prediction-error filter of order M.

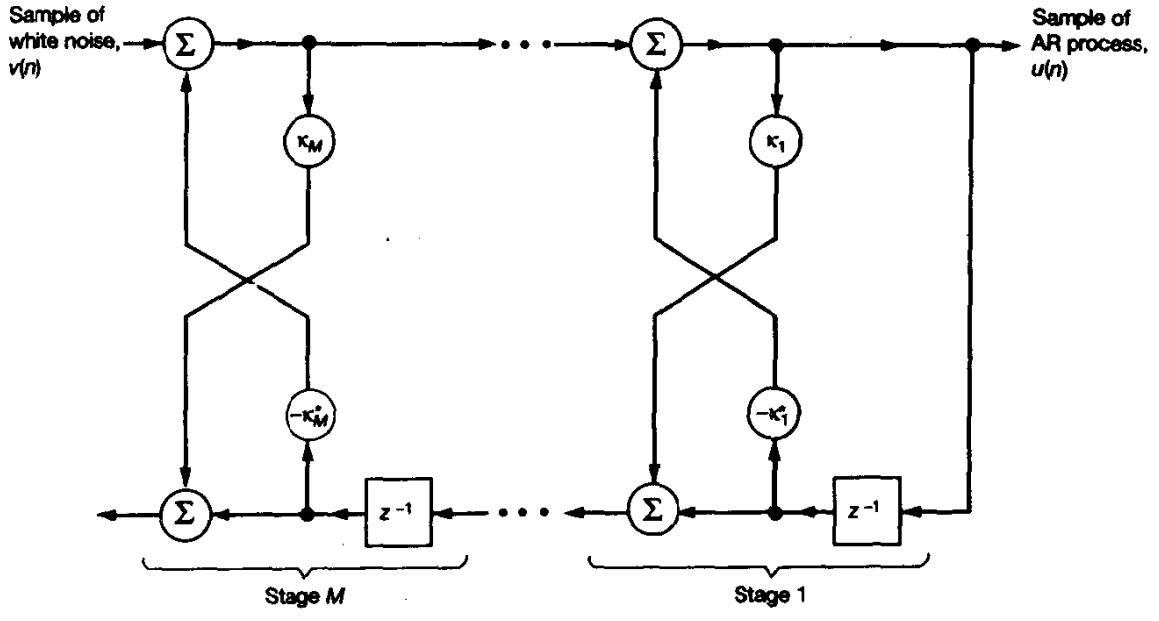


Figure 6.10 Lattice inverse filter of order M.

We illustrate the operation of the lattice inverse filter with an example.

Example 5

Consider the two-stage lattice inverse filter of Fig. 6.11(a). There are four possible paths that can contribute to the makeup of the sample $u(n)$ at the output, as illustrated in Fig. 6.11(b). In particular, we have

$$\begin{aligned} u(n) &= v(n) - \kappa_1^* u(n-1) - \kappa_1 \kappa_2^* u(n-1) - \kappa_2^* u(n-2) \\ &= v(n) - (\kappa_1^* + \kappa_1 \kappa_2^*) u(n-1) - \kappa_2^* u(n-2) \end{aligned}$$

From Example 2, we recall that

$$\begin{aligned} a_{2,1} &= \kappa_1 + \kappa_1^* \kappa_2 \\ a_{2,2} &= \kappa_2 \end{aligned}$$

We may therefore express the mechanism governing the generation of process $u(n)$ as follows:

$$u(n) + a_{2,1}^* u(n-1) + a_{2,2}^* u(n-2) = v(n)$$

which is recognized as the difference equation of a second-order AR process.

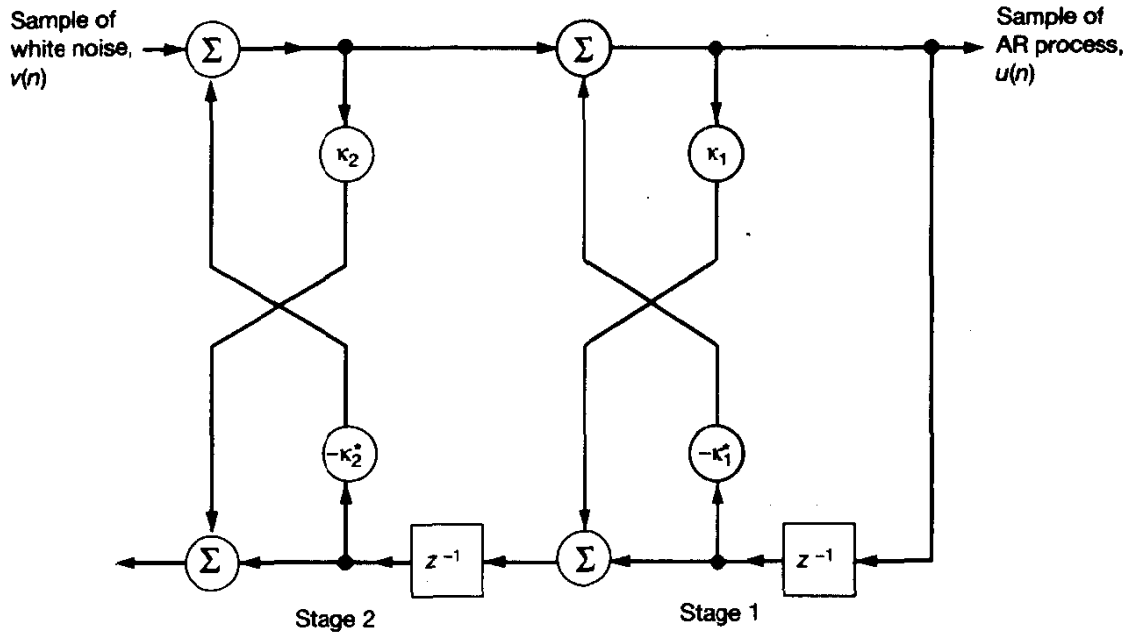


Figure 6.11 (a) Lattice inverse filter of order 2.

6.9 JOINT-PROCESS ESTIMATION

In this section, we use the lattice predictor as a subsystem to solve a *joint-process estimation problem* that is optimal in the mean-square sense (Griffiths, 1978; Makhoul, 1978). In particular, we consider the minimum mean-square estimation of a process $d(n)$, termed the desired response, by using a set of *observables* derived from a related process $u(n)$. We assume that the processes $d(n)$ and $u(n)$ are jointly stationary. This estimation problem is similar to that considered in Chapter 5, with one basic difference. In Chapter 5 we used samples of the process $u(n)$ as the observables directly. Our approach here is different in that for the observables we use the set of backward prediction errors obtained by feeding the input of a multistage lattice predictor with samples of the process $u(n)$. The fact that the backward prediction errors are orthogonal to each other simplifies the solution to the problem significantly.

The structure of the *joint-process estimator* is shown in Fig. 6.12. This device performs two optimum estimations jointly:

1. The *lattice predictor*, consisting of a cascade of M stages, characterized individually by the reflection coefficients $\kappa_1, \kappa_2, \ldots, \kappa_M$, performs predictions (of varying orders) on the input. In particular, it transforms the sequence of (correlated) input samples $u(n)$, $u(n-1)$, $\ldots$, $u(n-M)$ into a corresponding sequence of (uncorrelated) backward prediction errors $b_0(n)$, $b_1(n)$, $\ldots$, $b_M(n)$.

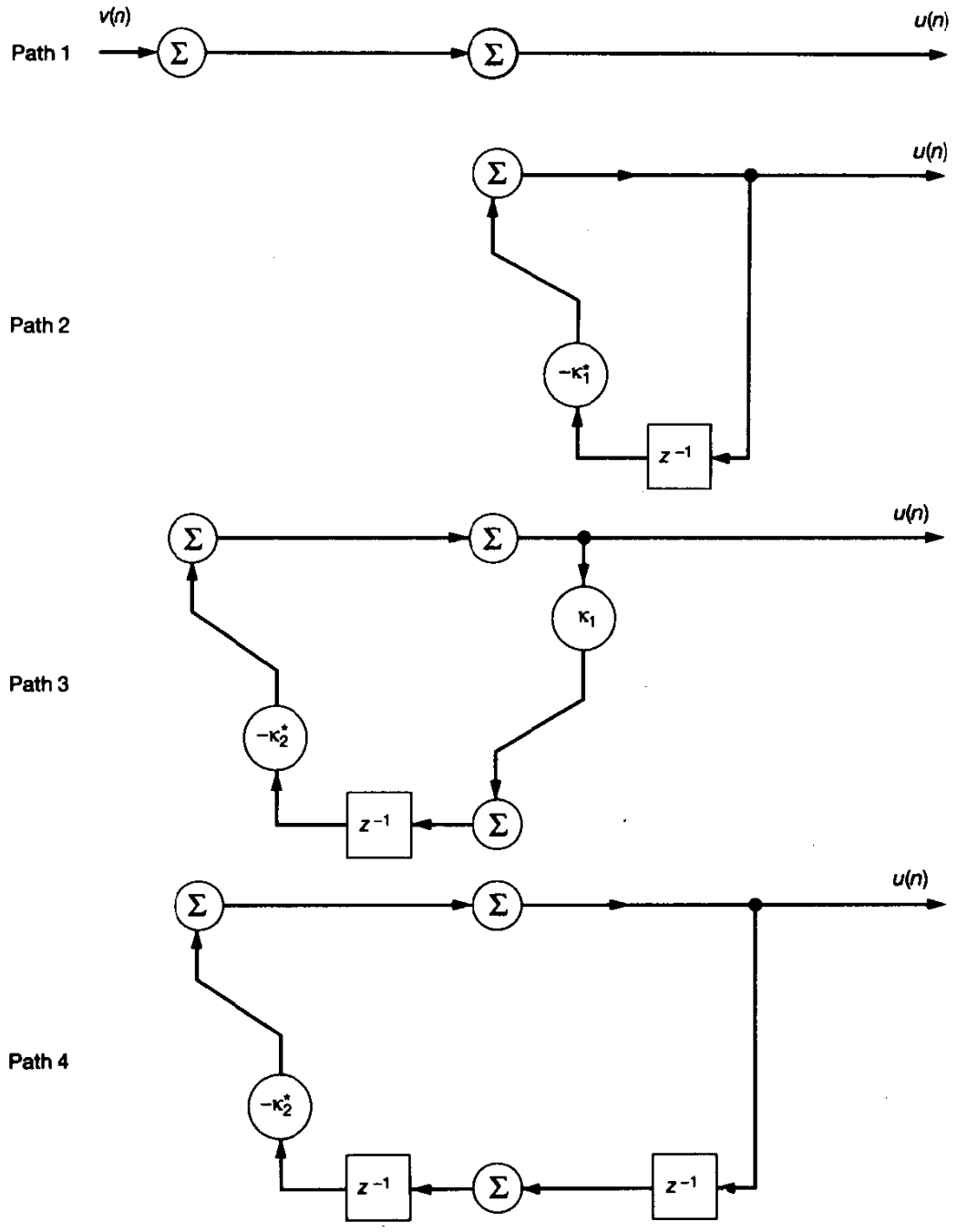


Figure 6.11 (b) The four possible paths that contribute to the makeup of the output $u(n)$ in the lattice inverse filter of Fig. 6.11(a).

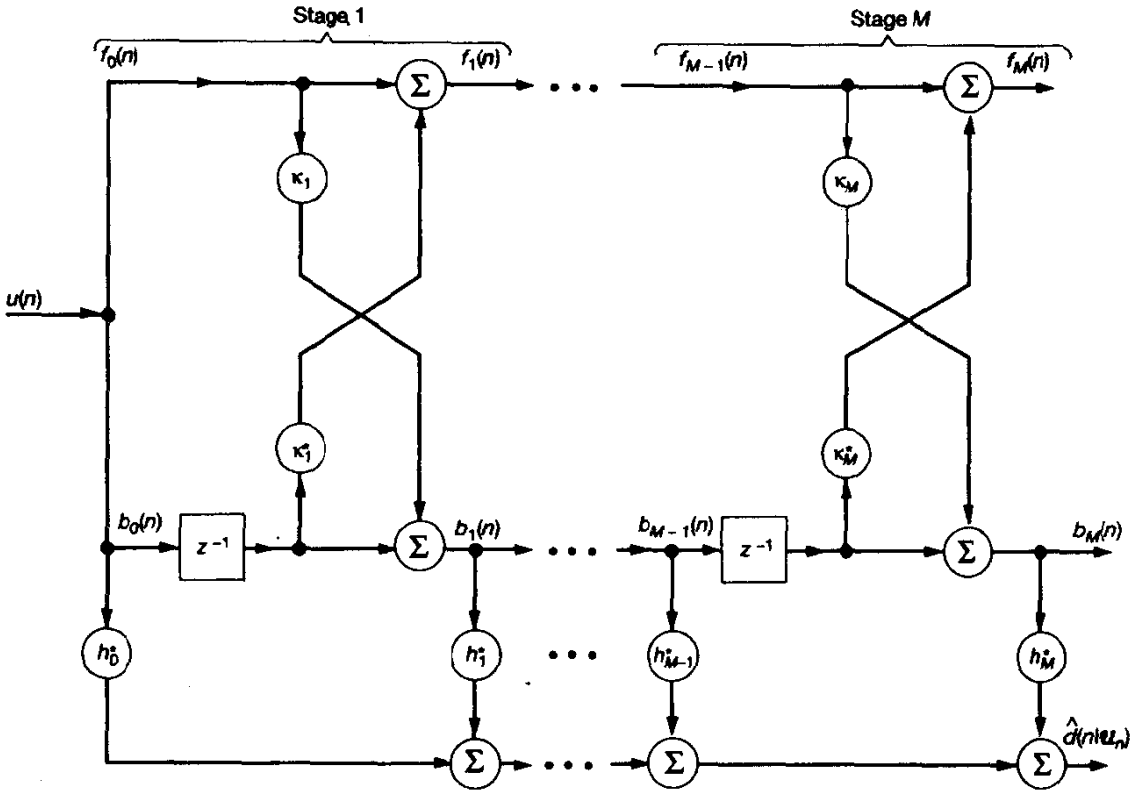


Figure 6.12 Lattice-based structure for joint-process estimation.

2. The multiple regression filter, characterized by the set of weights $h_0, h_1, \ldots, h_M$, operates on the sequence of backward prediction errors $b_0(n), b_1(n), \ldots, b_M(n)$ as inputs, respectively, to produce an estimate of the desired response $d(n)$. The resulting estimate is defined as the sum of the respective scalar inner products of these two sets of quantities, as shown by

$$\hat{d}(n|\mathcal{U}_n) = \sum_{i=0}^{M} h_i^* b_i(n) \tag{6.134}$$

where $\mathcal{U}_n$ is the space spanned by the inputs $u(n), u(n-1), \ldots, u(n-M)$. We may rewrite Eq. (6.134) in matrix form as follows:

$$\hat{d}(n|\mathcal{U}_n) = \mathbf{h}^H \mathbf{b}(n) \tag{6.135}$$

where $\mathbf{h}$ is an $(M+1)$-by-1 vector defined by

$$\mathbf{h} = [h_0, h_1, \ldots, h_M]^T \tag{6.136}$$

We refer to $h_0, h_1, \ldots, h_M$ as the regression coefficients of the estimator, and to $\mathbf{h}$ as the regression-coefficient vector.

Let $\mathbf{D}$ denote the $(M + 1)$-by-$(M + 1)$ correlation matrix of $\mathbf{b}(n)$, the $(M + 1)$-by-1 vector of backward prediction errors $b_0(n), b_1(n), \ldots, b_M(n)$. Let $\mathbf{z}$ denote the $(M + 1)$-by-1 cross-correlation vector between the backward prediction errors and the desired response as shown by

$$\mathbf{z} = E[\mathbf{b}(n)d^*(n)] \tag{6.137}$$

Therefore, applying the Wiener–Hopf equations to our present situation, we find that the optimum regression-coefficient vector $\mathbf{h}_o$ is defined by

$$\mathbf{D}\mathbf{h}_o = \mathbf{z} \tag{6.138}$$

Solving for $\mathbf{h}_o$, we get

$$\mathbf{h}_o = \mathbf{D}^{-1}\mathbf{z} \tag{6.139}$$

where the inverse matrix $\mathbf{D}^{-1}$ is a diagonal matrix, defined in terms of various prediction-error powers as in Eq. (6.117). Note that, unlike the ordinary transversal filter realization of the Wiener filter, the computation of $\mathbf{h}_o$ in the joint-process estimator of Fig. 6.12 is relatively simple to accomplish.

Relationship between the Optimum Regression-Coefficient Vector and the Wiener Solution

From the Cholesky factorization given in Eq. (6.118), we deduce that

$$\mathbf{D}^{-1} = \mathbf{L}^{-H}\mathbf{R}^{-1}\mathbf{L}^{-1} \tag{6.140}$$

Hence, substituting Eq. (6.140) in (6.139) yields

$$\mathbf{h}_o = \mathbf{L}^{-H}\mathbf{R}^{-1}\mathbf{L}^{-1}\mathbf{z} \tag{6.141}$$

Moreover, from Eq. (6.105) we note that

$$\mathbf{b}(n) = \mathbf{L}\mathbf{u}(n) \tag{6.142}$$

Therefore, substituting Eq. (6.142) in (6.137) yields

$$\begin{aligned}\mathbf{z} &= \mathbf{L}E[\mathbf{u}(n)d^*(n)] \\ &= \mathbf{L}\mathbf{p}\end{aligned} \tag{6.143}$$

where $\mathbf{p}$ is the cross-correlation vector between the tap-input vector $\mathbf{u}(n)$ and the desired response $d(n)$. Thus, using Eq. (6.143) in (6.141), we finally obtain

$$\begin{aligned}\mathbf{h}_o &= \mathbf{L}^{-H}\mathbf{R}^{-1}\mathbf{L}^{-1}\mathbf{L}\mathbf{p} \\ &= \mathbf{L}^{-H}\mathbf{R}^{-1}\mathbf{p} \\ &= \mathbf{L}^{-H}\mathbf{w}_o\end{aligned} \tag{6.144}$$

where $\mathbf{L}$ is a lower triangular matrix defined in terms of the equivalent forward prediction-error filter coefficients, as in Eq. (6.106). Equation (6.144) is the sought-after relationship between the optimum regression-coefficient vector $\mathbf{h}_o$ and the Wiener solution $\mathbf{w}_o = \mathbf{R}^{-1}\mathbf{p}$.

6.10 BLOCK ESTIMATION

In this chapter, we have discussed two basic structures for building a *linear predictor* or its natural extension, a *prediction-error filter*; the two structures are a *transversal filter* and a *lattice filter*.[8] The transversal filter is characterized by a set of *tap weights*, whereas the lattice filter is characterized by a corresponding set of *reflection coefficients*. In both cases, the filter coefficients provide the designer with *degrees of freedom*, the number of which equals the *prediction order*. The mathematical link between the tap weights of a transversal predictor and the reflection coefficients of a lattice predictor is provided by the Levinson–Durbin algorithm.

Regardless of the particular structure chosen, we clearly need a procedure for *estimating* the filter coefficients. To carry out this estimation we have two approaches to consider:

- Block estimation
- Adaptive estimation

In *block estimation*, the available data are divided into individual *blocks*, each of length N, say. The *block length* N is usually chosen short enough to ensure pseudostationarity of the input data over the length N. Under this assumption, the filter coefficients of interest are then computed on a *block-by-block* basis. Typically, the filter coefficients vary from one block of data to another. The block estimation algorithms may be categorized as follows:

1. *Indirect methods*. For each block of data, estimates of the autocorrelation function of the input are computed for different lags. The Levinson–Durbin algorithm is then used to compute the corresponding set of tap weights for a transversal predictor, or the corresponding set of reflection coefficients for a lattice predictor, depending on the application of interest.

[8] In actual fact, there is a third structure for building a linear predictor that is based on the *Schur algorithm* (Schur, 1917). Like the Levinson–Durbin algorithm, the Schur algorithm provides a procedure for computing the reflection coefficients from a known autocorrelation sequence. The Schur algorithm, however, lends itself to parallel implementation, with the result that it achieves a throughput rate higher than that obtained using the Levinson–Durbin algorithm. For a discussion of the Schur algorithm, including mathematical details and implementational considerations, see Haykin (1989).

2. *Direct methods*. For each block of data, estimates of the reflection coefficients for the different stages of a lattice predictor are computed directly from the data. The reflection coefficients lend themselves to direct computation because of the decoupling property of a multistage lattice predictor for wide-sense stationary inputs.

From a computational viewpoint, direct methods are more efficient than indirect methods if the application of interest requires knowledge of the reflection coefficients. If, on the other hand, the application is that of system identification in terms of the tap weights of a transversal filter, as in autoregressive modeling, for example, the computational advantage of direct methods over indirect methods may well disappear. In this section, we develop a popular algorithm known as the *Burg algorithm*, which may be classified as a direct block estimation method.[9]

Consider a lattice predictor consisting of M stages connected in cascade as.in Fig. 6.9(b). For wide-sense stationary input data, the M stages of this lattice predictor are decoupled from each other by virtue of the orthogonality of its backward prediction-error outputs; see Section 6.7. This decoupling property makes it possible for us to accomplish the *global optimization of a multistage lattice predictor as a sequence of local optimizations, one at each stage of the structure*. Moreover, it is a straightforward matter to increase the order of the predictor by simply adding one or more stages, as required, without affecting the earlier design computations. For example, suppose that we have optimized the design of a lattice predictor consisting of M stages. To increase the order of the predictor by 1, we simply add a new stage that is locally optimized, leaving the optimum design of the earlier stages unchanged.

Consider stage m of the lattice predictor shown in Fig. 6.9(a), for which the input–output relations are given in matrix form in Eq. (6.131). For convenience of presentation, these relations are reproduced here in the expanded form:

$$f_m(n) = f_{m-1}(n) + \kappa_m^* b_{m-1}(n-1) \tag{6.145}$$

$$b_m(n) = b_{m-1}(n-1) + \kappa_m f_{m-1}(n) \tag{6.146}$$

where $m = 1, 2, \ldots, M$, and M is the final order of the predictor. Several criteria may be used to optimize the design of this stage (Makhoul, 1977). However, one particular criterion yields a design with interesting properties that conform to the lattice predictor theory. Specifically, the reflection coefficient κ_m is chosen so as to minimize the sum of the mean-squared values of the forward and backward prediction errors. Let the cost function J_m denote this sum at the output of stage m of the lattice predictor:

$$J_m = E[|f_m(n)|^2] + E[|b_m(n)|^2] \tag{6.147}$$

[9] For a more complete discussion of block estimation, including both indirect and direct methods, see Marple (1987), Kay (1988), and Haykin (1989).

Substituting Eqs. (6.145) and (6.146) in (6.147), we get

$$\begin{aligned} J_m = {} & \{E[|f_{m-1}(n)|^2] + E[|b_{m-1}(n-1)|^2]\}[1 + |\kappa_m|^2] \\ & + 2\kappa_m E[f_{m-1}(n)b^*_{m-1}(n-1)] \\ & + 2\kappa^*_m E[b_{m-1}(n-1)f^*_{m-1}(n)] \end{aligned} \tag{6.148}$$

In general, the reflection coefficient κ_m is complex valued, as shown by

$$\kappa_m = \alpha_m + j\beta_m \tag{6.149}$$

Therefore, differentiating the cost function J_m with respect to both the real and imaginary parts of κ_m, we get the complex-valued *gradient*

$$\begin{aligned} \nabla J_m &= \frac{\partial J_m}{\partial \alpha_m} + j\frac{\partial J_m}{\partial \beta_m} \\ &= 2\kappa_m\{E[|f_{m-1}(n)|^2] + E[|b_{m-1}(n-1)|^2]\} \\ &\quad + 4E[b_{m-1}(n-1)f^*_{m-1}(n)] \end{aligned} \tag{6.150}$$

Putting this gradient to zero, we find that the optimum value of the reflection coefficient, for which the cost function J_m is minimum, equals

$$\kappa_{m,o} = -\frac{2E[b_{m-1}(n-1)f^*_{m-1}(n)]}{E[|f_{m-1}(n)|^2 + |b_{m-1}(n-1)|^2]}, \qquad m = 1, 2, \ldots, M \tag{6.151}$$

Equation (6.151) for the reflection coefficient is known as the *Burg formula* (Burg, 1968).[10] Its use offers two interesting properties (see Problem 23):

1. The reflection coefficient $\kappa_{m,o}$ satisfies the condition

$$|\kappa_{m,o}| \leq 1 \quad \text{for all } m \tag{6.152}$$

In other words, the Burg formula always yields a minimum-phase design for the lattice predictor.

2. The mean-square values of the forward and backward prediction errors at the output of stage m are related to those at its own input as follows, respectively:

$$E[|f_m(n)|^2] = (1 - |\kappa_{m,o}|^2)E[|f_{m-1}(n)|^2] \tag{6.153}$$

and

$$E[|b_m(n)|^2] = (1 - |\kappa_{m,o}|^2)E[|b_{m-1}(n-1)|^2] \tag{6.154}$$

The Burg formula, as described in Eq. (6.151), involves the use of ensemble averaging. Assuming that the input $u(n)$ is *ergodic*, we may substitute time averages for the

[10] The 1968 paper by Burg is reproduced in the book by Childers (1978).

expectations in the dominator and denominator of this equation. We thus get the *Burg estimate*[11] for the reflection coefficient of stage m in the lattice predictor

$$\hat{\kappa}_m = -\frac{2\sum_{n=m+1}^{N} b_{m-1}(n-1)f_{m-1}^*(n)}{\sum_{n=m+1}^{N}[|f_{m-1}(n)|^2 + |b_{m-1}(n-1)|^2]}, \qquad m = 1, 2, \ldots M \tag{6.155}$$

where N is the length of a block of input data, and $f_0(n) = b_0(n) = u(n)$.

With a lattice predictor of m stages, each of which contains a single unit-delay element, and with the input $u(n)$ zero for $n \leq 0$, we find that *all* the samples in the input data contribute to the outputs of stage m in the predictor for the first time at $n = m + 1$, hence the use of this value for the lower limits of the summation terms in Eq. (6.155). Note also that the estimate $\hat{\kappa}_m$ for the mth reflection coefficient is dependent on data length N. The choice of N is usually dictated by two conflicting factors. First, it should be large enough to smooth out the effects of noise in computing the time averages in the numerator and denominator of Eq. (6.155). Second, as mentioned previously, it should be small enough to ensure quasi-statistical stationarity of the input data during the computations, and thereby justify the application of Burg's formula.

The block-estimation approach usually requires a large amount of computation, as well as a large amount of storage. Furthermore, in this approach, we find that for any stage of the predictor the estimate of the reflection coefficient at time $n + 1$ does not depend in a simple way on its previous estimate at time n. This behavior is to be contrasted with the adaptive estimation procedure described in subsequent chapters of the book.

6.11 SUMMARY AND DISCUSSION

In this chapter we presented a detailed study of the linear prediction problem pertaining to wide-sense stationary stochastic processes. In particular, we used the Wiener filter theory of Chapter 5 to develop optimum solutions for the two basic forms of linear prediction:

- Forward linear prediction, in which case we are given the input sequence $u(n-1), u(n-2), \ldots, u(n-M)$ and the requirement is to make an optimum prediction of the next sample value $u(n)$.

[11] For some practical applications of the Burg estimator given in Eq. (6.155), see Haykin et al. (1982) and Swingler and Walker (1989). The first of these two papers describes a (temporal) procedure based on this estimator for classifying the different forms of radar clutter (e.g., radar returns from different targets) as encountered in an air traffic control environment. The second paper presents a demonstration of a linear array beamformer based on a spatial interpretation of the Burg estimator for sonar environment. The studies reported in both of these papers employ real-life data.

- Backward linear prediction, in which case we are given the input sequence $u(n), u(n-1), \ldots, u(n-M+1)$ and the requirement is to make an optimum prediction of the old sample value $u(n-M)$.

In both cases, the desired response is derived from the time series itself. In forward linear prediction $u(n)$ acts as the desired response, whereas in backward linear prediction $u(n-M)$ acts as the desired response.

The prediction process may be described in terms of a predictor or, equivalently, a prediction-error filter. These two linear devices differ from each other in their respective outputs. The output of a forward predictor is a one-step prediction of the input. On the other hand, the output of a forward prediction-error filter is the prediction error. In a similar way, we may distinguish between a backward predictor and a backward prediction-error filter.

The two structures most commonly used for building a prediction-error filter are:

- Transversal filter, where the issue of concern is the determination of the tap weights
- Lattice filter, where the issue of concern is the determination of the reflection coefficients

These two sets of parameters are in fact uniquely related to each other via the Levinson–Durbin recursion.

The important properties of prediction-error filters may be summarized as follows:

- The forward prediction-error filter is minimum phase, which means that all the zeros of its transfer function lie inside the unit circle in the z-plane. The corresponding inverse filter, representing an autoregressive model of the input process, is therefore stable.
- The backward prediction-error filter is maximum phase, which means that all the zeros of its transfer function lie outside the unit circle in the z-plane. In this case, the inverse filter is unstable and therefore of no practical value.
- The forward prediction-error filter is a whitening filter, whereas the backward prediction-error filter is an anticausal whitening filter (see Problem 14).

The lattice predictor offers some highly desirable properties:

- *Order-recursive structure*, which means that the prediction order may be increased by adding one or more stages to the structure without destroying the previous calculations.
- *Modularity*, which is exemplified by the fact that all the stages of the lattice predictor have exactly the same physical structure.
- *Simultaneous computation of forward and backward prediction errors*, which provides for computational efficiency.

- *Statistical decoupling of the individual stages*, which is another way of saying that the backward prediction errors of varying orders produced by the different stages of the lattice predictor are uncorrelated with each other. This property, embodied in the Cholesky factorization, is exploited in the joint-estimation process, where the backward prediction errors are used to provide an estimate of some desired response.

PROBLEMS

1. The augmented Wiener–Hopf equations (6.14) of a forward prediction-error filter were derived by first optimizing the linear prediction filter in the mean-square sense and then combining the two resultants: the Wiener–Hopf equations for the tap-weight vector and the minimum mean-squared prediction error. This problem addresses the issue of deriving Eq. (6.14) directly by proceeding as follows:
   (a) Formulate the expression for the mean-square value of the forward prediction error as a function of the tap-weight vector of the forward prediction-error filter.
   (b) Minimize this mean-squared prediction error, subject to the constraint that the leading element of the tap-weight vector of the forward prediction-error filter equals 1.

   Hint: Use the method of Lagrange multipliers to solve the constrained optimization problem. For details of this method, see Appendix C. This hint also applies to part (b) of Problem 2.
2. The augmented Wiener–Hopf equations (6.38) of a backward prediction-error filter were derived indirectly in Section 6.2. This problem addresses the issue of deriving Eq. (6.38) directly by proceeding as follows:
   (a) Formulate the expression for the mean-square value of the backward prediction error in terms of the tap-weight vector of the backward prediction-error filter.
   (b) Minimize this mean-squared prediction error, subject to the constraint that the last element of the tap-weight vector of the backward prediction-error filter equals 1.
3. (a) Equation (6.24) defines the Wiener–Hopf equations for backward linear prediction. This system of equations is reproduced here for convenience:

$$\mathbf{R}\mathbf{w}_b = \mathbf{r}^{B*}$$

   where $\mathbf{w}_b$ is the tap-weight vector of the predictor, $\mathbf{R}$ is the correlation matrix of the tap inputs $u(n), u(n-1), \ldots, u(n-M+1)$, and $\mathbf{r}^{B*}$ is the cross-correlation vector between these tap inputs and the desired response $u(n-M)$. Show that if the elements of the column vector $\mathbf{r}^{B*}$ are rearranged in reverse order, the effect of this reversal is to modify the Wiener–Hopf equations as

$$\mathbf{R}^T\mathbf{w}_b^B = \mathbf{r}^*$$

   (b) Show that the inner products $\mathbf{r}^{BT}\mathbf{w}_b$ and $\mathbf{r}^T\mathbf{w}_b^B$ are equal.
4. Consider a wide-sense stationary process $u(n)$ whose autocorrelation function has the following values for different lags:

$$r(0) = 1$$
$$r(1) = 0.8$$
$$r(2) = 0.6$$
$$r(3) = 0.4$$

(a) Use the Levinson–Durbin recursion to evaluate the reflection coefficients κ_1, κ_2, and κ_3.
(b) Set up a three-stage lattice predictor for this process using the values for the reflection coefficients found in part (a).
(c) Evaluate the average power of the prediction error produced at the output of each of the three stages in this lattice predictor. Hence, make a plot of prediction-error power versus prediction order. Comment on your results.

5. Consider the filtering structure described in Fig. P6.1, where the delay Δ is an integer greater than one. The requirement is to choose the weight vector $\mathbf{w}$ so as to minimize the mean-square value of the estimation error $e(n)$. Find the optimum value of $\mathbf{w}(n)$.

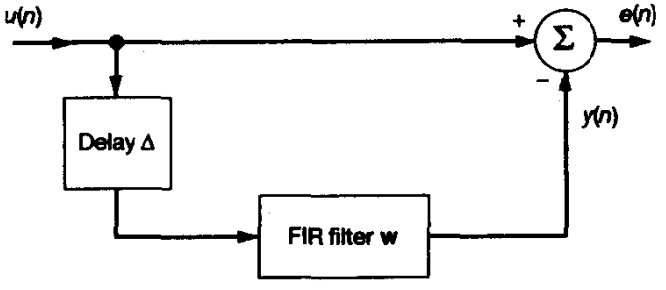


Figure P6.1

6. Consider the linear prediction of a stationary autoregressive process $u(n)$, generated from the *first-order difference equation*

$$u(n) = 0.9u(n-1) + v(n)$$

where $v(n)$ is a white noise process of zero mean and unit variance. The prediction order is two.
(a) Determine the tap weights $a_{2,1}$ and $a_{2,2}$ of the forward prediction-error filter.
(b) Determine the reflection coefficients κ_1 and κ_2 of the corresponding lattice predictor.
Comment on your results in parts (a) and (b).

7. (a) A process $u_1(n)$ consists of a single sinusoidal process of complex amplitude α and angular frequency ω in additive white noise of zero mean and variance σ_v^2, as shown by

$$u_1(n) = \alpha e^{j\omega n} + v(n)$$

where

$$E[|\alpha|^2] = \sigma_\alpha^2$$

and

$$E[|v(n)|^2] = \sigma_v^2$$

The process $u_1(n)$ is applied to a linear predictor of order M, optimized in the mean-squared error sense. Do the following:
(i) Determine the tap weights of the prediction-error filter of order M, and the final value of the prediction error power P_M.
(ii) Determine the reflection coefficients $\kappa_1, \kappa_2, \ldots, \kappa_M$ of the corresponding lattice predictor.
(iii) How are the results in part (i) and (ii) modified when we let the noise variance σ_v^2 approach zero?

(b) Consider next an AR process $u_2(n)$ described by

$$u_2(n) = -\alpha e^{j\omega} u_2(n-1) + v(n)$$

where, as before, $v(n)$ is an additive white noise process of zero mean and variance σ_v^2. Assume that $0 < |\alpha| < 1$ but very close to 1. The process $u_2(n)$ is also applied to a linear predictor of order M, optimized in the mean-squared error sense.

(i) Determine the tap weights of the new prediction-error filter of order M.

(ii) Determine the reflection coefficients $\kappa_1, \kappa_2, \ldots, \kappa_M$ of the corresponding lattice predictor.

(c) Use your results in parts (a) and (b) to compare the similarities and differences between the linear prediction of the processes $u_1(n)$ and $u_2(n)$.

8. Equation (6.40) defines the Levinson–Durbin recursion for forward linear prediction. By rearranging the elements of the tap-weight vector $\mathbf{a}_m$ backward and then complex-conjugating them, reformulate the Levinson–Durbin recursion for backward linear prediction as in Eq. (6.42).

9. Starting with the definition of Eq. (6.47) for Δ_{m-1}, show that Δ_{m-1} equals the cross-correlation between the delayed backward prediction error $b_{m-1}(n-1)$ and the forward prediction error $f_{m-1}(n)$.

10. Develop in detail the relationship between the Schur–Cohn method and the inverse recursion as outlined by Eqs. (6.97) through (6.100).

11. Consider an autoregressive process $u(n)$ of order 2, described by the difference equation

$$u(n) = u(n-1) - 0.5u(n-2) + v(n)$$

where $v(n)$ is a white-noise process of zero mean and variance 0.5.

(a) Find the average power of $u(n)$.

(b) Find the reflection coefficients κ_1 and κ_2.

(c) Find the average prediction-error powers P_1 and P_2.

12. Using the one-to-one correspondence between the two sequences of numbers $\{P_0, \kappa_1, \kappa_2\}$ and $\{r(0), r(1), r(2)\}$, compute the autocorrelation function values $r(1)$ and $r(2)$ that correspond to the reflection coefficients κ_1, and κ_2 found in Problem 11 for the second-order autoregressive process $u(n)$.

13. In Section 6.4, we presented a proof of the minimum-phase property of a prediction-error filter by using Rouché's theorem. In this problem, we explore another proof of this property by contradiction. Consider Fig. P6.2, which shows the prediction-error filter (of order M) represented as the cascade of two functional blocks, one with transfer function $C_i(z)$ and the other with its transfer function equal to the zero factor $(1 - z_i z^{-1})$. Let $S(\omega)$ denote the power spectral density of the process $u(n)$ applied to the input of the prediction-error filter.

(a) Show that the mean-square value of the forward prediction error $f_M(n)$ equals

$$\epsilon = \int_{-\pi}^{\pi} S(\omega)|C_i(e^{j\omega})|^2[1 - 2\rho_i\cos(\omega - \omega_i) + \rho_i^2]d\omega$$

where

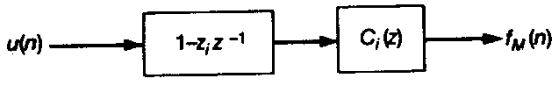


Figure P6.2

(b) Suppose that $\rho_i > 1$ so that the complex zero lies outside the unit circle. Hence, show that under this condition $\partial\epsilon/\partial\rho_i > 0$. Is such a condition possible and at the same time the filter operates at its optimum condition? What conclusion can you draw from your answers?

14. When an autoregressive process of order M is applied to a forward prediction-error filter of order M, the output consists of white noise. Show that when such a process is applied to a backward prediction-error filter of order M, the output consists of an anticausal realization of white noise.

15. Consider a forward prediction-error filter characterized by a real-valued set of coefficients $a_{m,1}$, $a_{m,2}, \ldots, a_{m,m}$. Define a polynomial $\phi_m(z)$ as follows:

$$\sqrt{P_m}\,\phi_m(z) = z^m + a_{m,1}z^{m-1} + \cdots + a_{m,m}$$

where P_m is the average prediction-error power of order m, and z^{-1} is the unit-delay operator. [Note the difference between the definition of $\phi_m(z)$ and that of the corresponding transfer function $H_{f,m}(z)$ of the filter.] The filter coefficients bear a one-to-one correspondence with the sequence of autocorrelations $r(0), r(1), \ldots, r(m)$. Define.

$$S(z) = \sum_{i=-m}^{m} r(i)z^{-i}$$

Show that

$$\frac{1}{2\pi j}\oint_{\mathscr{C}} \phi_m(z)\phi_k(z^{-1})S(z)\,dz = \delta_{mk}$$

where δ_{mk} is the Kronecker delta:

$$\delta_{mk} = \begin{cases} 1, & k = m \\ 0, & k \neq m \end{cases}$$

and the contour $\mathscr{C}$ is the unit circle. The polynomial $\phi_m(z)$ is referred to as a *Szegö polynomial*.

16. (a) Construct the two-stage lattice predictor for the second-order autoregressive process $u(n)$ considered in Problem 11.

(b) Given a white-noise process $v(n)$, construct the two-stage lattice synthesizer for generating the autoregressive process $u(n)$. Check your answer against the second-order difference equation for the process $u(n)$ that was considered in Problem 11.

17. In a *normalized* lattice predictor, the forward and backward prediction errors at the various stages of the predictor are all normalized to have *unit variance*. Such an operation makes it possible to utilize the full dynamic range of multipliers used in the hardware implementation of a lattice predictor. For stage m of the normalized lattice predictor, the normalized forward and backward prediction errors are defined as follows, respectively:

$$\bar{f}_m(n) = \frac{f_m(n)}{P_m^{1/2}}$$

and

$$\bar{b}_m(n) = \frac{b_m(n)}{P_m^{1/2}}$$

where P_m is the average power (or variance) of the forward prediction error $f_m(n)$ or that of the backward prediction error $b_m(n)$. Show that the structure of stage m of the normalized lattice predictor is as shown in Fig. P6.3 for real-valued data.

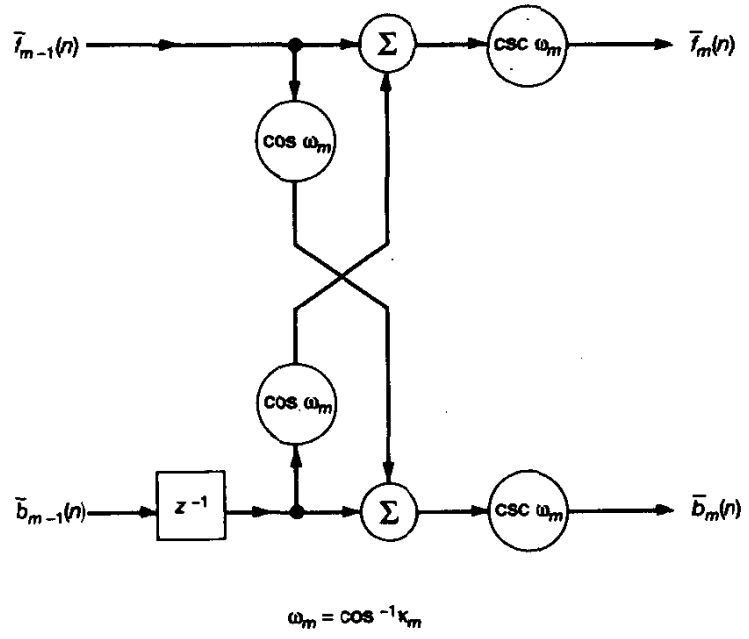


Figure P6.3

18. **(a)** Consider the matrix product **LR** that appears in the decomposition of Eq. (6.114), where the $(M + 1)$-by-$(M + 1)$ lower triangular matrix **L** is defined in Eq. (6.106) and **R** is the $(M + 1)$-by-$(M + 1)$ correlation matrix. Let **Y** denote this matrix product, and let y_{mk} denote the mkth element of **Y**. Hence, show that

$$y_{mm} = P_m, \qquad m = 0, 1, \ldots, M$$

where P_m is the prediction-error power for order m.

(b) Show that the mth column of matrix **Y** is obtained by passing the autocorrelation sequence $\{r(0), r(1), \ldots, r(m)\}$ through a corresponding sequence of backward prediction-error filters represented by the transfer functions $H_{b,0}(z), H_{b,1}(z), \ldots, H_{b,m}(z)$.

(c) Suppose that we apply the autocorrelation sequence $\{r(0), r(1), \ldots, r(m)\}$ to the input of a lattice predictor of order m. Show that the variables appearing at the various points on the lower line of the predictor at time m equal the elements of the mth column of matrix **Y**.

(d) For the situation described in part (c), show that the lower output of stage m in the predictor at time m equals P_m, and that the upper output of this same stage at time $m + 1$ equals Δ_m^*. How is the ratio of these two outputs related to the reflection coefficient of stage $m + 1$?

(e) Use the results of part (d) to develop a recursive procedure for computing the sequence of reflection coefficients from the autocorrelation sequence.

19. In Section 6.8, we considered the use of an inverse lattice filter as the generator of an autoregressive process. The lattice inverse filter may also be used to efficiently compute the autocor-

relation sequence $r(1), r(2), \ldots, r(m)$ normalized with respect to $r(0)$. The procedure involves initializing the states (i.e., unit-delay elements) of the lattice inverse filter to $1, 0, \ldots, 0$ and then allowing the filter to operate with zero input. In effect, this procedure provides a lattice interpretation of Eq. (6.67) that relates the autocorrelation sequence $\{r(0), r(1), \ldots, r(M)\}$ and the augmented sequence of reflection coefficients $\{P_0, \kappa_1, \ldots, \kappa_M\}$. Demonstrate the validity of this procedure for the following values of final order M:

(a) $M = 1$

(b) $M = 2$

(c) $M = 3$

20. Prove the following correlation properties of lattice filters:

(a) $E[f_m(n)u^*(n-k)] = 0, \quad 1 \le k \le m$

$E[b_m(n)u^*(n-k)] = 0, \quad 0 \le k \le m-1$

(b) $E[f_m(n)u^*(n)] = E[b_m(n)u^*(n-m)] = P_m$

(c) $E[b_m(n)b_i^*(n)] = \begin{cases} P_m, & m = i \\ 0, & m \neq i \end{cases}$

(d) $E[f_m(n)f_i^*(n-l)] = E[f_m(n+l)f_i^*(n)] = 0, \quad 1 \le l \le m-i, \; m > i$

$E[b_m(n)b_i^*(n-l)] = E[b_m(n+l)b_i^*(n)] = 0, \quad 0 \le l \le m-i-1, \; m > i$

(e) $E[f_m(n+m)f_i^*(n+i)] = \begin{cases} P_m, & m = i \\ 0, & m \neq i \end{cases}$

$E[b_m(n+m)b_i^*(n+i)] = P_m, \quad m \ge i$

(f) $E[f_m(n)b_i^*(n)] = \begin{cases} \kappa_i^* P_m, & m \ge i \\ 0, & m < i \end{cases}$

21. The *entropy* of a random input vector $\mathbf{u}(n)$ of joint probability density function $f_{\mathbf{U}}(\mathbf{u})$ is defined by the multiple integral

$$H_u = -\int_{-\infty}^{\infty} f_{\mathbf{U}}(\mathbf{u}) \ln [f_{\mathbf{U}}(\mathbf{u})] \, d\mathbf{u}$$

Given that the backward prediction-error vector $\mathbf{b}(n)$ is related to $\mathbf{u}(n)$ by the Gram–Schmidt algorithm of Eq. (6.105), show that the vectors $\mathbf{b}(n)$ and $\mathbf{u}(n)$ are equivalent in the sense that they have the same entropy, and therefore convey the same amount of information.

22. Consider the problem of optimizing stage m of the lattice predictor. The cost function to be used in the optimization is described by

$$J_m(\kappa_m) = aE[|f_m(n)|^2] + (1-a)E[|b_m(n)|^2]$$

where a is a constant that lies between zero and one; $f_m(n)$ and $b_m(n)$ denote the forward and backward prediction errors at the output of stage m, respectively.

(a) Show that the optimum value of the reflection coefficient κ_m for which J_m is at minimum equals

$$\kappa_{m,o}(a) = -\frac{E[b_{m-1}(n-1)f_{m-1}^*(n)]}{(1-a)E[|f_{m-1}(n)|^2] + aE[|b_{m-1}(n-1)|^2]}$$

(b) Evaluate $\kappa_{m,o}(a)$ for each of the following three special conditions:

(1) $a = 1$

(2) $a = 2$

(3) $a = \frac{1}{2}$

Notes: When the parameter $a = 1$, the cost function reduces to

$$J_m(\kappa_m) = E[|f_m(n)|^2]$$

We refer to this criterion as the *forward method.*

When the parameter $a = 0$, the cost function reduces to

$$J_m(\kappa_m) = E[|b_m(n)|^2]$$

We refer to this criterion as the *backward method.*

When the parameter $a = \frac{1}{2}$, the formula for $\kappa_{m,o}(a)$ reduces to the Burg formula.

23. Let $\kappa_{m,o}(1)$ and $\kappa_{m,o}(0)$ denote the optimum values of the reflection coefficient κ_m for stage m of the lattice predictor using the forward method and backward method, respectively, as determined in Problem 22.

(a) Show that the optimum value of $\kappa_{m,o}$ obtained from the Burg formula equals the harmonic mean of the two values $\kappa_{m,o}(1)$ and $\kappa_{m,o}(0)$, as shown by

$$\frac{2}{\kappa_{m,o}} = \frac{1}{\kappa_{m,o}(1)} + \frac{1}{\kappa_{m,o}(0)}$$

(b) Using the result of part (a), show that

$$|\kappa_{m,o}| \leq 1 \qquad \text{for all } m$$

(c) For the case of a lattice predictor using the Burg formula, show that the mean-square values of the forward and backward prediction errors at the output of stage m are related to those at the input as follows, respectively:

$$E[|f_m(n)|^2] = (1 - |\kappa_{m,o}|^2)E[|f_{m-1}(n)|^2]$$

and

$$E[|b_m(n)|^2] = (1 - |\kappa_{m,o}|^2)E[|b_{m-1}(n-1)|^2]$$

CHAPTER

7

Kalman Filters

In this chapter we complete our study of linear optimum filters by developing the basic ideas of *Kalman filtering*. A distinctive feature of a Kalman filter is that its mathematical formulation is described in terms of *state-space concepts*. Another novel feature of a Kalman filter is that its solution is computed *recursively*. In particular, each updated estimate of the state is computed from the previous estimate and the new input data, so only the previous estimate requires storage. In addition to eliminating the need for storing the entire past observed data, a Kalman filter is computationally more efficient than computing the estimate directly from the entire past observed data at each step of the filtering process. The Kalman filter is thus ideally suited for implementation on a digital computer. Most importantly, it has been applied successfully to many practical problems in diverse fields, particularly in aerospace and aeronautical applications.

Our interest in the Kalman filter is motivated by the fact that it provides a unifying framework for the derivation of an important family of adaptive filters known as recursive least-squares filters, as demonstrated in subsequent chapters of the book. To pave the way for the development of the Kalman filter, we begin by solving the *recursive minimum mean-squared estimation problem* for the simple case of scalar random variables. For this solution, we use the *innovations approach* that exploits the correlation properties of a special stochastic process known as the *innovations process* (Kailath, 1968, 1970).

7.1 RECURSIVE MINIMUM MEAN-SQUARE ESTIMATION FOR SCALAR RANDOM VARIABLES

Let us assume that, based on a complete set of observed random variables $y(1), y(2), \ldots, y(n-1)$, starting with the first observation at time 1 and extending up to and including time $n-1$, we have found the minimum mean-square estimate $\hat{x}(n-1|\mathcal{Y}_{n-1})$ of a related zero-mean random variable $x(n-1)$. We are assuming that the observation at (or before) $n=0$ is zero. The space spanned by the observations $y(1), \ldots, y(n-1)$ is denoted by $\mathcal{Y}_{n-1}$. Suppose that we now have an additional observation $y(n)$ at time n, and the requirement is to compute an *updated* estimate $\hat{x}(n|\mathcal{Y}_n)$ of the related random variable $x(n)$, where $\mathcal{Y}_n$ denotes the space spanned by $y(1), \ldots, y(n)$. We may do this computation by storing the *past* observations, $y(1), y(2), \ldots, y(n-1)$, and then redoing the whole problem with the available data $y(1), y(2), \ldots, y(n-1), y(n)$, including the new observation. Computationally, however, it is much more efficient to use a *recursive estimation procedure*. In this procedure we *store* the previous estimate $\hat{x}(n-1|\mathcal{Y}_{n-1})$ and exploit it to compute the updated estimate $\hat{x}(n|\mathcal{Y}_n)$ in light of the new observation $y(n)$. There are several ways of developing the algorithm to do this recursive estimation. We will use the notion of *innovations* (Kailath, 1968, 1970), the origin of which may be traced back to Kolmogorov (1941).

Define the forward prediction error

$$f_{n-1}(n) = y(n) - \hat{y}(n|\mathcal{Y}_{n-1}), \quad n = 1, 2, \ldots \tag{7.1}$$

where $\hat{y}(n|\mathcal{Y}_{n-1})$ is the *one-step prediction* of the observed random variable $y(n)$ at time n, using *all* past observations available up to and including time $n-1$. The past observations used in this estimation are $y(1), y(2), \ldots, y(n-1)$, so the order of the prediction equals $n-1$. We may view $f_{n-1}(n)$ as the output of a forward prediction-error filter of order $n-1$, and with the filter input fed by the time series $y(1), y(2), \ldots, y(n)$. Note that the *prediction order* $n-1$ *increases linearly with* n. According to the principle of orthogonality, the prediction error $f_{n-1}(n)$ is orthogonal to all past observations $y(1), y(2), \ldots, y(n-1)$ and may therefore be regarded as a *measure* of the new information in the random variable $y(n)$ observed at time n, hence the name "innovation." The fact is that the observation $y(n)$ does not itself convey completely new information, since the predictable part, $\hat{y}(n|\mathcal{Y}_{n-1})$, is already completely determined by the past observations $y(1), y(2), \ldots, y(n-1)$. Rather, the part of the observation $y(n)$ that is really new is contained in the forward prediction error $f_{n-1}(n)$. We may therefore refer to this prediction error as the *innovation*, and for simplicity of notation write

$$\alpha(n) = f_{n-1}(n), \quad n = 1, 2, \ldots \tag{7.2}$$

The innovation $\alpha(n)$ has several important properties, as described here.

Property 1. *The innovation $\alpha(n)$, associated with the observed random variable $y(n)$, is orthogonal to the past observations $y(1), y(2), \ldots, y(n-1)$, as shown by*

$$E[\alpha(n)y^*(k)] = 0, \quad 1 \le k \le n-1 \tag{7.3}$$

This is simply a restatement of the principle of orthogonality.

Property 2. *The innovations* $\alpha(1), \alpha(2), \ldots, \alpha(n)$ *are orthogonal to each other, as shown by*

$$E[\alpha(n)\alpha^*(k)] = 0, \quad 1 \leq k \leq n-1 \tag{7.4}$$

This is a restatement of the fact that [see part (e) of Problem 20, Chapter 6]:

$$E[f_{n-1}(n)f^*_{k-1}(k)] = 0, \quad 1 \leq k \leq n-1$$

Equation (7.4), in effect, states that the innovation process $\alpha(n)$, described by Eqs. (7.1) and (7.2), is *white*.

Property 3. *There is a one-to-one correspondence between the observed data* $\{y(1), y(2), \ldots, y(n)\}$ *and the innovations* $\{\alpha(1), \alpha(2), \ldots, \alpha(n)\}$, *in that the one sequence may be obtained from the other by means of a causal and causally invertible filter without any loss of information. We may thus write*

$$\{y(1), y(2), \ldots, y(n)\} \rightleftharpoons \{\alpha(1), \alpha(2), \ldots, \alpha(n)\} \tag{7.5}$$

To prove this property, we use a form of the Gram–Schmidt orthogonalization procedure (described in Chapter 6). The procedure assumes that the observations $y(1), y(2), \ldots, y(n)$ are linearly independent in an algebraic sense. We first put

$$\alpha(1) = y(1) \tag{7.6}$$

where it is assumed that $\hat{y}(1|\mathcal{Y}_0)$ is zero. Next we put

$$\alpha(2) = y(2) + a_{1,1}y(1) \tag{7.7}$$

The coefficient $a_{1,1}$ is chosen such that the innovations $\alpha(1)$ and $\alpha(2)$ are orthogonal, as shown by

$$E[\alpha(2)\alpha^*(1)] = 0 \tag{7.8}$$

This requirement is satisfied by choosing

$$a_{1,1} = -\frac{E[y(2)y^*(1)]}{E[y(1)y^*(1)]} \tag{7.9}$$

Except for the minus sign, $a_{1,1}$ is a partial correlation coefficient in that it equals the cross-correlation between the observations $y(2)$ and $y(1)$, normalized with respect to the mean-square value of $y(1)$.

Next, we put

$$\alpha(3) = y(3) + a_{2,1}y(2) + a_{2,2}y(1) \tag{7.10}$$

where the coefficients $a_{2,1}$ and $a_{2,2}$ are chosen such that $\alpha(3)$ is orthogonal to both $\alpha(1)$ and $\alpha(2)$, and so on. Thus, in general, we may express the transformation of the observed data $y(1), y(2), \ldots, y(n)$ into the innovations $\alpha(1), \alpha(2), \ldots, \alpha(n)$ by writing

$$\begin{bmatrix} \alpha(1) \\ \alpha(2) \\ \cdot \\ \cdot \\ \cdot \\ \alpha(n) \end{bmatrix} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ a_{1,1} & 1 & \cdots & 0 \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ a_{n-1,n-1} & a_{n-1,n-2} & \cdots & 1 \end{bmatrix} \begin{bmatrix} y(1) \\ y(2) \\ \cdot \\ \cdot \\ \cdot \\ y(n) \end{bmatrix} \tag{7.11}$$

The nonzero elements of row k of the *lower triangular transformation matrix* on the right-hand side of Eq. (7.11) are deliberately denoted as $a_{k-1,k-1}, a_{k-1,k-2} \ldots, 1$, where $k = 1, 2, \ldots, n$. These elements represent the coefficients of a *forward prediction-error filter* of order $k - 1$. Note that $a_{k,0} = 1$ for all k. Accordingly, given the observed data $y(1), y(2), \ldots, y(n)$, we may compute the innovations $\alpha(1), \alpha(2), \ldots, \alpha(n)$. There is no loss of information in the course of this transformation, since we may recover the original observed data $y(1), y(2), \ldots, y(n)$ from the innovations $\alpha(1), \alpha(2), \ldots, \alpha(n)$. This we do by premultiplying both sides of Eq. (7.11) by the inverse of the lower triangular transformation matrix. This matrix is nonsingular, since its determinant equals 1 for all n. The transformation is therefore reversible.

Using Eq. (7.5), we may thus write

$$\hat{x}(n|\mathcal{Y}_n) = \text{minimum mean-square estimate of } x(n) \text{ given the observed data } y(1), y(2), \ldots, y(n)$$

or, equivalently,

$$\hat{x}(n|\mathcal{Y}_n) = \text{minimum mean-square estimate of } x(n) \text{ given the innovations } \alpha(1), \alpha(2), \ldots, \alpha(n)$$

Define the estimate $\hat{x}(n|\mathcal{Y}_n)$ as a linear combination of the innovations $\alpha(1), \alpha(2), \ldots, \alpha(n)$:

$$\hat{x}(n|\mathcal{Y}_n) = \sum_{k=1}^{n} b_k \alpha(k) \tag{7.12}$$

where the b_k are to be determined. With the innovations $\alpha(1), \alpha(2), \ldots, \alpha(n)$ orthogonal to each other, and the b_k chosen to minimize the mean-square value of the estimation error $x(n) - \hat{x}(n|\mathcal{Y}_n)$, we find that

$$b_k = \frac{E[x(n)\alpha^*(k)]}{E[\alpha(k)\alpha^*(k)]}, \quad 1 \le k \le n \tag{7.13}$$

We rewrite Eq. (7.12) in the form

$$\hat{x}(n|\mathcal{Y}_n) = \sum_{k=0}^{n-1} b_k \alpha(k) + b_n \alpha(n) \tag{7.14}$$

where

$$b_n = \frac{E[x(n)\alpha^*(n)]}{E[\alpha(n)\alpha^*(n)]} \tag{7.15}$$

However, by definition, the summation term on the right-hand side of Eq. (7.14) equals the previous estimate $\hat{x}(n-1|\mathcal{Y}_{n-1})$. We may thus express the recursive estimation algorithm that we are seeking as

$$\hat{x}(n|\mathcal{Y}_n) = \hat{x}(n-1|\mathcal{Y}_{n-1}) + b_n\alpha(n) \tag{7.16}$$

where b_n is defined by Eq. (7.15). Thus, by adding a *correction term* $b_n\alpha(n)$ to the previous estimate $\hat{x}(n-1|\mathcal{Y}_{n-1})$, with the correction being proportional to the innovation $\alpha(n)$, we get the updated estimate $\hat{x}(n|\mathcal{Y}_n)$.

The simple formulas of Eq. (7.15) and (7.16) are the basis of all recursive linear estimation schemes. Equipped with these simple and yet powerful ideas, we are now ready to study the more general Kalman filtering problem.

7.2 STATEMENT OF THE KALMAN FILTERING PROBLEM

Consider a *linear, discrete-time dynamical system* described by the signal-flow graph shown in Fig. 7.1. The time-domain description of the system presented here offers the following advantages (Gelb, 1974):

- Mathematical and notational convenience
- Close relationship to physical reality
- Useful basis for accounting for statistical behavior of the system

The notation of *state* plays a key role in this formulation. The *state vector*, denoted by $\mathbf{x}(n)$ in Fig. 7.1, is defined as any set of quantities that would be sufficient to uniquely describe the unforced dynamical behavior of the system. Typically, the state vector $\mathbf{x}(n)$, assumed to be of dimension M, is unknown. To estimate it, we use a set of observed data, denoted by the vector $\mathbf{y}(n)$ in Fig. 7.1. The *observation vector* $\mathbf{y}(n)$ is assumed to be of dimension N.

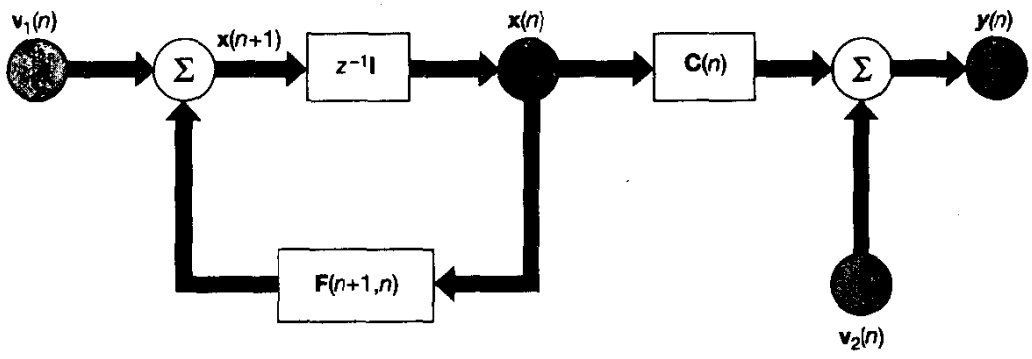


Figure 7.1 Signal-flow graph representation of a linear, discrete-time dynamical system.

In mathematical terms, the signal-flow graph of Fig. 7.1 embodies the following pair of equations:

1. A *process equation*

$$\mathbf{x}(n+1) = \mathbf{F}(n+1, n)\mathbf{x}(n) + \mathbf{v}_1(n) \tag{7.17}$$

where $\mathbf{F}(n+1, n)$ is a known M-by-M *state transition matrix* relating the state of the system at times $n+1$ and n. The M-by-1 vector $\mathbf{v}_1(n)$ represents *process noise*. The vector $\mathbf{v}_1(n)$ is modeled as a zero-mean, white-noise process whose correlation matrix is defined by

$$E[\mathbf{v}_1(n)\mathbf{v}_1^H(k)] = \begin{cases} \mathbf{Q}_1(n), & n = k \\ \mathbf{0}, & n \neq k \end{cases} \tag{7.18}$$

2. A *measurement equation*, describing the observation vector as

$$\mathbf{y}(n) = \mathbf{C}(n)\mathbf{x}(n) + \mathbf{v}_2(n) \tag{7.19}$$

where $\mathbf{C}(n)$ is a known N-by-M *measurement matrix*. The N-by-1 vector $\mathbf{v}_2(n)$ is called *measurement noise*. It is modeled as a zero-mean, white-noise process whose correlation matrix is

$$E[\mathbf{v}_2(n)\mathbf{v}_2^H(k)] = \begin{cases} \mathbf{Q}_2(n), & n = k \\ \mathbf{0}, & n \neq k \end{cases} \tag{7.20}$$

It is assumed that $\mathbf{x}(0)$, the initial value of the state, is uncorrelated with both $\mathbf{v}_1(n)$ and $\mathbf{v}_2(n)$ for $n \geq 0$. The noise vectors $\mathbf{v}_1(n)$ and $\mathbf{v}_2(n)$ are statistically independent, so we may write

$$E[\mathbf{v}_1(n)\,\mathbf{v}_2^H(k)] = \mathbf{0} \qquad \text{for all } n \text{ and } k \tag{7.21}$$

The Kalman filtering problem may now be formally stated as follows: Use the entire observed data, consisting of the vectors $\mathbf{y}(1), \mathbf{y}(2), \ldots, \mathbf{y}(n)$, to find for each $n \geq 1$ the minimum mean-square estimates of the components of the state $\mathbf{x}(i)$. The problem is called the *filtering* problem if $i = n$, the *prediction* problem if $i > n$, and the *smoothing* problem if $1 \leq i < n$. In this chapter we will only be concerned with the filtering and prediction problems, which are closely related. As remarked earlier in the introduction, we will solve the Kalman filtering problem by using the innovations approach (Kailath, 1968, 1970, 1981; Tretter, 1976).

7.3 THE INNOVATIONS PROCESS

Let the vector $\hat{\mathbf{y}}(n|\mathcal{Y}_{n-1})$ denote the minimum mean-square estimate of the observed data $\mathbf{y}(n)$ at time n, given all the past values of the observed data starting at time $n = 1$ and extending up to and including time $n - 1$. These past values are represented by the vec-

tors $\mathbf{y}(1), \mathbf{y}(2), \ldots, \mathbf{y}(n-1)$, which span the vector space $\mathcal{Y}_{n-1}$. We define the *innovations process* associated with $\mathbf{y}(n)$ as

$$\boldsymbol{\alpha}(n) = \mathbf{y}(n) - \hat{\mathbf{y}}(n|\mathcal{Y}_{n-1}), \quad n = 1, 2, \ldots \tag{7.22}$$

The M-by-1 vector $\boldsymbol{\alpha}(n)$ represents the new information in the observed data $\mathbf{y}(n)$.

Generalizing the results of Eqs. (7.3), (7.4) and (7.5), we find that the innovations process $\boldsymbol{\alpha}(n)$ has the following properties:

1. The innovations process $\boldsymbol{\alpha}(n)$, associated with the observed data $\mathbf{y}(n)$ at time n, is orthogonal to all past observations $\mathbf{y}(1), \mathbf{y}(2), \ldots, \mathbf{y}(n-1)$ as shown by

$$E[\boldsymbol{\alpha}(n)\mathbf{y}^H(k)] = \mathbf{0}, \quad 1 \leq k \leq n-1 \tag{7.23}$$

2. The innovations process consists of a sequence of vector random variables that are orthogonal to each other, as shown by

$$E[\boldsymbol{\alpha}(n)\boldsymbol{\alpha}^H(k)] = \mathbf{0}, \quad 1 \leq k \leq n-1 \tag{7.24}$$

3. There is a one-to-one correspondence between the sequence of vector random variables $\{\mathbf{y}(1), \mathbf{y}(2), \ldots, \mathbf{y}(n)\}$ representing the observed data and the sequence of vector random variables $\{\boldsymbol{\alpha}(1), \boldsymbol{\alpha}(2), \ldots, \boldsymbol{\alpha}(n)\}$ representing the innovations process, in that the one sequence may be obtained from the other by means of linear stable operators without loss of information. Thus, we may state that

$$\{\mathbf{y}(1), \mathbf{y}(2), \ldots, \mathbf{y}(n)\} \rightleftharpoons \{\boldsymbol{\alpha}(1), \boldsymbol{\alpha}(2), \ldots, \boldsymbol{\alpha}(n)\} \tag{7.25}$$

To form the sequence of vector random variables defining the innovations process, we may use a Gram–Schmidt orthogonalization procedure similar to that described in Section 7.1, except that the procedure is now formulated in terms of vectors and matrices (see Problem 1).

Correlation Matrix of the Innovations Process

To determine the correlation matrix of the innovations process $\boldsymbol{\alpha}(n)$, we first solve the state equation (7.17) recursively to obtain

$$\mathbf{x}(k) = \mathbf{F}(k, 0)\mathbf{x}(0) + \sum_{i=1}^{k-1} \mathbf{F}(k, i+1)\mathbf{v}_1(i) \tag{7.26}$$

where we have made use of the following assumptions and properties:

1. The initial value of the state vector $\mathbf{x}(0)$.
2. As previously assumed, the observed data [and therefore the noise vector $\mathbf{v}_1(n)$] are zero for $n \leq 0$.

3. The state transition matrix has the properties

$$\mathbf{F}(k, k-1)\mathbf{F}(k-1, k-2)\ldots\mathbf{F}(i+1, i) = \mathbf{F}(k, i)$$

and

$$\mathbf{F}(k, k) = \mathbf{I}$$

where $\mathbf{I}$ is the identity matrix. Note that for a time-invariant system we have

$$\mathbf{F}(n+1, n) = \mathbf{F}(n+1-n) = \mathbf{F}(1) = \text{constant}.$$

Equation (7.26) shows that $\mathbf{x}(k)$ is a linear combination of $\mathbf{x}(0)$ and $\mathbf{v}_1(1), \mathbf{v}_1(2), \ldots, \mathbf{v}_1(k-1)$.

By hypothesis, the measurement noise vector $\mathbf{v}_2(n)$ is uncorrelated with both the initial state vector $\mathbf{x}(0)$ and the process noise vector $\mathbf{v}_1(n)$. Accordingly, premultiplying both sides of Eq. (7.26) by $\mathbf{v}_2^H(n)$, and taking expectations, we deduce that

$$E[\mathbf{x}(k)\mathbf{v}_2^H(n)] = \mathbf{O}, \qquad k, n \leq 0 \tag{7.27}$$

Correspondingly, we deduce from the measurement equation (7.19) that

$$E[\mathbf{y}(k)\mathbf{v}_2^H(n)] = \mathbf{O}, \qquad 0 \leq k \leq n-1 \tag{7.28}$$

Moreover, we may write

$$E[\mathbf{y}(k)\mathbf{v}_1^H(n)] = \mathbf{O}, \qquad 0 \leq k \leq n \tag{7.29}$$

Given the past observations $\mathbf{y}(1), \ldots, \mathbf{y}(n-1)$ that span the space $\mathcal{Y}_{n-1}$, we also find from the measurement equation (7.19) that the minimum mean-square estimate of the present value $\mathbf{y}(n)$ of the observation vector equals

$$\hat{\mathbf{y}}(n|\mathcal{Y}_{n-1}) = \mathbf{C}(n)\hat{\mathbf{x}}(n|\mathcal{Y}_{n-1}) + \hat{\mathbf{v}}_2(n|\mathcal{Y}_{n-1})$$

However, the estimate $\hat{\mathbf{v}}_2(n|\mathcal{Y}_{n-1})$ of the measurement noise vector is zero since $\mathbf{v}_2(n)$ is orthogonal to the past observations $\mathbf{y}(1), \ldots, \mathbf{y}(n-1)$; see Eq. (7.28). Hence, we may simply write

$$\hat{\mathbf{y}}(n|\mathcal{Y}_{n-1}) = \mathbf{C}(n)\hat{\mathbf{x}}(n|\mathcal{Y}_{n-1}) \tag{7.30}$$

Therefore, using Eqs. (7.22) and (7.30), we may express the innovations process in the form

$$\boldsymbol{\alpha}(n) = \mathbf{y}(n) - \mathbf{C}(n)\hat{\mathbf{x}}(n|\mathcal{Y}_{n-1}) \tag{7.31}$$

Substituting the measurement equation (7.19) in (7.31), we get

$$\boldsymbol{\alpha}(n) = \mathbf{C}(n)\boldsymbol{\epsilon}(n, n-1) + \mathbf{v}_2(n) \tag{7.32}$$

where $\boldsymbol{\epsilon}(n, n-1)$ is the *predicted state-error vector* at time n, using data up to time

$n-1$. That is, $\boldsymbol{\epsilon}(n, n-1)$ is the difference between the state vector $\mathbf{x}(n)$ and the one-step prediction vector $\hat{\mathbf{x}}(n|\mathcal{Y}_{n-1})$, as shown by

$$\boldsymbol{\epsilon}(n, n-1) = \mathbf{x}(n) - \hat{\mathbf{x}}(n|\mathcal{Y}_{n-1}) \tag{7.33}$$

Note that the predicted state-error vector is orthogonal to both the process noise vector $\mathbf{v}_1(n)$ and the measurement noise vector $\mathbf{v}_2(n)$; see Problem 2.

The correlation matrix of the innovations process $\boldsymbol{\alpha}(n)$ is defined by

$$\mathbf{R}(n) = E[\boldsymbol{\alpha}(n)\boldsymbol{\alpha}^H(n)] \tag{7.34}$$

Therefore, substituting Eq. (7.32) in (7.34), expanding the pertinent terms, and then using the fact that the vectors $\boldsymbol{\epsilon}(n, n-1)$ and $\mathbf{v}_2(n)$ are orthogonal, we obtain the result:

$$\mathbf{R}(n) = \mathbf{C}(n)\mathbf{K}(n, n-1)\mathbf{C}^H(n) + \mathbf{Q}_2(n) \tag{7.35}$$

where $\mathbf{Q}_2(n)$ is the correlation matrix of the noise vector $\mathbf{v}_2(n)$. The M-by-M matrix $\mathbf{K}(n, n-1)$ is called the *predicted state-error correlation matrix*; it is defined by

$$\mathbf{K}(n, n-1) = E[\boldsymbol{\epsilon}(n, n-1)\boldsymbol{\epsilon}^H(n, n-1)] \tag{7.36}$$

where $\boldsymbol{\epsilon}(n, n-1)$ is the predicted state-error vector. The matrix $\mathbf{K}(n, n-1)$ is used as the statistical description of the error in the predicted estimate $\hat{\mathbf{x}}(n|\mathcal{Y}_{n-1})$.

7.4 ESTIMATION OF THE STATE USING THE INNOVATIONS PROCESS

Consider next the problem of deriving the minimum mean-square estimate of the state $\mathbf{x}(i)$ from the innovations process. From the discussion presented in Section 7.1, we deduce that this estimate may be expressed as a linear combination of the sequence of innovations processes $\boldsymbol{\alpha}(1), \boldsymbol{\alpha}(2), \ldots, \boldsymbol{\alpha}(n)$ [see Eq. (7.12) for comparison]:

$$\hat{\mathbf{x}}(i|\mathcal{Y}_n) = \sum_{k=1}^{n} \mathbf{B}_i(k)\boldsymbol{\alpha}(k) \tag{7.37}$$

where $\mathbf{B}_i(k)$, $k = 1, 2, \ldots, n$, is a set of M-by-N matrices to be determined. According to the principle of orthogonality, the predicted state-error vector is orthogonal to the innovation process, as shown by

$$\begin{aligned} E[\boldsymbol{\epsilon}(i, n)\boldsymbol{\alpha}^H(m)] &= E\{[\mathbf{x}(i) - \hat{\mathbf{x}}(i|\mathcal{Y}_n)]\boldsymbol{\alpha}^H(m)\} \\ &= \mathbf{O}, \qquad m = 1, 2, \ldots, n \end{aligned} \tag{7.38}$$

Substituting Eq. (7.37) in (7.38) and using the orthogonality property of the innovations process, namely, Eq. (7.24), we get

$$\begin{aligned} E[\mathbf{x}(i)\boldsymbol{\alpha}^H(m)] &= \mathbf{B}_i(m)E[\boldsymbol{\alpha}(m)\boldsymbol{\alpha}^H(m)] \\ &= \mathbf{B}_i(m)\mathbf{R}(m) \end{aligned} \tag{7.39}$$

Hence, postmultiplying both sides of Eq. (7.39) by the inverse matrix $\mathbf{R}^{-1}(m)$, we find that $\mathbf{B}_i(m)$ is given by

$$\mathbf{B}_i(m) = E[\mathbf{x}(i)\boldsymbol{\alpha}^H(m)]\,\mathbf{R}^{-1}(m) \tag{7.40}$$

Finally, substituting Eq. (7.40) in (7.37), we get the minimum mean-square estimate

$$\begin{aligned}\hat{\mathbf{x}}(i|\mathcal{Y}_n) &= \sum_{k=1}^{n} E[\mathbf{x}(i)\boldsymbol{\alpha}^H(k)]\,\mathbf{R}^{-1}(k)\boldsymbol{\alpha}(k)\\ &= \sum_{k=1}^{n-1} E[\mathbf{x}(i)\boldsymbol{\alpha}^H(k)]\,\mathbf{R}^{-1}(k)\,\boldsymbol{\alpha}(k)\\ &\quad + E[\mathbf{x}(i)\boldsymbol{\alpha}^H(n)]\,\mathbf{R}^{-1}(n)\boldsymbol{\alpha}(n)\end{aligned}$$

For $i = n + 1$, we may therefore write

$$\begin{aligned}\hat{\mathbf{x}}(n+1|\mathcal{Y}_n) &= \sum_{k=1}^{n-1} E[\mathbf{x}(n+1)\boldsymbol{\alpha}^H(k)]\,\mathbf{R}^{-1}(k)\boldsymbol{\alpha}(k)\\ &\quad + E[\mathbf{x}(n+1)\boldsymbol{\alpha}^H(n)]\,\mathbf{R}^{-1}(n)\boldsymbol{\alpha}(n)\end{aligned} \tag{7.41}$$

However, the state $\mathbf{x}(n + 1)$ at time $n + 1$ is related to the state $\mathbf{x}(n)$ at time n by Eq. (7.17). Therefore, using this relation, we may write for $0 \leq k \leq n$:

$$\begin{aligned}E[\mathbf{x}(n+1)\boldsymbol{\alpha}^H(k)] &= E\{[\mathbf{F}(n+1,n)\mathbf{x}(n) + \mathbf{v}_1(n)]\boldsymbol{\alpha}^H(k)\}\\ &= \mathbf{F}(n+1,n)E[\mathbf{x}(n)\boldsymbol{\alpha}^H(k)]\end{aligned} \tag{7.42}$$

where we have made use of the fact that $\boldsymbol{\alpha}(k)$ depends only on the observed data $\mathbf{y}(1), \ldots, \mathbf{y}(k)$, and therefore from Eq. (7.29) we see that $\mathbf{y}(n)$ and $\boldsymbol{\alpha}(k)$ are orthogonal for $0 \leq k \leq n$. We may thus rewrite the summation term on the right-hand side of Eq. (7.41) as follows:

$$\begin{aligned}\sum_{k=1}^{n-1} E[\mathbf{x}(n+1)\boldsymbol{\alpha}^H(k)]\,\mathbf{R}^{-1}(k)\boldsymbol{\alpha}(k) &= \mathbf{F}(n+1,n)\sum_{k=1}^{n-1} E[\mathbf{x}(n)\boldsymbol{\alpha}^H(k)]\,\mathbf{R}^{-1}(k)\boldsymbol{\alpha}(k)\\ &= \mathbf{F}(n+1,n)\hat{\mathbf{x}}(n|\mathcal{Y}_{n-1})\end{aligned} \tag{7.43}$$

To proceed further, we introduce some basic definitions, as described next.

Kalman Gain

Define the M-by-N matrix:

$$\mathbf{G}(n) = E[\mathbf{x}(n+1)\boldsymbol{\alpha}^H(n)]\,\mathbf{R}^{-1}(n) \tag{7.44}$$

Then, using this definition and the result of Eq. (7.43), we may rewrite Eq. (7.41) as follows:

$$\hat{\mathbf{x}}(n+1|\mathcal{Y}_n) = \mathbf{F}(n+1,n)\hat{\mathbf{x}}(n|\mathcal{Y}_{n-1}) + \mathbf{G}(n)\boldsymbol{\alpha}(n) \tag{7.45}$$

Equation (7.45) is of fundamental significance. It shows that we may compute the minimum mean-square estimate $\hat{\mathbf{x}}(n+1|\mathcal{Y}_n)$ of the state of a linear dynamical system by adding to the previous estimate $\hat{\mathbf{x}}(n|\mathcal{Y}_{n-1})$, which is premultiplied by the state transition matrix $\mathbf{F}(n+1, n)$, a correction term equal to $\mathbf{G}(n)\boldsymbol{\alpha}(n)$. The correction term equals the innovations process $\boldsymbol{\alpha}(n)$ premultiplied by the matrix $\mathbf{G}(n)$. Accordingly, and in recognition of the pioneering work by Kalman, the matrix $\mathbf{G}(n)$ is called the *Kalman gain*.

There now remains only the problem of expressing the Kalman gain $\mathbf{G}(n)$ in a form convenient for computation. To do this, we first use Eqs. (7.32) and (7.42) to express the expectation of the product of $\mathbf{x}(n+1)$ and $\boldsymbol{\alpha}^H(n)$ as follows:

$$\begin{aligned} E[\mathbf{x}(n+1)\boldsymbol{\alpha}^H(n)] &= \mathbf{F}(n+1, n)E[\mathbf{x}(n)\boldsymbol{\alpha}^H(n)] \\ &= \mathbf{F}(n+1, n)E[\mathbf{x}(n)(\mathbf{C}(n)\boldsymbol{\epsilon}(n, n-1) + \mathbf{v}_2(n))^H] \\ &= \mathbf{F}(n+1, n)E[\mathbf{x}(n)\boldsymbol{\epsilon}^H(n,n-1)]\mathbf{C}^H(n) \end{aligned} \tag{7.46}$$

where we have used the fact that the state $\mathbf{x}(n)$ and noise vector $\mathbf{v}_2(n)$ are uncorrelated [see Eq. (7.27)]. We further note that the predicted state-error vector $\boldsymbol{\epsilon}(n, n-1)$ is orthogonal to the estimate $\hat{\mathbf{x}}(n|\mathcal{Y}_{n-1})$. Therefore, the expectation of the product of $\mathbf{x}(n|\mathcal{Y}_{n-1})$ and $\boldsymbol{\epsilon}^H(n, n-1)$ is zero, and so we may rewrite Eq. (7.46) by replacing the multiplying factor $\mathbf{x}(n)$ by the predicted state-error vector $\boldsymbol{\epsilon}(n, n-1)$ as follows:

$$E[\mathbf{x}(n+1)\boldsymbol{\alpha}^H(n)] = \mathbf{F}(n+1, n)E[\boldsymbol{\epsilon}(n, n-1)\boldsymbol{\epsilon}^H(n, n-1)]\mathbf{C}^H(n) \tag{7.47}$$

From Eq. (7.36), we see that the expectation on the right-hand side of Eq. (7.47) equals the predicted state-error correlation matrix. Hence, we may rewrite Eq. (7.47) as follows:

$$E[\mathbf{x}(n+1)\boldsymbol{\alpha}^H(n)] = \mathbf{F}(n+1,n)\mathbf{K}(n, n-1)\mathbf{C}^H(n) \tag{7.48}$$

We may now redefine the Kalman gain. In particular, substituting Eq. (7.48) in (7.44), we get

$$\mathbf{G}(n) = \mathbf{F}(n+1, n)\mathbf{K}(n, n-1)\mathbf{C}^H(n)\mathbf{R}^{-1}(n) \tag{7.49}$$

where the correlation matrix $\mathbf{R}(n)$ is itself defined in Eq. (7.35).

The block diagram of Fig. 7.2 shows the signal-flow graph representation of Eq. (7.49) for computing the Kalman gain $\mathbf{G}(n)$. Having computed the Kalman gain $\mathbf{G}(n)$, we may then use Eq. (7.45) to update the one-step prediction, that is, to compute $\hat{\mathbf{x}}(n+1|\mathcal{Y}_n)$ given its old value $\hat{\mathbf{x}}(n|\mathcal{Y}_{n-1})$, as illustrated in Fig. 7.3. In this figure we have also used Eq. (7.31) for the innovations process $\boldsymbol{\alpha}(n)$.

Riccati Equation

As it stands, the formula of Eq. (7.49) is not particularly useful for computing the Kalman gain $\mathbf{G}(n)$, since it requires that the predicted state-error correlation matrix $\mathbf{K}(n, n-1)$ be known. To overcome this difficulty, we derive a formula for the recursive computation of $\mathbf{K}(n, n-1)$.

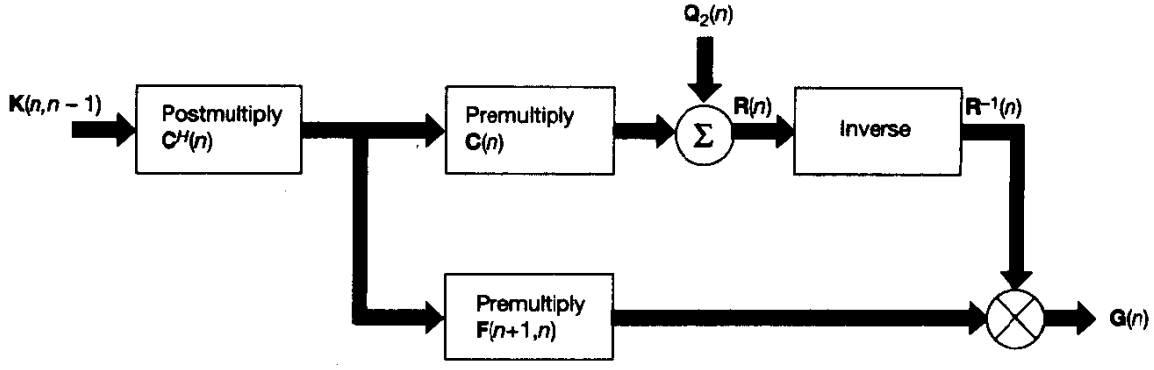


Figure 7.2 Kalman gain computer.

The predicted state-error vector $\boldsymbol{\epsilon}(n+1, n)$ equals the difference between the state $\mathbf{x}(n+1)$ and the one-step prediction $\hat{\mathbf{x}}(n+1|\mathcal{Y}_n)$[see Eq. (7.33)]:

$$\boldsymbol{\epsilon}(n+1, n) = \mathbf{x}(n+1) - \hat{\mathbf{x}}(n+1|\mathcal{Y}_n) \tag{7.50}$$

Substituting Eqs. (7.17) and (7.45) in (7.50), and using Eq. (7.31) for the innovations process $\boldsymbol{\alpha}(n)$, we get

$$\begin{aligned}\boldsymbol{\epsilon}(n+1, n) = \mathbf{F}(n+1, n)[\mathbf{x}(n) - \hat{\mathbf{x}}(n|\mathcal{Y}_{n-1})] \\ - \mathbf{G}(n)\,[\mathbf{y}(n) - \mathbf{C}(n)\hat{\mathbf{x}}(n|\mathcal{Y}_{n-1})] + \mathbf{v}_1(n)\end{aligned} \tag{7.51}$$

Next, using the measurement equation (7.19) to eliminate $\mathbf{y}(n)$ in Eq. (7.51), we get the following difference equation for recursive computation of the predicted state-error vector:

$$\begin{aligned}\boldsymbol{\epsilon}(n+1, n) = [\mathbf{F}(n+1, n) - \mathbf{G}(n)\mathbf{C}(n)]\,\boldsymbol{\epsilon}(n, n-1) \\ + \mathbf{v}_1(n) - \mathbf{G}(n)\,\mathbf{v}_2(n)\end{aligned} \tag{7.52}$$

The correlation matrix of the predicted state-error vector $\boldsymbol{\epsilon}(n+1, n)$ equals [see Eq. (7.36)]

$$\mathbf{K}(n+1, n) = E[\boldsymbol{\epsilon}(n+1, n)\boldsymbol{\epsilon}^H(n+1, n)] \tag{7.53}$$

Substituting Eq. (7.52) in (7.53), and recognizing that the error vector $\boldsymbol{\epsilon}(n, n-1)$ and the noise vectors $\mathbf{v}_1(n)$ and $\mathbf{v}_2(n)$ are mutually uncorrelated, we may express the predicted state-error correlation matrix as follows:

$$\begin{aligned}\mathbf{K}(n+1, n) = [\mathbf{F}(n+1, n) - \mathbf{G}(n)\mathbf{C}(n)]\mathbf{K}(n, n-1)\,[\mathbf{F}(n+1, n) - \mathbf{G}(n)\mathbf{C}(n)]^H \\ + \mathbf{Q}_1(n) + \mathbf{G}(n)\mathbf{Q}_2(n)\mathbf{G}^H(n)\end{aligned} \tag{7.54}$$

where $\mathbf{Q}_1(n)$ and $\mathbf{Q}_2(n)$ are the correlation matrices of $\mathbf{v}_1(n)$ and $\mathbf{v}_2(n)$, respectively. By expanding the right-hand side of Eq. (7.54), and then using Eqs. (7.49) and (7.35) for the

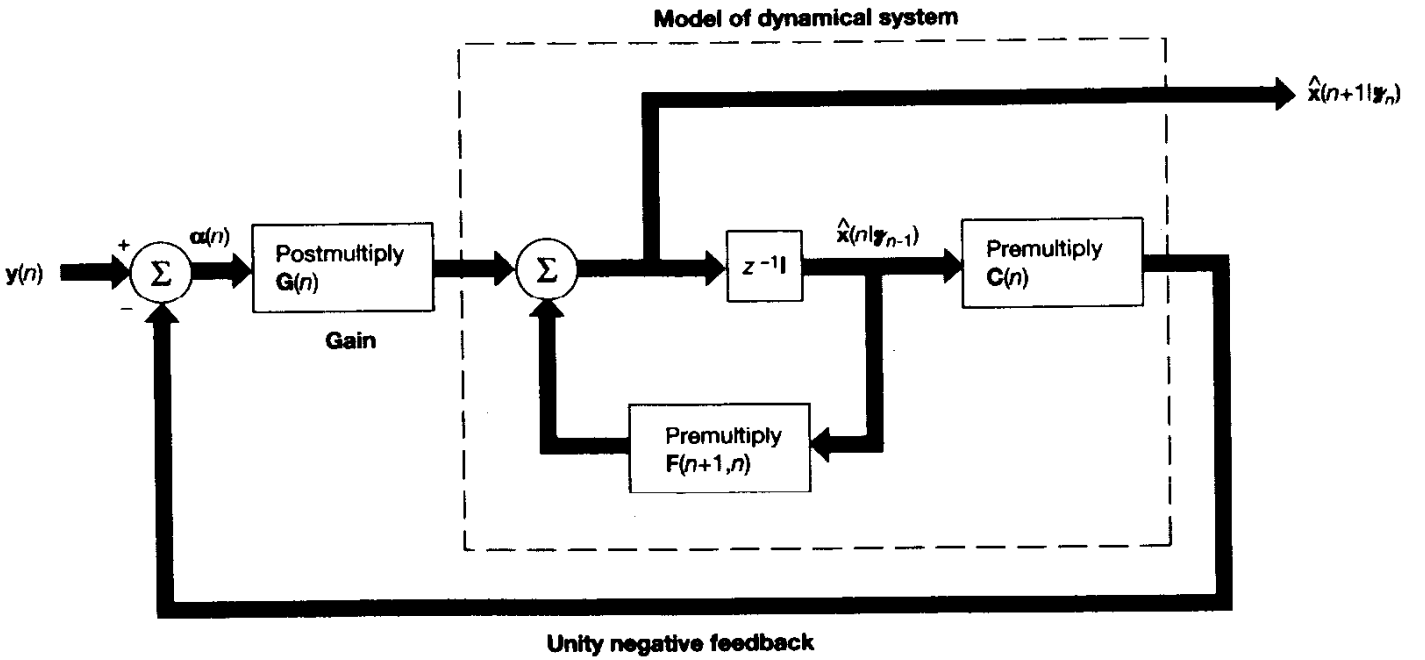


Figure 7.3 One-step predictor.

Kalman gain, we get the *Riccati difference equation*[1] for the recursive computation of the predicted state-error correlation matrix:

$$\mathbf{K}(n+1, n) = \mathbf{F}(n+1, n)\mathbf{K}(n)\mathbf{F}^H(n+1, n) + \mathbf{Q}_1(n) \tag{7.55}$$

The M-by-M matrix $\mathbf{K}(n)$ is described by the recursion:

$$\mathbf{K}(n) = \mathbf{K}(n, n-1) - \mathbf{F}(n, n+1)\mathbf{G}(n)\mathbf{C}(n)\mathbf{K}(n, n-1) \tag{7.56}$$

Here we have used the property

$$\mathbf{F}(n+1,n)\,\mathbf{F}(n, n+1) = \mathbf{I} \tag{7.57}$$

where $\mathbf{I}$ is the identity matrix. This property follows from the definition of the transition matrix. The mathematical significance of the matrix $\mathbf{K}(n)$ in Eq. (7.56) will be explained later in Section 7.5.

Figure 7.4 is a signal-flow graph representation of Eqs. (7.56) and (7.55), in that order. This diagram may be viewed as a representation of the *Riccati equation solver* in that, given $\mathbf{K}(n, n-1)$, it computes the updated value $\mathbf{K}(n+1, n)$.

Equations (7.49), (7.35), (7.31), (7.45), (7.56), and (7.55), in that order, define Kalman's one-step prediction algorithm.

Comments

The process applied to the input of the Kalman filter consists of the observed data $\mathbf{y}(1)$, $\mathbf{y}(2), \ldots, \mathbf{y}(n)$ that span the space $\mathcal{Y}_n$. The resulting filter output equals the predicted state vector $\hat{\mathbf{x}}(n+1|\mathcal{Y}_n)$. Given that the matrices $\mathbf{F}(n+1, n)$, $\mathbf{C}(n)$, $\mathbf{Q}_1(n)$, and $\mathbf{Q}_2(n)$ are all known quantities, we find from Eqs. (7.44), (7.55), and (7.56) that the predicted state-error correlation matrix $\mathbf{K}(n+1, n)$ is actually independent of the input $\mathbf{y}(n)$, which it has to be. The Kalman gain $\mathbf{G}(n)$ is also independent of the input $\mathbf{y}(n)$. Consequently, the predicted state-error correlation matrix $\mathbf{K}(n+1, n)$ and the Kalman gain $\mathbf{G}(n)$ may be computed before the Kalman filter is actually put into operation. With the correlation matrix $\mathbf{K}(n+1, n)$ providing a statistical description of the error in the predicted state vector $\hat{\mathbf{x}}(n+1|\mathcal{Y}(n))$, we may examine this matrix before actually using the Kalman filter to produce a realization of a physical system of interest; in this way, we may determine whether the solution supplied by the Kalman filter is indeed satisfactory.

As already mentioned, the Kalman filter theory assumes knowledge of the matrices $\mathbf{F}(n+1, n)$, $\mathbf{C}(n)$, $\mathbf{Q}_1(n)$ and $\mathbf{Q}_2(n)$. However, the theory may be *generalized* to include a situation where one or more of these matrices may assume values that depend on the input $\mathbf{y}(n)$. In such a situation, we find that although $\hat{\mathbf{x}}(n+1|\mathcal{Y}_n)$ and $\mathbf{K}(n+1, n)$ are still given by Eqs. (7.45) and (7.55), respectively, the Kalman gain $\mathbf{G}(n)$ and the predicted state-error correlation matrix $\mathbf{K}(n+1, n)$ are *not* precomputable (Anderson and Moore, 1979).

[1]The Riccati difference equation is named in honor of Count Jacopo Francisco Riccati. This equation has become of particular importance in control theory.

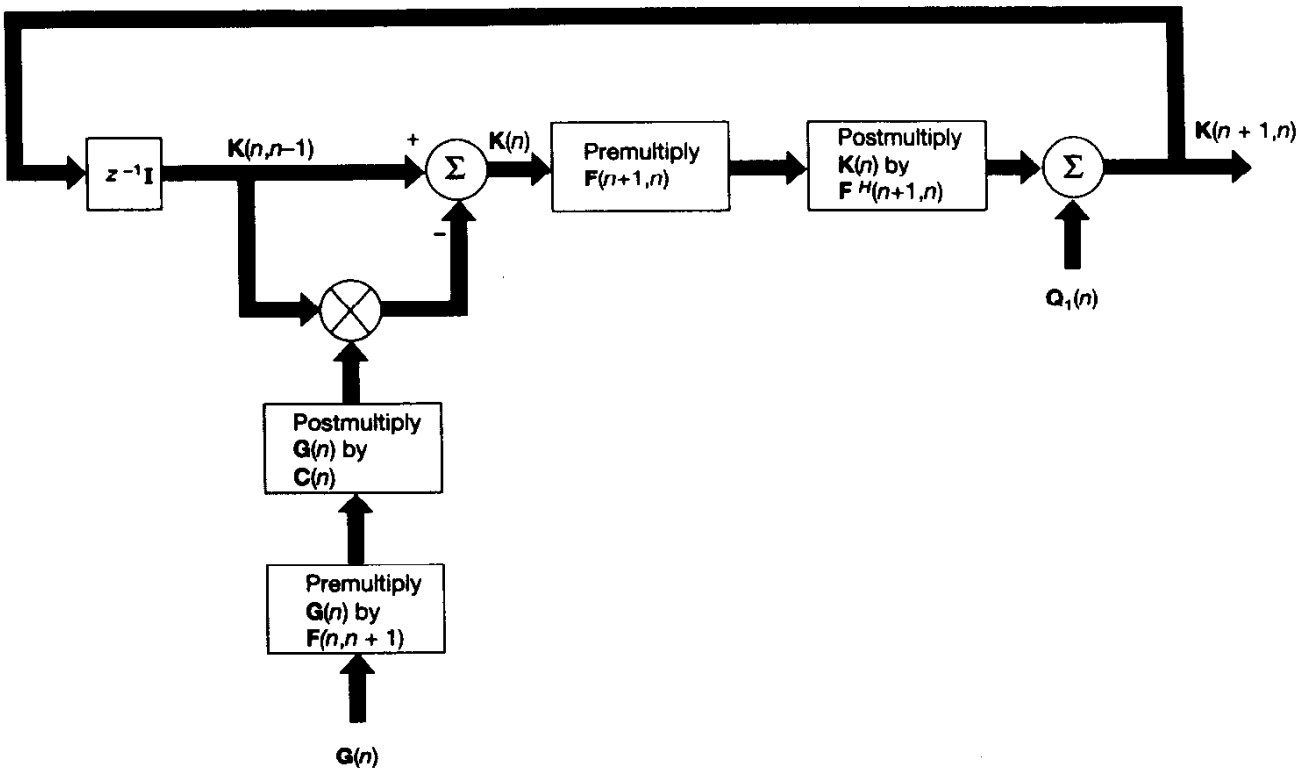


Figure 7.4 Riccati equation solver.

Rather, they both now depend on the input $\mathbf{y}(n)$. This means that $\mathbf{K}(n+1, n)$ is a *conditional* error-correlation matrix, conditional on the input $\mathbf{y}(n)$.

7.5 FILTERING

The next signal-processing operation we wish to consider is that of filtering. In particular, we wish to compute the *filtered estimate* $\hat{\mathbf{x}}(n|\mathcal{Y}_n)$ by using the one-step prediction algorithm described previously.

We first note that the state $\mathbf{x}(n)$ and the noise vector $\mathbf{v}_1(n)$ are independent of each other. Hence, from the state equation (7.17) we find that the minimum mean-square estimate of the state $\mathbf{x}(n+1)$ at time $n+1$, given the observed data up to and including time n [i.e., given $\mathbf{y}(1), \ldots, \mathbf{y}(n)$], equals

$$\hat{\mathbf{x}}(n+1|\mathcal{Y}_n) = \mathbf{F}(n+1, n)\hat{\mathbf{x}}(n|\mathcal{Y}_n) + \hat{\mathbf{v}}_1(n|\mathcal{Y}_n) \tag{7.58}$$

Since the noise vector $\mathbf{v}_1(n)$ is independent of the observed data $\mathbf{y}(1), \ldots, \mathbf{y}(n)$, it follows that the corresponding minimum mean-square estimate $\hat{\mathbf{v}}_1(n|\mathcal{Y}_n)$ is zero. Accordingly, Eq. (7.58) simplifies to

$$\hat{\mathbf{x}}(n+1|\mathcal{Y}_n) = \mathbf{F}(n+1, n)\,\hat{\mathbf{x}}(n|\mathcal{Y}_n) \tag{7.59}$$

To find the filtered estimate $\hat{\mathbf{x}}(n|\mathcal{Y}_n)$, we premultiply both sides of Eq. (7.59) by the inverse of the transition matrix $\mathbf{F}(n+1, n)$, and thus write

$$\hat{\mathbf{x}}(n|\mathcal{Y}_n) = \mathbf{F}^{-1}(n+1, n)\hat{\mathbf{x}}(n+1|\mathcal{Y}_n) \tag{7.60}$$

Using the property of the state transition matrix given in Eq. (7.57), we have

$$\mathbf{F}^{-1}(n+1, n) = \mathbf{F}(n, n+1) \tag{7.61}$$

We may therefore rewrite Eq. (7.60) in the equivalent form:

$$\hat{\mathbf{x}}(n|\mathcal{Y}_n) = \mathbf{F}(n, n+1)\,\hat{\mathbf{x}}(n+1|\mathcal{Y}_n) \tag{7.62}$$

This shows that knowing the solution to the one-step prediction problem, that is, the minimum mean-square estimate $\hat{\mathbf{x}}(n+1|\mathcal{Y}_n)$, we may determine the corresponding filtered estimate $\hat{\mathbf{x}}(n|\mathcal{Y}_n)$ simply by multiplying $\hat{\mathbf{x}}(n+1|\mathcal{Y}_n)$ by the state transition matrix $\mathbf{F}(n, n+1)$.

Filtered Estimation Error and Conversion Factor

In a filtering framework, it is natural that we define a *filtered estimation error vector* in terms of the filtered estimate of the state as follows:

$$\mathbf{e}(n) = \mathbf{y}(n) - \mathbf{C}(n)\hat{\mathbf{x}}(n|\mathcal{Y}_n) \tag{7.63}$$

This definition is similar to that of Eq. (7.31) for the innovations vector $\boldsymbol{\alpha}(n)$, except that we have substituted the filtered estimate $\hat{\mathbf{x}}(n|\mathcal{Y}_n)$ for the predicted estimate $\hat{\mathbf{x}}(n|\mathcal{Y}_{n-1})$. Using Eqs. (7.45) and (7.62) in (7.63), we get

$$\begin{aligned}\mathbf{e}(n) &= \mathbf{y}(n) - \mathbf{C}(n)\hat{\mathbf{x}}(n\,|\mathcal{Y}_{n-1}) - \mathbf{C}(n)\mathbf{F}(n, n+1)\mathbf{G}(n)\boldsymbol{\alpha}(n)\\ &= \boldsymbol{\alpha}(n) - \mathbf{C}(n)\mathbf{F}(n, n+1)\,\mathbf{G}(n)\boldsymbol{\alpha}(n)\\ &= [\mathbf{I} - \mathbf{C}(n)\mathbf{F}(n, n+1)\mathbf{G}(n)]\boldsymbol{\alpha}(n)\end{aligned} \tag{7.64}$$

The matrix-valued quantity inside the square brackets in Eq. (7.64) is called the *conversion factor*, which provides a formula for converting the innovations vector $\boldsymbol{\alpha}(n)$ into the filtered estimation error vector $\mathbf{e}(n)$. Using Eq. (7.49) to eliminate the Kalman gain $\mathbf{G}(n)$ from this definition and canceling common terms, we may rewrite Eq. (7.64) in the equivalent form:

$$\mathbf{e}(n) = \mathbf{Q}_2(n)\,\mathbf{R}^{-1}(n)\boldsymbol{\alpha}(n) \tag{7.65}$$

where $\mathbf{Q}_2(n)$ is the correlation matrix of the measurement noise process $\mathbf{v}_2(n)$, and the matrix $\mathbf{R}(n)$ is itself defined in Eq. (7.35) as the correlation matrix of the innovations process $\boldsymbol{\alpha}(n)$. Thus, except for a premultiplication by $\mathbf{Q}_2(n)$, Eq. (7.65) shows that the inverse matrix $\mathbf{R}^{-1}(n)$ plays the role of a conversion factor in the Kalman filter theory. Indeed, for the special case of $\mathbf{Q}_2(n)$ equal to the identity matrix, the inverse matrix $\mathbf{R}^{-1}$ is exactly the conversion factor defined herein.

Filtered State-Error Correlation Matrix

Earlier we introduced the M-by-M matrix $\mathbf{K}(n)$ in the formulation of the Riccati difference equation (7.55). We conclude our present discussion of the standard Kalman filter theory by showing that this matrix equals the correlation matrix of the error inherent in the filtered estimate $\hat{\mathbf{x}}(n|\mathcal{Y}_n)$.

Define the *filtered state-error vector* $\boldsymbol{\epsilon}(n)$ as the difference between the state $\mathbf{x}(n)$ and the filtered estimate $\hat{\mathbf{x}}(n|\mathcal{Y}_n)$, as shown by

$$\boldsymbol{\epsilon}(n) = \mathbf{x}(n) - \hat{\mathbf{x}}(n|\mathcal{Y}_n) \tag{7.66}$$

Substituting Eqs. (7.45) and (7.62) in (7.66), and recognizing that the product of $\mathbf{F}(n, n+1)$ and $\mathbf{F}(n+1, n)$ equals the identity matrix, we get

$$\begin{aligned}\boldsymbol{\epsilon}(n) &= \mathbf{x}(n) - \hat{\mathbf{x}}(n|\mathcal{Y}_{n-1}) - \mathbf{F}(n, n+1)\,\mathbf{G}(n)\,\boldsymbol{\alpha}(n)\\ &= \boldsymbol{\epsilon}(n, n-1) - \mathbf{F}(n, n+1)\mathbf{G}(n)\,\boldsymbol{\alpha}(n)\end{aligned} \tag{7.67}$$

where $\boldsymbol{\epsilon}(n, n, -1)$ is the predicted state-error vector at time n, using data up to time $n-1$, and $\boldsymbol{\alpha}(n)$ is the innovations process.

By definition, the correlation matrix of the filtered state-error vector $\boldsymbol{\epsilon}(n)$ equals the expectation $E[\boldsymbol{\epsilon}(n)\boldsymbol{\epsilon}^H(n)]$. Hence, using Eq. (7.67), we may express this expectation as follows:

$$E[\boldsymbol{\epsilon}(n)\boldsymbol{\epsilon}^H(n)] = E[\boldsymbol{\epsilon}(n, n-1)\,\boldsymbol{\epsilon}^H(n, n-1)] + \mathbf{F}(n, n+1)\,\mathbf{G}(n)\,E[\boldsymbol{\alpha}(n)\boldsymbol{\alpha}^H(n)]\,\mathbf{G}^H(n)\,\mathbf{F}^H(n, n+1) - 2E[\boldsymbol{\epsilon}(n, n-1)\boldsymbol{\alpha}^H(n)]\,\mathbf{G}^H(n)\,\mathbf{F}^H(n, n+1) \tag{7.68}$$

Examining the right-hand side of Eq. (7.68), we find that the three expectations contained in it may be interpreted individually as follows:

1. The first expectation equals the predicted state-error correlation matrix:

$$\mathbf{K}(n, n-1) = E[\boldsymbol{\epsilon}(n, n-1)\,\boldsymbol{\epsilon}^H(n, n-1)]$$

2. The expectation in the second term equals the correlation matrix of the innovations process $\boldsymbol{\alpha}(n)$:

$$\mathbf{R}(n) = E[\boldsymbol{\alpha}(n)\boldsymbol{\alpha}^H(n)]$$

3. The expectation in the third term may be expressed as follows:

$$E[\boldsymbol{\epsilon}(n, n-1)\,\boldsymbol{\alpha}^H(n)] = E[(\mathbf{x}(n) - \hat{\mathbf{x}}(n|\mathcal{Y}_{n-1}))\boldsymbol{\alpha}^H(n)] = E[\mathbf{x}(n)\,\boldsymbol{\alpha}^H(n)]$$

where, in the last line, we have used the fact that the estimate $\hat{\mathbf{x}}(n|\mathcal{Y}_{n-1})$ is orthogonal to the innovations process $\boldsymbol{\alpha}(n)$ acting as input. Next, from Eq. (7.42) we see, by putting $k = n$ and then premultiplying both sides by the inverse matrix $\mathbf{F}^{-1}(n+1, n) = \mathbf{F}(n, n+1)$, that

$$E[\mathbf{x}(n)\,\boldsymbol{\alpha}^H(n)] = \mathbf{F}(n, n+1)\,E[\mathbf{x}(n+1)\boldsymbol{\alpha}^H(n)] = \mathbf{F}(n, n+1)\mathbf{G}(n)\,\mathbf{R}(n)$$

where, in the last line, we have made use of Eq. (7.44). Hence,

$$E[\boldsymbol{\epsilon}(n, n-1)\,\boldsymbol{\alpha}^H(n)] = \mathbf{F}(n, n+1)\,\mathbf{G}(n)\,\mathbf{R}(n)$$

We may now use these results in Eq. (7.68), and so obtain

$$E[\boldsymbol{\epsilon}(n)\boldsymbol{\epsilon}^H(n)] = \mathbf{K}(n, n-1) - \mathbf{F}(n, n+1)\,\mathbf{G}(n)\,\mathbf{R}(n)\mathbf{G}^H(n)\,\mathbf{F}^H(n, n+1) \tag{7.69}$$

We may further simplify this result by noting that [see Eq. (7.49)]

$$\mathbf{G}(n)\,\mathbf{R}(n) = \mathbf{F}(n+1, n)\,\mathbf{K}(n, n-1)\,\mathbf{C}^H(n) \tag{7.70}$$

Accordingly, using Eqs. (7.69) and (7.70), and recognizing that the product of $\mathbf{F}(n, n+1)$ and $\mathbf{F}(n+1, n)$ equals the identity matrix, we get the desired result for the filtered state-error correlation matrix:

$$E[\boldsymbol{\epsilon}(n)\boldsymbol{\epsilon}^H(n)] = \mathbf{K}(n, n-1) - \mathbf{K}(n, n-1)\mathbf{C}^H(n)\,\mathbf{G}^H(n)\mathbf{F}^H(n, n+1)$$

Equivalently, using the Hermitian property of $E[\boldsymbol{\epsilon}(n)\boldsymbol{\epsilon}^H(n)]$ and that of $\mathbf{K}(n, n-1)$, we may write

$$E[\boldsymbol{\epsilon}(n)\boldsymbol{\epsilon}^H(n)] = \mathbf{K}(n, n-1) - \mathbf{F}(n, n+1)\mathbf{G}(n)\,\mathbf{C}(n)\,\mathbf{K}(n, n-1) \tag{7.71}$$

Comparing Eq. (7.71) with (7.56), we readily see that

$$E[\boldsymbol{\epsilon}(n)\,\boldsymbol{\epsilon}^H(n)] = \mathbf{K}(n)$$

This shows that the matrix $\mathbf{K}(n)$ used in the Riccati difference equation (7.55) is in fact the *filtered state-error correlation matrix*. The matrix $\mathbf{K}(n)$ is used as the statistical description of the error in the filtered estimate $\hat{\mathbf{x}}(n|\mathcal{Y}_n)$.

7.6 INITIAL CONDITIONS

To operate the one-step prediction and filtering algorithms described in Sections 7.4 and 7.5, we obviously need to specify the *initial conditions*. We now address this issue.

The initial state of the process equation (7.17) is not known precisely. Rather, it is usually described by its mean and correlation matrix. In the absence of any observed data at time $n = 0$, we may choose the *initial predicted estimate* as

$$\hat{\mathbf{x}}(1|\mathcal{Y}_0) = E[\mathbf{x}(1)] \tag{7.72}$$

and its *correlation matrix*

$$\begin{aligned}\mathbf{K}(1,0) &= E[(\mathbf{x}(1) - E[\mathbf{x}(1)])(\mathbf{x}(1) - E[\mathbf{x}(1)])^H] \\ &= \boldsymbol{\Pi}_0\end{aligned} \tag{7.73}$$

This choice for the initial conditions is not only intuitively satisfying but also has the advantage of yielding a filtered estimate of the state $\hat{\mathbf{x}}(n|\mathcal{Y}_n)$ that is *unbiased* (see Problem 10). Assuming that the state vector $\mathbf{x}(n)$ has *zero mean*, we may simplify Eqs. (7.72) and (7.73) by setting

$$\hat{\mathbf{x}}(1|\mathcal{Y}_0) = \mathbf{0}$$

and

$$\mathbf{K}(1,0) = E[\mathbf{x}(1)\,\mathbf{x}^H(1)] = \boldsymbol{\Pi}_0$$

7.7 SUMMARY OF THE KALMAN FILTER

Table 7.1 presents a summary of the variables used to formulate the solution to the Kalman filtering problem. The input of the filter is the vector process $\mathbf{y}(n)$, represented by the vector space $\mathcal{Y}_n$, and the output is the filtered estimate $\hat{\mathbf{x}}(n|\mathcal{Y}_n)$ of the state vector. In Table 7.2, we present a summary of the Kalman filter (including initial conditions) based on the one-step prediction algorithm.

TABLE 7.1 SUMMARY OF THE KALMAN VARIABLES

Variable	Definition	Dimension
$\mathbf{x}(n)$	State vector at time n	M-by-1
$\mathbf{y}(n)$	Observation vector at time n	N-by-1
$\mathbf{F}(n+1, n)$	State transition matrix from time n to $n+1$	M-by-M
$\mathbf{C}(n)$	Measurement matrix at time n	N-by-M
$\mathbf{Q}_1(n)$	Correlation matrix of process noise vector $\mathbf{v}_1(n)$	M-by-M
$\mathbf{Q}_2(n)$	Correlation matrix of measurement noise vector $\mathbf{v}_2(n)$	N-by-N
$\hat{\mathbf{x}}(n+1\|\mathcal{Y}_n)$	Predicted estimate of the state vector at time $n+1$, given the observation vectors $\mathbf{y}(1), \mathbf{y}(2), \ldots, \mathbf{y}(n)$	M-by-1
$\hat{\mathbf{x}}(n\|\mathcal{Y}_n)$	Filtered estimate of the state vector at time n, given the observation vectors $\mathbf{y}(1), \mathbf{y}(2), \ldots, \mathbf{y}(n)$	M-by-1
$\mathbf{G}(n)$	Kalman gain at time n	M-by-N
$\boldsymbol{\alpha}(n)$	Innovations vector at time n	N-by-1
$\mathbf{R}(n)$	Correlation matrix of the innovations vector $\boldsymbol{\alpha}(n)$	N-by-N
$\mathbf{K}(n+1, n)$	Correlation matrix of the error in $\hat{\mathbf{x}}(n+1\|\mathcal{Y}_n)$	M-by-M
$\mathbf{K}(n)$	Correlation matrix of the error in $\hat{\mathbf{x}}(n\|\mathcal{Y}_n)$	M-by-M

TABLE 7.2 SUMMARY OF THE KALMAN FILTER BASED ON ONE-STEP PREDICTION

Input vector process

Observations = $\{\mathbf{y}(1), \mathbf{y}(2), \ldots, \mathbf{y}(n)\}$

Known parameters

State transition matrix = $\mathbf{F}(n+1, n)$

Measurement matrix = $\mathbf{C}(n)$

Correlation matrix of process noise vector = $\mathbf{Q}_1(n)$

Correlation matrix of measurement noise vector = $\mathbf{Q}_2(n)$

Computation: $n = 1, 2, 3, \ldots$

$$\mathbf{G}(n) = \mathbf{F}(n+1, n)\mathbf{K}(n, n-1)\mathbf{C}^H(n)[\mathbf{C}(n)\mathbf{K}(n, n-1)\mathbf{C}^H(n) + \mathbf{Q}_2(n)]^{-1}$$

$$\boldsymbol{\alpha}(n) = \mathbf{y}(n) - \mathbf{C}(n)\hat{\mathbf{x}}(n|\mathcal{Y}_{n-1})$$

$$\hat{\mathbf{x}}(n+1|\mathcal{Y}_n) = \mathbf{F}(n+1, n)\hat{\mathbf{x}}(n|\mathcal{Y}_{n-1}) + \mathbf{G}(n)\boldsymbol{\alpha}(n)$$

$$\mathbf{K}(n) = \mathbf{K}(n, n-1) - \mathbf{F}(n, n+1)\mathbf{G}(n)\mathbf{C}(n)\mathbf{K}(n, n-1)$$

$$\mathbf{K}(n+1, n) = \mathbf{F}(n+1, n)\mathbf{K}(n)\mathbf{F}^H(n+1, n) + \mathbf{Q}_1(n)$$

Initial conditions:

$$\hat{\mathbf{x}}(1|\mathcal{Y}_0) = E[\mathbf{x}(1)]$$

$$\mathbf{K}(1, 0) = E[(\mathbf{x}(1) - E[\mathbf{x}(1)])(\mathbf{x}(1) - E[\mathbf{x}(1)])^H] = \Pi_0$$

A block diagram representation of the Kalman filter is given in Fig. 7.5, which is based on three functional blocks:

- Kalman gain computer, described in Fig. 7.2
- One-step predictor, described in Fig. 7.3.
- Riccati equation solver, described in Fig. 7.4

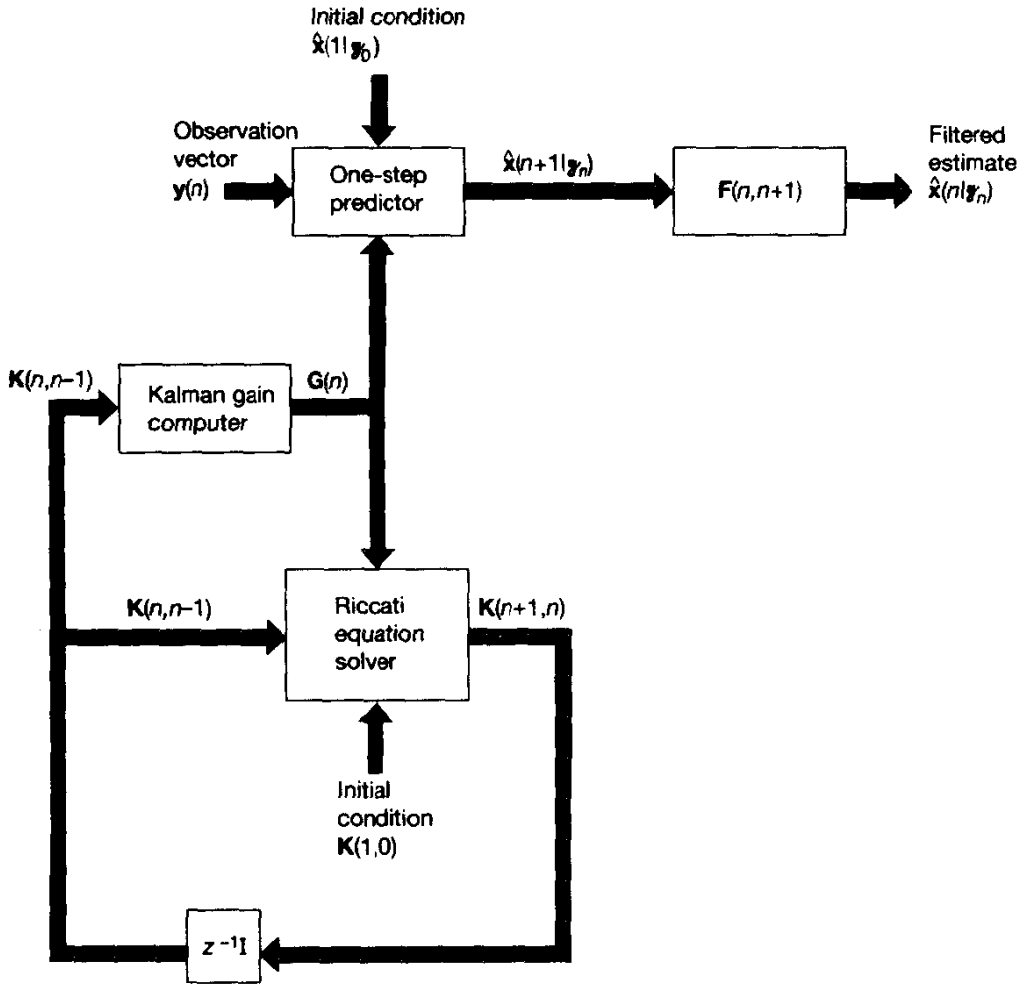


Figure 7.5 Block diagram of the Kalman filter based on one-step prediction.

7.8 VARIANTS OF THE KALMAN FILTER

As mentioned in the introductory remarks to this chapter, the main reason for our interest in Kalman filter theory in this book is that it provides a general framework for the derivation of certain adaptive filtering algorithms known collectively as the family of recursive least-squares (RLS) algorithms.

The application of Kalman filter theory to adaptive filtering was apparently first reported in the literature by Lawrence and Kaufman (1971); see Problem 8. This was followed by Godard (1974), who used an approach different from that of Lawrence and Kaufman. In particular, Godard formulated the adaptive filtering problem (using a tap-delay-line structure) as the estimation of a state vector in Gaussian noise, which represents

a classical Kalman filtering problem. Godard's paper prompted many other investigators to explore the application of Kalman filter theory to adaptive filtering problems.

However, we had to await the paper by Sayed and Kailath (1994) to discover how indeed the Riccati-based Kalman filtering algorithm and its variants can be correctly framed into one-to-one correspondences with all the known algorithms in the RLS family. We will take up the details of this unifying framework later in the book. For now, we will focus our attention on a special dynamical model that befits our future needs.

Special Case: Unforced Dynamics

Consider a linear dynamical system whose state-space model is described by the following pair of state equations (Sayed and Kailath, 1994):

$$\mathbf{x}(n+1) = \lambda^{-1/2}\mathbf{x}(n) \tag{7.74}$$

$$y(n) = \mathbf{u}^H(n)\mathbf{x}(n) + v(n) \tag{7.75}$$

where λ is a positive real scalar. According to this model, the process noise is zero, and the measurement noise, denoted by the scalar $v(n)$, is a zero-mean white noise process with unit variance, as shown by

$$E[v(n)v^*(k)] = \begin{cases} 1, & n = k \\ 0, & n \neq k \end{cases} \tag{7.76}$$

Thus, comparing this model with the general model described by Eqs. (7.17) to (7.21), we note the following:

$$\mathbf{F}(n+1, n) = \lambda^{-1/2}\mathbf{I} \tag{7.77}$$

$$\mathbf{Q}_1(n) = \mathbf{O} \tag{7.78}$$

$$\mathbf{C}(n) = \mathbf{u}^H(n) \tag{7.79}$$

$$\mathbf{Q}_2(n) = 1 \tag{7.80}$$

The state-space model described by Eqs. (7.74) to (7.76) is referred to as an *unforced dynamical model* by virtue of the fact that the process equation (7.74) is free of an external force. Most importantly, the state transition matrix of the model is equal to the identity matrix $\mathbf{I}$ scaled by the constant $\lambda^{-1/2}$. Consequently, the predicted state-error correlation matrix $\mathbf{K}(n+1, n)$ and the filtered state-error correlation matrix $\mathbf{K}(n)$ assume a common value; see Problem 9.

This special unforced dynamical model holds the key to the formulation of a general framework for deriving the RLS family of adaptive filtering algorithms. As we shall see later in the book, the constant λ has a significant role in the operation of these algorithms. For now we content ourselves by considering variants of the Kalman filtering algorithm based on this model.

TABLE 7.3 SUMMARY OF THE COVARIANCE (KALMAN) FILTERING ALGORITHM FOR THE SPECIAL UNFORCED DYNAMICAL MODEL

Input scalar process:
Observations: $y(1), y(2), \ldots y(n)$
Known parameters:
state transition matrix $= \lambda^{-1/2}\mathbf{I}$, $\mathbf{I}$ = identity matrix
measurement matrix $= \mathbf{u}^H(n)$
variance of measurement noise $v(n) = 1$
Initial conditions:

$$\hat{\mathbf{x}}(1\,|\mathcal{Y}_0) = E[\mathbf{x}(1)]$$

$$\mathbf{K}(1,0) = E[(\mathbf{x}(1) - E[\mathbf{x}(1)])(\mathbf{x}(1) - E[\mathbf{x}(1)])^H] = \Pi_0$$

Computation: $n = 1, 2, 3, \ldots$

$$\mathbf{g}(n) = \frac{\lambda^{-1/2}\mathbf{K}(n-1)\mathbf{u}(n)}{\mathbf{u}^H(n)\mathbf{K}(n-1)\mathbf{u}(n) + 1}$$

$$\alpha(n) = y(n) - \mathbf{u}^H(n)\hat{\mathbf{x}}(n\,|\mathcal{Y}_{n-1})$$

$$\hat{\mathbf{x}}(n+1\,|\mathcal{Y}_n) = \lambda^{-1/2}\hat{\mathbf{x}}(n\,|\mathcal{Y}_{n-1}) + \mathbf{g}(n)\alpha(n)$$

$$\mathbf{K}(n) = \lambda^{-1}\mathbf{K}(n-1) - \lambda^{-1/2}\mathbf{g}(n)\mathbf{u}^H(n)\mathbf{K}(n-1)$$

Covariance (Kalman) Filtering Algorithm

The Kalman filtering algorithm summarized in Table 7.2 is designed to propagate the correlation (covariance) matrix $\mathbf{K}(n+1, n)$ that refers to the error in the state's estimate $\hat{\mathbf{x}}(n+1|\mathcal{Y}_n)$. This algorithm is therefore commonly referred to as the *covariance (Kalman) filtering algorithm*. For the unforced dynamical model at hand, we find that substituting Eqs. (7.77) to (7.80) in Table 7.2 yields the simplified covariance filtering algorithm summarized in Table 7.3. In this table we have used $\mathbf{g}(n)$ to denote the Kalman gain, as it takes the form of a vector here.

Information Filtering Algorithm

The Kalman filter may also be implemented by propagating the inverse matrix $\mathbf{K}^{-1}(n)$ which accentuates the recursive least-squares nature of the filtering process. The *inverse state-error correlation matrix*, $\mathbf{K}^{-1}(n)$, is related to *Fisher's information matrix*[2], which permits an interpretation of filter performance in information-theoretic terms. For this reason, an implementation of the Kalman filtering algorithm based on $\mathbf{K}^{-1}(n)$ is termed the *information filtering algorithm* (Fraser, 1967).

For the derivation of the information filtering algorithm, we may proceed in the manner described next.

Step 1. We start with the Riccati difference equation which, for the special unforced dynamical model, has the form (see the last line of the algorithm in Table 7.3):

[2]Fisher's information matrix is discussed in Appendix D.

$$\mathbf{K}(n) = \lambda^{-1}\mathbf{K}(n-1) - \lambda^{-1/2}\mathbf{g}(n)\mathbf{u}^H(n)\mathbf{K}(n-1) \tag{7.81}$$

Solving this equation for the matrix product $\mathbf{g}(n)\mathbf{u}^H(n)\mathbf{K}(n-1)$, we get

$$\mathbf{g}(n)\mathbf{u}^H(n)\mathbf{K}(n-1) = \lambda^{-1/2}\mathbf{K}(n-1) - \lambda^{1/2}\mathbf{K}(n) \tag{7.82}$$

Next, from the first line of the algorithm in Table 7.3, the Kalman gain for the unforced dynamical model of interest is defined by

$$\mathbf{g}(n) = \frac{\lambda^{-1/2}\mathbf{K}(n-1)\mathbf{u}(n)}{\mathbf{u}^H(n)\mathbf{K}(n-1)\mathbf{u}(n) + 1} \tag{7.83}$$

Cross-multiplying and rearranging terms, we may rewrite Eq. (7.83) as

$$\mathbf{g}(n) = \lambda^{-1/2}\mathbf{K}(n-1)\mathbf{u}(n) - (\mathbf{g}(n)\mathbf{u}^H(n)\mathbf{K}(n-1))\mathbf{u}(n) \tag{7.84}$$

Substituting Eq. (7.82) in (7.84), and then canceling common terms, we get a new definition for the Kalman gain:

$$\mathbf{g}(n) = \lambda^{1/2}\mathbf{K}(n)\mathbf{u}(n) \tag{7.85}$$

Next, eliminating $\mathbf{g}(n)$ between Eqs. (7.82) and (7.85), and multiplying the result by $\lambda^{1/2}$, we get

$$\mathbf{K}(n-1) = \lambda\mathbf{K}(n)\mathbf{u}(n)\mathbf{u}^H(n)\mathbf{K}(n-1) + \lambda\mathbf{K}(n) \tag{7.86}$$

Premultiplying Eq. (7.86) by the inverse matrix $\mathbf{K}^{-1}(n)$ and postmultiplying it by $\mathbf{K}^{-1}(n-1)$, we get the first recursion of the information filtering algorithm:

$$\mathbf{K}^{-1}(n) = \lambda\mathbf{K}^{-1}(n-1) + \lambda\mathbf{u}(n)\mathbf{u}^H(n) \tag{7.87}$$

Step 2. From the second and third lines of the algorithm summarized in Table 7.3, we have, respectively,

$$\alpha(n) = y(n) - \mathbf{u}^H(n)\hat{\mathbf{x}}(n \mid \mathcal{Y}_{n-1}) \tag{7.88}$$

and

$$\hat{\mathbf{x}}(n+1 \mid \mathcal{Y}_n) = \lambda^{-1/2}\hat{\mathbf{x}}(n \mid \mathcal{Y}_{n-1}) + \mathbf{g}(n)\alpha(n) \tag{7.89}$$

Therefore, substituting Eq. (7.85) in (7.89), we get

$$\hat{\mathbf{x}}(n+1 \mid \mathcal{Y}_n) = \lambda^{-1/2}\hat{\mathbf{x}}(n \mid \mathcal{Y}_{n-1}) + \lambda^{1/2}\mathbf{K}(n)\mathbf{u}(n)\alpha(n) \tag{7.90}$$

Next, eliminating $\alpha(n)$ between Eqs. (7.88) and (7.90) yields

$$\hat{\mathbf{x}}(n+1 \mid \mathcal{Y}_n) = [\lambda^{-1/2}\mathbf{I} - \lambda^{1/2}\mathbf{K}(n)\mathbf{u}(n)\mathbf{u}^H(n)]\hat{\mathbf{x}}(n \mid \mathcal{Y}_{n-1}) + \lambda^{1/2}\mathbf{K}(n)\mathbf{u}(n)y(n) \tag{7.91}$$

But, from Eq. (7.86), we readily deduce the following relation:

$$\lambda^{-1/2}\mathbf{I} - \lambda^{1/2}\mathbf{K}(n)\mathbf{u}(n)\mathbf{u}^H(n) = \lambda^{1/2}\mathbf{K}(n)\mathbf{K}^{-1}(n-1) \tag{7.92}$$

Accordingly, we may simplify Eq. (7.91) as follows:

$$\hat{\mathbf{x}}(n+1 \mid \mathcal{Y}_n) = \lambda^{1/2}\mathbf{K}(n)\mathbf{K}^{-1}(n-1)\hat{\mathbf{x}}(n \mid \mathcal{Y}_{n-1}) + \lambda^{1/2}\mathbf{K}(n)\mathbf{u}(n)y(n)$$

Premultiplying this equation by the inverse matrix $\mathbf{K}^{-1}(n)$, we get the second recursion of the information filtering algorithm:

$$\mathbf{K}^{-1}(n)\hat{\mathbf{x}}(n+1 \mid \mathcal{Y}_n) = \lambda^{1/2}[\mathbf{K}^{-1}(n-1)\hat{\mathbf{x}}(n \mid \mathcal{Y}_{n-1}) + \mathbf{u}(n)y(n)] \tag{7.93}$$

Note that in Eq. (7.93) the algorithm propagates the product $\mathbf{K}^{-1}(n-1)\hat{\mathbf{x}}(n|\mathcal{Y}_{n-1})$ rather than the estimate $\hat{\mathbf{x}}(n|\mathcal{Y}_{n-1})$ by itself.

Step 3. Finally, the updated value of the state's estimate is computed by combining the results of steps 1 and 2 as follows:

$$\begin{aligned}\hat{\mathbf{x}}(n+1|\mathcal{Y}_n) &= \mathbf{K}(n)\,(\mathbf{K}^{-1}(n)\hat{\mathbf{x}}(n+1|\mathcal{Y}_n)) \\ &= [\mathbf{K}^{-1}(n)]^{-1}(\mathbf{K}^{-1}(n)\hat{\mathbf{x}}(n+1|\mathcal{Y}_n))\end{aligned} \tag{7.94}$$

Equations (7.87), (7.93), and (7.94), in that order, constitute the information-filtering algorithm for the unforced dynamical model of Eqs. (7.74) to (7.76). A summary of the algorithm is presented in Table 7.4.

Although the covariance and information implementations of the Kalman filter, as described herein, are algebraically equivalent, the numerical properties of these two algorithms may differ substantially from each other (Kaminski et al., 1971). However, both algorithms require the same number of algebraic operations (i.e., multiplications and additions), which, for the special model at hand, is $O(M^2)$, where M is the state dimension.

Square-root Filtering

The covariance implementation of the Kalman filter, summarized in Table 7.2, is the optimal solution to the linear filtering problem posed in Section 7.2. However, this algorithm is prone to serious numerical difficulties that are well documented in the literature (Kamin-

TABLE 7.4 SUMMARY OF THE INFORMATION-FILTERING ALGORITHM FOR THE SPECIAL UNFORCED DYNAMICAL MODEL

Input scalar process:
observations = $y(1), y(2), \ldots, y(n)$
Known parameters:
state transition matrix $= \lambda^{-1/2}\mathbf{I}$, $\mathbf{I}$ = identity matrix
measurement matrix $= \mathbf{u}^H(n)$
variance of measurement noise $v(n) = 1$
Initial conditions:

$$\hat{\mathbf{x}}(1 \mid \mathcal{Y}_0) = E[\mathbf{x}(1)]$$

$$\mathbf{K}(1,0) = E[(\mathbf{x}(1) - E[\mathbf{x}(1)])(\mathbf{x}(1) - E[\mathbf{x}(1)])^H] = \Pi_0$$

Computation: $n = 1, 2, 3 \ldots$

$$\mathbf{K}^{-1}(n) = \lambda[\mathbf{K}^{-1}(n-1) + \mathbf{u}(n)\mathbf{u}^H(n)]$$

$$\mathbf{K}^{-1}(n)\hat{\mathbf{x}}(n+1 \mid \mathcal{Y}_n) = \lambda^{1/2}[\mathbf{K}^{-1}(n-1)\hat{\mathbf{x}}(n|\mathcal{Y}_{n-1}) + \mathbf{u}(n)y(n)]$$

$$\hat{\mathbf{x}}(n+1 \mid \mathcal{Y}_n) = [\mathbf{K}^{-1}(n)]^{-1}\mathbf{K}^{-1}(n)\hat{\mathbf{x}}(n+1|\mathcal{Y}_n)$$

ski et al., 1971; Bierman and Thornton, 1977). For example, according to Eq. (7.56) the matrix $\mathbf{K}(n)$ is defined as the difference between two nonnegative definite matrices; hence, unless the numerical accuracy employed at every iteration of the algorithm is high enough, the matrix $\mathbf{K}(n)$ resulting from this computation may *not* be nonnegative definite. Such a situation is clearly unacceptable, because $\mathbf{K}(n)$ represents a correlation matrix. The unstable behavior of the Kalman filter, which results from numerical inaccuracies due to the use of finite wordlength arithmetic, is called the *divergence phenomenon.*

This problem may be overcome by using numerically stable unitary transformations at every iteration of the Kalman filtering algorithm (Potter, 1963; Kaminski et al., 1971; Morf and Kailath, 1975). In particular, the matrix $\mathbf{K}(n)$ is propagated in a square-root form by using the *Cholesky factorization*[3]:

$$\mathbf{K}(n) = \mathbf{K}^{1/2}(n)\mathbf{K}^{H/2}(n) \tag{7.95}$$

where $\mathbf{K}^{1/2}(n)$ is reserved for a lower triangular matrix, and $\mathbf{K}^{H/2}$ is its Hermitian transpose. In linear algebra, the Cholesky factor $\mathbf{K}^{1/2}(n)$ is commonly referred to as the *square root* of the matrix $\mathbf{K}(n)$. Accordingly, any variant of the Kalman filtering algorithm based on the Cholesky factorization is referred to as *square-root filtering*. The important point to note here is that the matrix product $\mathbf{K}^{1/2}(n)\mathbf{K}^{H/2}(n)$ is much less likely to become indefinite, because the product of any square matrix and its Hermitian transpose is always positive definite. Indeed, even in the presence of roundoff errors, the numerical conditioning of the Cholesky factor $\mathbf{K}^{1/2}(n)$ is generally much better than that of $\mathbf{K}(n)$ itself; see Problem 12.

The information filtering algorithm may also be implemented in a square-root form of its own by propagating the square root $\mathbf{K}^{-1/2}(n)$ rather than the inverse matrix $\mathbf{K}^{-1}(n)$ itself (Kaminski et al., 1971; Bierman, 1977). In this variant of the Kalman filter, the Cholesky factorization is used to express the inverse matrix $\mathbf{K}^{-1}(n)$ as follows:

$$\mathbf{K}^{-1}(n) = \mathbf{K}^{-H/2}(n)\mathbf{K}^{-1/2}(n) \tag{7.96}$$

where $\mathbf{K}^{-1/2}(n)$ is a lower triangular matrix, and $\mathbf{K}^{-H/2}$ is its Hermitian transpose.

UD-factorization

The square-root implementation of a Kalman filter requires more computation than the conventional Kalman filter. This problem of computational efficiency led to the development of a modified version of the square-root filtering algorithm known as the *UD-factorization algorithm* (Bierman, 1977). In this second approach, the filtered state-error correlation matrix $\mathbf{K}(n)$ is factored into an upper triangular matrix $\mathbf{U}(n)$ with 1's along its main diagonal and a real diagonal matrix $\mathbf{D}(n)$, as shown by

$$\mathbf{K}(n) = \mathbf{U}(n)\mathbf{D}(n)\mathbf{U}^{H}(n) \tag{7.97}$$

[3]The Cholesky factorization was also discussed in Section 6.7 in the context of linear prediction.

Equivalently, the factorization may be written as

$$\mathbf{K}(n) = (\mathbf{U}(n)\mathbf{D}^{1/2}(n))\,(\mathbf{U}(n)\mathbf{D}^{1/2}(n))^H \tag{7.98}$$

where $\mathbf{D}^{1/2}(n)$ is the *square-root* of $\mathbf{D}(n)$. The nonnegative definiteness of the computed matrix $\mathbf{K}(n)$ is guaranteed by updating the factors $\mathbf{U}(n)$ and $\mathbf{D}(n)$ instead of $\mathbf{K}(n)$ itself. However, a Kalman filter based on the UD-factorization does *not* possess the numerical advantage of a standard square-root Kalman filter. Moreover, a Kalman filter using UD-factorization may suffer from serious overflow/underflow problems (Stewart and Chapman, 1990). When an arithmetic operation produces a resultant number with too large or too small a characteristic, it is said to suffer from *overflow* or *underflow*, respectively.

One final comment is in order. With the ever-increasing improvements in digital technology, the old argument that square roots are expensive and awkward to calculate is no longer as compelling as it used to be. Accordingly, to avoid the divergence of a Kalman filter, we will only pursue a detailed discussion of square-root filtering in this book. This we do in Chapter 14, after equipping ourselves with certain unitary transformations in Chapter 12.

7.9 THE EXTENDED KALMAN FILTER

The Kalman filtering problem considered up to this point in the discussion has addressed the estimation of a state vector in a linear model of a dynamical system. If, however, the model is *nonlinear*, we may extend the use of Kalman filtering through a linearization procedure. The resulting filter is naturally referred to as the *extended Kalman filter* (EKF). Such an extension is feasible by virtue of the fact that the Kalman filter is described in terms of differential equations (in the case of continuous-time systems) or difference equations (in the case of discrete-time systems). This is in contrast to the Wiener filter that is limited to linear systems, since the notion of an impulse response (on which the Wiener filter is based) is meaningful only in the context of linear systems. Here is another important advantage of the Kalman filter over the Wiener filter.

To set the stage for a development of the extended Kalman filter in the discrete-time domain, consider first the standard linear state-space model that we studied in the earlier part of this chapter [Eqs. (7.17) and (7.19)], reproduced here for convenience of presentation:

$$\mathbf{x}(n+1) = \mathbf{F}(n+1, n)\mathbf{x}(n) + \mathbf{v}_1(n) \tag{7.99}$$

$$\mathbf{y}(n) = \mathbf{C}(n)\mathbf{x}(n) + \mathbf{v}_2(n) \tag{7.100}$$

where $\mathbf{v}_1(n)$ and $\mathbf{v}_2(n)$ are uncorrelated zero-mean white-noise processes with correlation matrices $\mathbf{Q}_1(n)$ and $\mathbf{Q}_2(n)$, respectively, as defined in Eqs. (7.18), (7.20), and (7.21). The corresponding Kalman filter equations are summarized in Table 7.2. In this section, how-

ever, we will rewrite these equations in a slightly modified form that is more convenient for our present discussion. Specifically, the update of the state estimate is performed in two steps. The first step updates $\hat{\mathbf{x}}(n|\mathcal{Y}_n)$ to $\hat{\mathbf{x}}(n+1|\mathcal{Y}_n)$; this update equation is simply (7.59). The second step updates $\hat{\mathbf{x}}(n|\mathcal{Y}_{n-1})$ to $\hat{\mathbf{x}}(n|\mathcal{Y}_n)$ and is obtained by substituting Eq. (7.45) into Eq. (7.60), and by defining a new gain matrix:

$$\mathbf{G}_f(n) = \mathbf{F}^{-1}(n+1, n)\mathbf{G}(n): \tag{7.101}$$

We may thus write

$$\hat{\mathbf{x}}(n+1|\mathcal{Y}_n) = \mathbf{F}(n+1, n)\hat{\mathbf{x}}(n|\mathcal{Y}_n) \tag{7.102}$$

$$\hat{\mathbf{x}}(n|\mathcal{Y}_n) = \hat{\mathbf{x}}(n|\mathcal{Y}_{n-1}) + \mathbf{G}_f(n)\boldsymbol{\alpha}(n) \tag{7.103}$$

$$\boldsymbol{\alpha}(n) = \mathbf{y}(n) - \mathbf{C}(n)\hat{\mathbf{x}}(n|\mathcal{Y}_{n-1}) \tag{7.104}$$

$$\mathbf{G}_f(n) = \mathbf{K}(n, n-1)\mathbf{C}^H(n)[\mathbf{C}(n)\mathbf{K}(n, n-1)\mathbf{C}^H(n) + \mathbf{Q}_2(n)]^{-1} \tag{7.105}$$

$$\mathbf{K}(n+1, n) = \mathbf{F}(n+1, n)\mathbf{K}(n)\mathbf{F}^H(n+1, n) + \mathbf{Q}_1(n) \tag{7.106}$$

$$\mathbf{K}(n) = [\mathbf{I} - \mathbf{G}_f(n)\mathbf{C}(n)]\mathbf{K}(n, n-1) \tag{7.107}$$

We next make the following observation. Suppose that instead of the state equations (7.99) and (7.100), we are given the alternative state-space model

$$\mathbf{x}(n+1) = \mathbf{F}(n+1, n)\mathbf{x}(n) + \mathbf{v}_1(n) + \mathbf{d}(n) \tag{7.108}$$

$$\mathbf{y}(n) = \mathbf{C}(n)\mathbf{x}(n) + \mathbf{v}_2(n) \tag{7.109}$$

where $\mathbf{d}(n)$ is a known (i.e., nonrandom) vector. In this case, it is easily verified that the same Kalman equations (7.103) through (7.107) apply except for a modification in the first equation (7.102), which now reads as follows:

$$\hat{\mathbf{x}}(n+1|\mathcal{Y}_n) = \mathbf{F}(n+1, n)\hat{\mathbf{x}}(n|\mathcal{Y}_n) + \mathbf{d}(n) \tag{7.110}$$

This modification arises in the derivation of the extended Kalman filter, as discussed in the sequel.

As mentioned previously, the extended Kalman filter (EKF) is an *approximate* solution that allows us to extend the Kalman filtering idea to *nonlinear* state-space models (Jazwinski, 1970; Maybeck, 1982; Ljung and Söderstrom, 1983). In particular, the nonlinear model considered here has the following form:

$$\mathbf{x}(n+1) = \mathbf{F}(n,\mathbf{x}(n)) + \mathbf{v}_1(n) \tag{7.111}$$

$$\mathbf{y}(n) = \mathbf{C}(n,\mathbf{x}(n)) + \mathbf{v}_2(n) \tag{7.112}$$

where, as before, $\mathbf{v}_1(n)$ and $\mathbf{v}_2(n)$ are uncorrelated zero-mean white-noise processes with correlation matrices $\mathbf{Q}_1(n)$ and $\mathbf{Q}_2(n)$, respectively. Here, however, the functional

$\mathbf{F}(n,\mathbf{x}(n))$ denotes a nonlinear transition matrix function that is possibly time-variant. In the linear case,we simply have

$$\mathbf{F}(n,\mathbf{x}(n)) = \mathbf{F}(n+1,n)\mathbf{x}(n)$$

But in a general nonlinear setting, the entries of the state vector $\mathbf{x}(n)$ may be combined nonlinearly by the action of the functional $\mathbf{F}(n,\mathbf{x}(n))$. Moreover, this nonlinear operation may vary with time. Likewise, the functional $\mathbf{C}(n,\mathbf{x}(n))$ denotes a *nonlinear measurement matrix* that may be time-variant too.

As an example, consider the following two-dimensional nonlinear state-space model:

$$\begin{bmatrix} x_1(n+1) \\ x_2(n+1) \end{bmatrix} = \begin{bmatrix} x_1(n) + x_2^2(n) \\ nx_1(n) - x_1(n)x_2(n) \end{bmatrix} + \begin{bmatrix} v_{1,1}(n) \\ v_{1,2}(n) \end{bmatrix}$$

$$y(n) = x_1(n)x_2^2(n) + v_2(n)$$

In this example, we have

$$\mathbf{F}(n,\mathbf{x}(n)) = \begin{bmatrix} x_1(n) + x_2^2(n) \\ nx_1(n) - x_1(n)x_2(n) \end{bmatrix}$$

and

$$\mathbf{C}(n,\mathbf{x}(n)) = x_1(n)x_2^2(n)$$

The basic idea of the extended Kalman filter is to *linearize* the state-space model of Eqs. (7.111) and (7.112) at each time instant around the most recent state estimate, which is taken to be either $\hat{\mathbf{x}}(n|\mathcal{Y}_n)$ or $\hat{\mathbf{x}}(n|\mathcal{Y}_{n-1})$, depending on which particular functional is being considered. Once a linear model is obtained, the standard Kalman filter equations are applied.

More explicitly, the approximation proceeds in two stages.

Stage 1. The following two matrices are constructed

$$\mathbf{F}(n+1,n) = \left.\frac{\partial \mathbf{F}(n,\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}=\hat{\mathbf{x}}(n|\mathcal{Y}_n)} \tag{7.113}$$

and

$$\mathbf{C}(n) = \left.\frac{\partial \mathbf{C}(n,\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}=\hat{\mathbf{x}}(n|\mathcal{Y}_{n-1})} \tag{7.114}$$

That is, the ijth entry of $\mathbf{F}(n+1,n)$ is equal to the partial derivative of the ith component of $\mathbf{F}(n,\mathbf{x})$ *with respect to the* jth component of $\mathbf{x}$. Likewise, the ijth entry of $\mathbf{C}(n)$ is equal to the partial derivative of the ith component of $\mathbf{C}(n,\mathbf{x})$ with respect to the jth component of $\mathbf{x}$. In the former case, the derivatives are evaluated at $\hat{\mathbf{x}}(n|\mathcal{Y}_n)$, while in the latter case

the derivatives are evaluated at $\hat{\mathbf{x}}(n|\mathcal{Y}_{n-1})$. The entries of the matrices $\mathbf{F}(n+1,n)$ and $\mathbf{C}(n)$ are all known (i.e., computable), since $\hat{\mathbf{x}}(n|\mathcal{Y}_n)$ and $\hat{\mathbf{x}}(n|\mathcal{Y}_{n-1})$ are made available as described later.

Applying the definitions of Eqs. (7.113) and (7.114) to the previous example, we get

$$\frac{\partial \mathbf{F}(n, \mathbf{x})}{\partial \mathbf{x}} = \begin{bmatrix} 1 & 2x_2 \\ n - x_2 & -x_1 \end{bmatrix}$$

$$\frac{\partial \mathbf{C}(n, \mathbf{x})}{\partial \mathbf{x}} = [x_2^2 \quad 2x_1x_2]$$

which leads to

$$\mathbf{F}(n+1, n) = \begin{bmatrix} 1 & 2\hat{x}_2(n|\mathcal{Y}_n) \\ n - \hat{x}_2(n|\mathcal{Y}_n) & -\hat{x}_1(n|\mathcal{Y}_n) \end{bmatrix}$$

and

$$\mathbf{C}(n) = [\hat{x}_2^2(n|\mathcal{Y}_{n-1}) \quad 2\hat{x}_1(n|\mathcal{Y}_{n-1})\,\hat{x}_2(n|\mathcal{Y}_{n-1})]$$

Stage 2. Once the matrices $\mathbf{F}(n+1,n)$ and $\mathbf{C}(n)$ are evaluated, they are then employed in a *first-order Taylor approximation* of the nonlinear functionals $\mathbf{F}(n,\mathbf{x}(n))$ and $\mathbf{C}(n,\mathbf{x}(n))$ around $\hat{\mathbf{x}}(n|\mathcal{Y}_n)$ and $\hat{\mathbf{x}}(n|\mathcal{Y}_{n-1})$, respectively. Specifically, $\mathbf{F}(n,\mathbf{x}(n))$ and $\mathbf{C}(n,\mathbf{x}(n))$ are approximated as follows, respectively:

$$\mathbf{F}(n,\mathbf{x}(n)) \simeq \mathbf{F}(n, \hat{\mathbf{x}}(n|\mathcal{Y}_n)) + \mathbf{F}(n+1,n)\,[\mathbf{x}(n) - \hat{\mathbf{x}}(n|\mathcal{Y}_n)] \tag{7.115}$$

$$\mathbf{C}(n, \mathbf{x}(n)) \simeq \mathbf{C}(n, \hat{\mathbf{x}}(n|\mathcal{Y}_{n-1})) + \mathbf{C}(n)[\mathbf{x}(n) - \hat{\mathbf{x}}(n|\mathcal{Y}_{n-1})] \tag{7.116}$$

With the above approximate expressions at hand, we may now proceed to approximate the nonlinear state-equations (7.111) and (7.112) as shown by, respectively,

$$\mathbf{x}(n+1) \simeq \mathbf{F}(n+1, n)\mathbf{x}(n) + \mathbf{v}_1(n) + \mathbf{d}(n) \tag{7.117}$$

$$\bar{\mathbf{y}}(n) \simeq \mathbf{C}(n)\mathbf{x}(n+\mathbf{v}_2(n) \tag{7.118}$$

where we have introduced two new quantities:

$$\bar{\mathbf{y}}(n) = \mathbf{y}(n) - [\mathbf{C}(n,\hat{\mathbf{x}}(n|\mathcal{Y}_{n-1})) - \mathbf{C}(n)\hat{\mathbf{x}}(n|\mathcal{Y}_{n-1})] \tag{7.119}$$

and

$$\mathbf{d}(n) = \mathbf{F}(n,\hat{\mathbf{x}}(n|\mathcal{Y}_n)) - \mathbf{F}(n+1,n)\hat{\mathbf{x}}(n|\mathcal{Y}_n) \tag{7.120}$$

The entries in the term $\bar{\mathbf{y}}(n)$ are all known at time n, and, therefore, $\bar{\mathbf{y}}(n)$ can be regarded as an observation vector at time n. Likewise, the entries in the term $\mathbf{d}(n)$ are all known at time n.

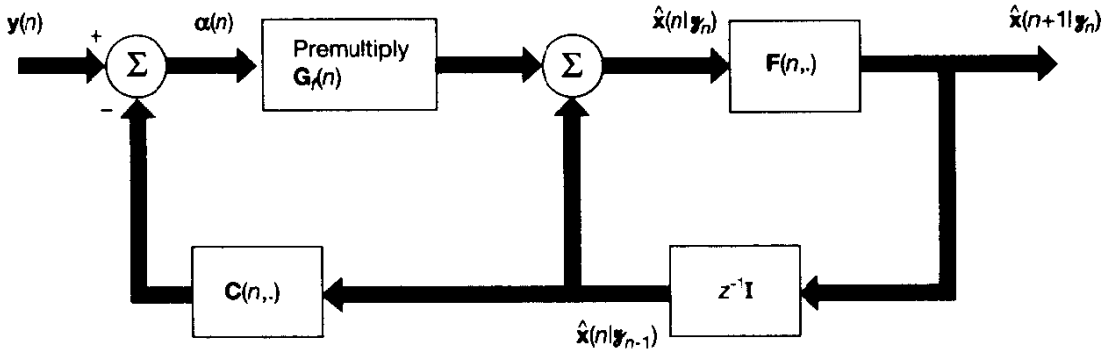


Figure 7.6 One-step predictor for the extended Kalman filter.

The approximate state-space model of Eqs. (7.117) and (7.118) is a linear model of the same mathematical form as that described in Eqs. (7.108) and (7.109); indeed, it is with this objective in mind that earlier on we formulated the state-space model of Eqs. (7.108) and (7.109). The extended Kalman filter equations simply correspond to applying the standard Kalman equations (7.103) through (7.109) and (7.110) to the above linear model. This leads to the following set of equations:

$$\begin{aligned}
\hat{\mathbf{x}}(n+1|\mathcal{Y}_n) &= \mathbf{F}(n+1,n)\hat{\mathbf{x}}(n|\mathcal{Y}_n) + \mathbf{d}(n) \\
&= \mathbf{F}(n+1,n)\hat{\mathbf{x}}(n|\mathcal{Y}_n) + [\mathbf{F}(n,\hat{\mathbf{x}}(n|\mathcal{Y}_n)) - \mathbf{F}(n+1,n)\hat{\mathbf{x}}(n|\mathcal{Y}_n)] \\
&= \mathbf{F}(n,\hat{\mathbf{x}}(n|\mathcal{Y}_n)) \qquad (7.121) \\
\hat{\mathbf{x}}(n|\mathcal{Y}_n) &= \hat{\mathbf{x}}(n|\mathcal{Y}_{n-1}) + \mathbf{G}_f(n)\boldsymbol{\alpha}(n) \\
\boldsymbol{\alpha}(n) &= \tilde{\mathbf{y}}(n) - \mathbf{C}(n)\hat{\mathbf{x}}(n|\mathcal{Y}_{n-1}) \\
&= \mathbf{y}(n) - \mathbf{C}(n,\hat{\mathbf{x}}(n|\mathcal{Y}_{n-1})) + \mathbf{C}(n)\hat{\mathbf{x}}(n|\mathcal{Y}_{n-1}) - \mathbf{C}(n)\hat{\mathbf{x}}(n|\mathcal{Y}_{n-1}) \\
&= \mathbf{y}(n) - \mathbf{C}(n,\hat{\mathbf{x}}(n|\mathcal{Y}_{n-1})) \qquad (7.122)
\end{aligned}$$

On the basis of Eqs. (7.121) and (7.122), we may formulate the signal-flow graph of Fig. 7.6 for updating the one-step prediction in the extended Kalman filter.

In Table 7.5 we present a summary of the extended Kalman filtering algorithm, where the linearized matrices $\mathbf{F}(n+1, n)$ and $\mathbf{C}(n)$ are computed from their respective nonlinear counterparts using Eqs. (7.113) and (7.114). Given a nonlinear state-space model of the form described in Eqs. (7.111) and (7.112), we may thus use this algorithm to compute state estimates recursively. Comparing the equations of the extended Kalman filter

TABLE 7.5 SUMMARY OF THE EXTENDED KALMAN FILTER

Input vector process

Observations = $\{\mathbf{y}(1), \mathbf{y}(2), \ldots, \mathbf{y}(n)\}$

Known parameters

Nonlinear state transition matrix = $\mathbf{F}(n, \mathbf{x}(n))$

Nonlinear measurement matrix = $\mathbf{C}(n, \mathbf{x}(n))$

Correlation matrix of process noise vector = $\mathbf{Q}_1(n)$

Correlation matrix of measurement noise vector = $\mathbf{Q}_2(n)$

Computation: $n = 1, 2, 3, \ldots$.

$$\mathbf{G}_f(n) = \mathbf{K}(n, n-1)\mathbf{C}^H(n)[\mathbf{C}(n)\mathbf{K}(n, n-1)\mathbf{C}^H(n) + \mathbf{Q}_2(n)]^{-1}$$

$$\boldsymbol{\alpha}(n) = \mathbf{y}(n) - \mathbf{C}(n, \hat{\mathbf{x}}(n|\mathcal{Y}_{n-1}))$$

$$\hat{\mathbf{x}}(n|\mathcal{Y}_n) = \hat{\mathbf{x}}(n|\mathcal{Y}_{n-1}) + \mathbf{G}_f(n)\boldsymbol{\alpha}(n)$$

$$\hat{\mathbf{x}}(n+1|\mathcal{Y}_n) = \mathbf{F}(n, \hat{\mathbf{x}}(n|\mathcal{Y}_n))$$

$$\mathbf{K}(n) = [\mathbf{I} - \mathbf{G}_f(n)\mathbf{C}(n)]\mathbf{K}(n, n-1)$$

$$\mathbf{K}(n+1, n) = \mathbf{F}(n+1, n)\mathbf{K}(n)\mathbf{F}^H(n+1, n) + \mathbf{Q}_1(n)$$

Note: The linearized matrices $\mathbf{F}(n+1, n)$ and $\mathbf{C}(n)$ are computed from their nonlinear counterparts $\mathbf{F}(n, \mathbf{x}(n))$ and $\mathbf{C}(n, \mathbf{x}(n))$ using Eqs. (7.113) and (7.114), respectively.

Initial conditions

$$\hat{\mathbf{x}}(1|\mathcal{Y}_0) = E[\mathbf{x}(1)]$$

$$\mathbf{K}(1,0) = E[(\mathbf{x}(1) - E[\mathbf{x}(1)])(\mathbf{x}(1) - E[\mathbf{x}(1)])^H] = \boldsymbol{\Pi}_0$$

summarized herein with those of the standard Kalman filter given in Eqs. (7.102) through (7.107), we see that the only differences between them arise in the computations of the innovations vector $\boldsymbol{\alpha}(n)$ and the updated estimate $\hat{\mathbf{x}}(n+1|\mathcal{Y}_n)$. Specifically, the linear terms $\mathbf{F}(n+1, n)\hat{\mathbf{x}}(n|\mathcal{Y}_n)$ and $\mathbf{C}(n)\hat{\mathbf{x}}(n|\mathcal{Y}_{n-1})$ in the standard Kalman filter are replaced by the approximate terms $\mathbf{F}(n, \hat{\mathbf{x}}(n|\mathcal{Y}_n))$ and $\mathbf{C}(n, \hat{\mathbf{x}}(n|\mathcal{Y}_{n-1}))$, respectively, in the extended Kalman filter. These differences also show up in comparing the signal-flow graph of Fig. 7.3 for one-step prediction in the standard Kalman filter with that of Fig. 7.6 for one-step prediction in the extended Kalman filter.

7.10 SUMMARY AND DISCUSSION

The Kalman filter is a linear, discrete-time, finite-dimensional system, the implementation of which is well suited for a digital computer. A key property of the Kalman filter is that it leads to minimization of the trace of the filtered state error correlation matrix $\mathbf{K}(n)$. This, in turn, means that the Kalman filter is the *linear minimum variance estimator* of the state vector $\mathbf{x}(n)$ (Anderson and Moore, 1979; Goodwin and Sin, 1984).

The Kalman filter has been successfully applied to solve many real-world problems as can be seen in the literature on control systems (Sorenson, 1985). Moreover, the Kalman

filter provides the general framework for deriving *all* of the known algorithms that constitute the recursive least-squares family of adaptive filters (Sayed and Kailath, 1994). In the intervening two decades between the paper by Sayed and Kailath and the seminal paper by Godard in 1974, many attempts were made to incorporate this important family of adaptive filtering algorithms into the framework of Kalman filter theory. However, some annoying discrepancies always remained, thereby hindering the full application of the extensive control literature on Kalman filters to adaptive filtering problems. For the first time, the paper by Sayed and Kailath has shown us how to devise a state-space model for the adaptive filtering problem that is a perfect match for the application of Kalman filter theory. It has been said by many that many of the problems encountered in signal processing and control theory are mathematically equivalent. The link between Kalman filter theory and adaptive filter theory demonstrated in the paper by Sayed and Kailath is further testimony to the validity of this mathematical equivalence.

PROBLEMS

1. The Gram–Schmidt orthogonalization procedure enables the set of observation vectors $\mathbf{y}(1)$, $\mathbf{y}(2), \ldots, \mathbf{y}(n)$ to be transformed into the set of innovations processes $\boldsymbol{\alpha}(1), \boldsymbol{\alpha}(2), \ldots, \boldsymbol{\alpha}(n)$ without loss of information, and vice versa. Illustrate this procedure for $n = 2$, and comment on the procedure for $n > 2$.

2. The predicted state-error vector is defined by

$$\boldsymbol{\epsilon}(n, n-1) = \mathbf{x}(n) - \hat{\mathbf{x}}(n|\mathcal{Y}_{n-1})$$

where $\hat{\mathbf{x}}(n|\mathcal{Y}_{n-1})$ is the minimum mean-square estimate of the state $\mathbf{x}(n)$, given the space $\mathcal{Y}_{n-1}$ that is spanned by the observed data $\mathbf{y}(1), \ldots, \mathbf{y}(n-1)$. Let $\mathbf{v}_1(n)$ and $\mathbf{v}_2(n)$ denote the process noise and measurement noise vectors, respectively. Show that $\boldsymbol{\epsilon}(n, n-1)$ is orthogonal to both $\mathbf{v}_1(n)$ and $\mathbf{v}_2(n)$; that is,

$$E[\boldsymbol{\epsilon}(n, n-1)\mathbf{v}_1^H(n)] = \mathbf{0}$$

and

$$E[\boldsymbol{\epsilon}(n, n-1)\mathbf{v}_2^H(n)] = \mathbf{0}$$

3. Consider a set of scalar observations $y(n)$ of zero mean, which is transformed into the corresponding set of innovations $\alpha(n)$ of zero mean and variance $\sigma_\alpha^2(n)$. Let the estimate of the state vector $\mathbf{x}(i)$, given this set of data, be expressed as

$$\hat{\mathbf{x}}(i|\mathcal{Y}_n) = \sum_{k=1}^{n} \mathbf{b}_i(k)\alpha(k)$$

where $\mathcal{Y}_n$ is the space spanned by $y(1), \ldots, y(n)$, and $\mathbf{b}_i(k)$, $k = 1, 2, \ldots, n$, is a set of vectors to be determined. The requirement is to choose the $\mathbf{b}_i(k)$ so as to minimize the expected value of the squared norm of the estimated state-error vector

$$\boldsymbol{\epsilon}(i|\mathcal{Y}_n) = \mathbf{x}(i) - \hat{\mathbf{x}}(i|\mathcal{Y}_n)$$

Show that this minimization yields the result

$$\hat{\mathbf{x}}(i|\mathcal{Y}_n) = \sum_{k=1}^{n} E[\mathbf{x}(i)\phi^*(k)]\phi(k)$$

where $\phi(k)$ is the normalized innovation

$$\phi(k) = \frac{\alpha(k)}{\sigma_\alpha(k)}$$

This result may be viewed as a special case of Eqs. (7.37) and (7.40).

4. The Kalman gain $\mathbf{G}(n)$ defined in Eq. (7.49) involves the inverse matrix $\mathbf{R}^{-1}(n)$. The matrix $\mathbf{R}(n)$ is itself defined in Eq. (7.35), reproduced here for convenience:

$$\mathbf{R}(n) = \mathbf{C}(n)\mathbf{K}(n, n-1)\mathbf{C}^H(n) + \mathbf{Q}_2(n)$$

The matrix $\mathbf{C}(n)$ is nonnegative definite but not necessarily nonsingular.

(a) Why is $\mathbf{R}(n)$ nonnegative definite?

(b) What prior condition would you impose on the matrix $\mathbf{Q}_2(n)$ to ensure that the inverse matrix $\mathbf{R}^{-1}(n)$ exists?

5. In many cases the predicted state-error correlation matrix $\mathbf{K}(n+1, n)$ converges to the steady-state value $\mathbf{K}$ as the number of iterations n approaches infinity. Show that the limiting value $\mathbf{K}$ satisfies the *algebraic Riccati equation*

$$\mathbf{K}\mathbf{C}^H(\mathbf{C}\mathbf{K}\mathbf{C}^H + \mathbf{Q}_2)^{-1}\mathbf{C}\mathbf{K} - \mathbf{Q}_1 = \mathbf{0}$$

where it is assumed that the state transition matrix equals the identity matrix, and the matrices $\mathbf{C}$, $\mathbf{Q}_1$, and $\mathbf{Q}_2$ are the limiting values of $\mathbf{C}(n)$, $\mathbf{Q}_1(n)$, and $\mathbf{Q}_2(n)$, respectively.

6. Consider a stochastic process $y(n)$ represented by an autoregressive-moving average (ARMA) model of order (1, 1):

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

where a and b are the ARMA parameters and $v(n)$ is a zero-mean white-noise process of variance σ^2.

(a) Show that a state-space representation of this model is

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

$$y(n) = [1 \quad 0]\mathbf{x}(n)$$

where $\mathbf{x}(n)$ is a 2-by-1 state vector.

(b) Assume the applicability of the algebraic Riccati equation described in Problem 5. Hence, show that the solution of this equation is

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

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

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

What are the two values of c that satisfy this equation? Determine the corresponding values of the matrix $\mathbf{K}$.

(c) Show that the Kalman gain is

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

Determine the values of $\mathbf{G}$ that correspond to the solutions for the scalar c found in part (b).

7. In this problem we consider the general case of *time-varying real-valued ARMA process* $y(n)$ described by the difference equation:

$$y(n) + \sum_{k=1}^{M} a_k(n)y(n-k) = \sum_{k=1}^{N} a_{M+k}(n)v(n-k) + v(n)$$

where $a_1(n), a_2(n), \ldots, a_M(n), a_{M+1}(n), a_{M+2}(n), \ldots, a_{M+N}(n)$ are the ARMA coefficients, the process $v(n)$ is the input, and process $y(n)$ is the output. The process $v(n)$ is a white Gaussian noise process of zero mean and variance σ^2. The ARMA coefficients are subject to random fluctuations, as shown in the model

$$a_k(n+1) = a_k(n) + w_k(n),\ k = 1, \ldots, M+N$$

where $w_k(n)$ is a zero-mean, white Gaussian noise process that is independent of $w_j(n)$ for $j \neq k$, and also independent of $v(n)$. The issue of interest is to provide a technique based on the Kalman filter for identifying the coefficients of the ARMA process. To do this, we define an $(M + N)$-dimensional state vector:

$$\mathbf{x}(n) = [a_1(n), \ldots, a_M(n), \ldots, a_{M+N}(n)]^T$$

We also define the measurement matrix (actually, a row vector):

$$\mathbf{C}(n) = [-y(n-1), \ldots, -y(n-M), v(n-1), \ldots, v(n-N)]$$

On this basis, do the following:

(a) Formulate the state-space equations for the ARMA process.

(b) Find an algorithm for computing the predicted value of the state vector $\mathbf{x}(n+1)$, given the observation $y(n)$.

(c) How would you initialize the algorithm in part (b)?

8. Consider a communication channel modeled as an FIR filter of known impulse response. The channel output $y(n)$ is defined by

$$y(n) = \mathbf{h}^T\mathbf{x}(n) + w(n)$$

where $\mathbf{h}$ is an M-by-1 vector representing the channel impulse response, $\mathbf{x}(n)$ is an M-by-1 vector representing the present value $u(n)$ of the channel input and $(M-1)$ previous transmissions, and $w(n)$ is a white Gaussian noise process of zero mean and variance σ_w^2. At time n, the channel input $u(n)$ consists of a coded binary sequence of zeros and ones, statistically independent of $w(n)$. This model suggests that we may view $\mathbf{x}(n)$ as a state vector, in which case the state equation is written as[4]

$$\mathbf{x}(n+1) = \mathbf{A}\mathbf{x}(n) + \mathbf{b}v(n)$$

[4]This problem is adapted from Lawrence and Kaufman (1971).

where $v(n)$ is a white Gaussian noise process of zero mean and variance σ_v^2 which is independent of $w(n)$. The matrix $\mathbf{A}$ is an M-by-M matrix whose ijth element is defined by

$$a_{ij} = \begin{cases} 1, & i = j + 1 \\ 0, & \text{otherwise} \end{cases}$$

The vector $\mathbf{b}$ is an M-by-1 vector whose ith element is defined by

$$b_i = \begin{cases} 1, & i = 1 \\ 0, & i = 2, \ldots, M \end{cases}$$

We may now state the problem: Given the foregoing channel model and a sequence $y(n)$ of noisy measurements made at the channel output, use the Kalman filter to construct an equalizer that yields a good estimate of the channel input $u(n)$ at some delayed time $(n + D)$, where $0 \leq D \leq M - 1$. Show that the equalizer so constructed is an IIR filter where coefficients are determined by two distinct sets of parameters: (a) the M-by-1 channel impulse response vector, and (b) the Kalman gain, which (in this problem) is an M-by-1 vector.

9. For the case when the transition matrix $\mathbf{F}(n + 1, n)$ is the identity matrix and the state noise vector is zero, show that the predicted state-error correlation matrix $\mathbf{K}(n + 1, n)$ and the filtered state-error correlation matrix $\mathbf{K}(n)$ are equal.

10. Using the initial conditions described in Eqs. (7.72) and (7.73), show that the resulting filtered estimate $\hat{\mathbf{x}}(n|\mathcal{Y}_n)$ produced by the Kalman filter is unbiased; that is,

$$E[\hat{\mathbf{x}}(n)|\mathcal{Y}(n)] = \mathbf{x}(n)$$

11. In the UD-factorization algorithm, the filtered state-error correlation matrix $\mathbf{K}(n)$ is expressed as follows

$$\mathbf{K}(n) = \mathbf{U}(n)\mathbf{D}(n)\mathbf{U}^H(n)$$

where $\mathbf{U}(n)$ is an upper triangular matrix with 1's along its main diagonal, and $\mathbf{D}(n)$ is a real diagonal matrix. Let λ_{max} and λ_{min} denote the maximum and minimum eigenvalues of the matrix $\mathbf{K}(n)$. Show that the condition number of the diagonal matrix $\mathbf{D}(n)$ is governed by

$$\chi(\mathbf{D}) \geq \frac{\lambda_{max}}{\lambda_{min}} = \chi(\mathbf{K})$$

12. Let $\chi(\mathbf{K})$ denote the condition number of the filtered state-error correlation matrix $\mathbf{K}(n)$, defined as the ratio of the largest eigenvalue λ_{max} to the smallest eigenvalue λ_{min}. Show that

$$\chi(\mathbf{K}) = (\chi(\mathbf{K}^{1/2}))^2$$

where $\mathbf{K}^{1/2}(n)$ is the square-root of $\mathbf{K}(n)$. What is the computational implication of this relation?

13. Consider the state-space model described in Eqs. (7.108) and (7.109). Show that the one-step prediction $\mathbf{x}(n + 1|\mathcal{Y}_n)$ of the state vector in this model is given by Eq. (7.110).

14. (a) Figures 7.3 and 7.6 are signal-flow graph representations of the one-step predictor for the standard Kalman filter and extended Kalman filter, respectively. Show that for a linear model of a dynamical system, these two representations are equivalent.

(b) Figure 7.5 shows a block diagram representation of the standard Kalman filter. How is this block diagram modified for representation of the extended Kalman filter?