5

Fundamentals of Adaptive Signal Processing

Samuel D. Stearns
Sandia National Laboratories

5.0 INTRODUCTION

Adaptive signal processing has undergone a remarkable increase in interest and attention in recent years. This growth has been promoted by developments in microelectronics and VLSI circuit design that increased tremendously the amount of computing possible in processing digital signals. Seismic signals at 10^2 Hz and below, speech and acoustic signals at 10^2–10^5 Hz, electromagnetic signals at 10^5 Hz and beyond, as well as other similar signals appearing in time and space are now all reasonable candidates for adaptive signal processing.

Adaptive signal processing systems are mainly time-varying digital signal processing systems, that is, digital filters and similar structures with time-varying coefficients or "weights." The "adaptive" notion derives from the human desire to emulate living systems found in nature, which adapt to their environments in various remarkable ways and reflect the work of a master Designer that we have been able to copy only in very limited ways. In adaptive systems, the journey from concept to reality has taken us away from direct emulation, somewhat like the development of manned aircraft. An adaptive signal processor usually resembles its natural counterpart about as much as, or if anything less than, an airplane resembles a bird.

Adaptive signal processing has its roots in adaptive control and in the mathematics of iterative processes, where the first attempts were made to design systems that adapt to their environments. In this chapter we will not trace the early developments

Samuel D. Stearns is with Sandia National Laboratories, Division 7111, Albuquerque, NM 87185.

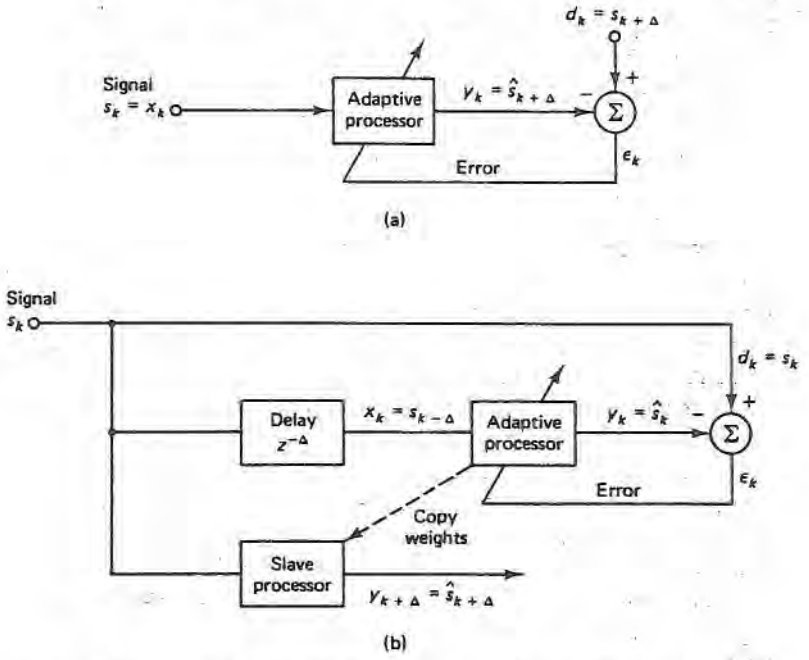

Figure 5.2 Adaptive prediction: (a) unrealizable form; (b) realizable form.

equivalent to Fig. 5.2(a), is used. The estimate, $\hat{s}_{k+\Delta}$, is produced using a slave processor. Adaptive prediction is used in speech encoding [6,7], spectral estimation [8], event detection [9–11], line enhancement [2,12], signal whitening, and other areas. Note how Fig. 5.1, the scheme common to many adaptive systems, is incorporated into the prediction schemes in Fig. 5.2.

Another application, *adaptive modeling*, is shown in Fig. 5.3. In forward modeling, illustrated in Fig. 5.3(a), the adaptive processor adjusts its weights to produce a response y_k that is as close as possible (in the least-squares sense) to the plant response d_k. If the stimulus s_k is robust in frequency content and if the internal plant noise n_k is small, then the adaptive processor will adapt to become a good model of the unknown system. Forward modeling has a wide range of applications in the biological, social, and economic sciences [5], adaptive control systems [13,14], digital filter design [15], coherence estimation [16], and geophysics [5]. With inverse modeling, shown in Fig. 5.3(b), the adaptive processor adjusts its weights to become the inverse of the plant, that is, to convert the plant output x_k back into a delayed version $s_{k-\Delta}$ of the input. The delay is usually included to account for the propagation delay through the plant and the adaptive processor, assuming that both are causal systems. As with forward modeling, if s_k is spectrally robust and if n_k is small, the adaptive processor can adapt to become an accurate inverse model of the unknown system. Inverse modeling is used in adaptive control [5,17,18], speech analysis [19], channel equalization [20,21], deconvolution [22], digital filter design [15], and other applications. Again, note the incorporation of the basic structure of Fig. 5.1 into Figs. 5.3(a) and (b).

in adaptive control in the 1940s and 1950s, many of which are still useful. We will describe only some of the basic theory and applications of modern adaptive signal processing. Much of what we describe here is based on the more recent work of Bernard Widrow and his colleagues [1–4], which began around 1960. We use Widrow's development of the geometry of adaptation [4], and we introduce the least-mean-square (LMS) algorithm as the simplest and most widely applicable adaptive processing scheme.

We will view the adaptive signal processor as a time-varying digital structure. To simplify our discussion, we assume that the signals to be processed are digital time series with regularly spaced samples. We begin with some basic aspects and examples of such structures. In adaptation with "performance feedback" [5], the type of adaptation we discuss in this chapter, we are usually able to identify in any system the basic elements in Fig. 5.1. The adaptive system contains a digital structure, the processor, with variable, adaptive weights. The weights are adjusted repeatedly, at regular intervals, in accordance with an *adaptive algorithm*. The adaptive algorithm usually uses, either explicitly or implicitly, all of the signals shown in Fig. 5.1. The details of adaptive algorithms are given in the following sections. The main point here is that the weights in Fig. 5.1 are adjusted continually during adaptation to reduce the mean-square error $E[\epsilon_k^2]$ toward its minimum value. In this way the adaptive system seeks continually to reduce the difference between a desired response d_k and its own response y_k; thus, as we will see, the system "adapts" to its prescribed signal environment.

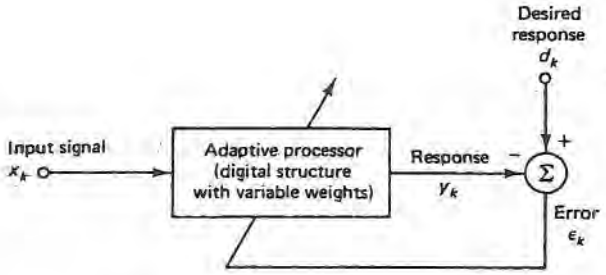

Figure 5.1 Basic elements in a single-input, performance-feedback adaptive system.

The subscript k in Fig. 5.1 denotes time and generally ranges from zero to infinity. For example, the input time series, which begins at some time which we label 0 for convenience, is $[x_0, x_1, x_2, \ldots, x_k, \ldots]$.† We have shown in Fig. 5.1 a single-input system, but multiple-input systems are also possible, as with adaptive arrays, and we will introduce a notation to cover both cases. The other three signals in Fig. 5.1 are all single time series in this chapter.

We now consider some examples of the application of Fig. 5.1, beginning with Fig. 5.2, which illustrates *adaptive prediction*. Figure 5.2(a) gives the simpler but unrealizable case, where the adaptive processor tries to minimize $E[\epsilon_k^2]$ and thereby make its output y_k as close as possible to a future value of the input, $s_{k+\Delta}$. Since the latter is not available in real time, the "realizable" form in Fig. 5.2(b), which is

†The subscript notation is used in this chapter in place of the "argument" notation in previous chapters to conform with the adaptive literature and to make the equations less cumbersome. Thus, x_k denotes the kth sample of x, the same as $x(k)$.

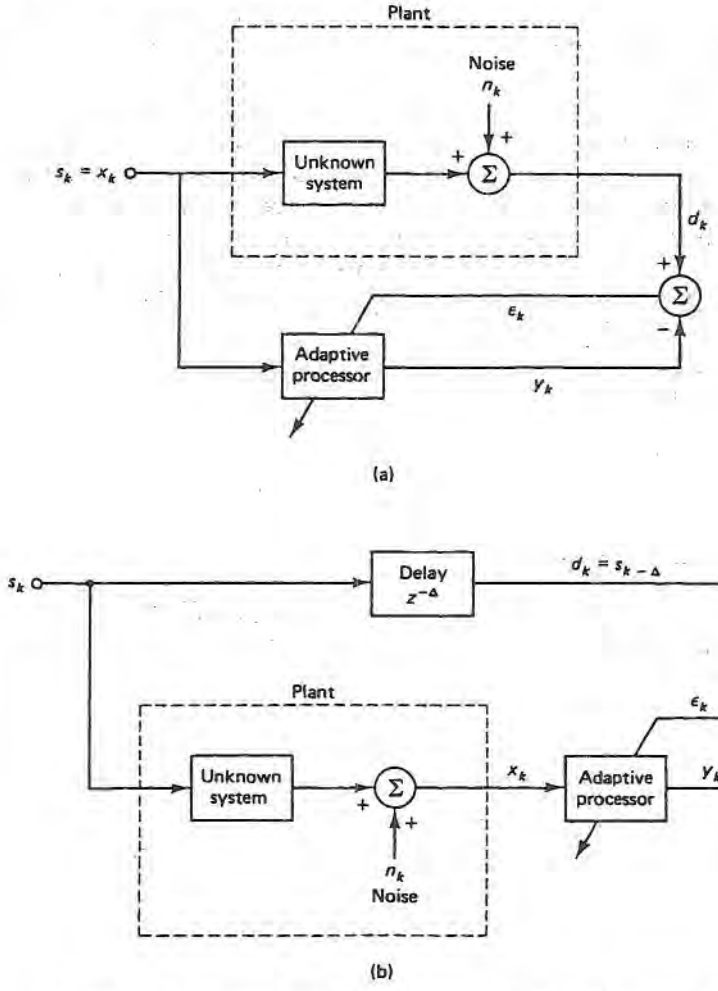

Figure 5.3 Adaptive modeling: (a) forward modeling; (b) inverse modeling.

Another application of adaptive signal processing, known as *adaptive interference canceling*, is illustrated in Fig. 5.4. This simple structure appears in a wide variety of applications [2]. The desired response d_k is in this case a signal with additive noise, $s_k + n_k$. The input x_k to the adaptive processor is another noise, n_k', correlated with n_k. When s_k and n_k are uncorrelated, the adaptive processor tries to minimize $E[\epsilon_k^2]$ by making $y_k = \hat{n}_k$ an approximation to n_k, thereby making ϵ_k an approximation

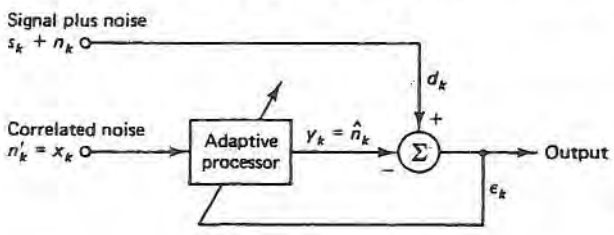

Figure 5.4 Basic structure for adaptive interference canceling.

to the signal s_k. The result is reduced noise in ϵ_k. A more quantitative analysis is given elsewhere [5]; here we wish mainly to emphasize again the incorporation of Fig. 5.1 into Fig. 5.4.

Our final application example, *adaptive array processing*, is illustrated in Fig. 5.5. The array processing system is itself a type of multiple-input interference canceler [5,23], as seen by comparing Figs. 5.4 and 5.5. As in ordinary beamforming, steering delays are used to form a beam and produce a peak array gain in a desired look direction. Thus a noisy target signal, $s_k + n_k$, is obtained through the fixed filter shown in Fig. 5.5. An estimate $\hat{n}_k$ of the noise is obtained through the multiple-input adaptive processor and is used to cancel n_k, just as in Fig. 5.4. Further details, including details on how the adaptive processor manages to exclude the signal s_k from its estimate of n_k, are given in the literature [3,5,23]; for now we see in Fig. 5.5 still another application of Fig. 5.1.

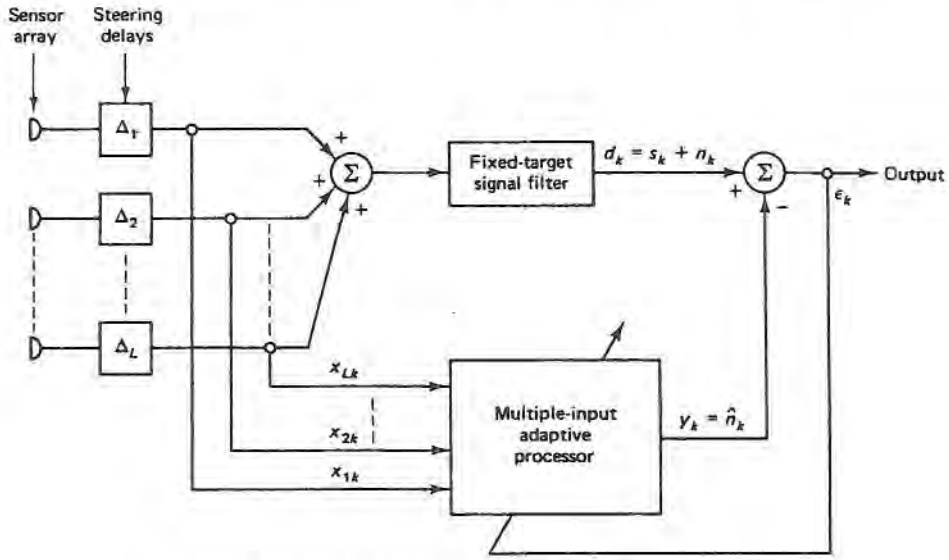

Figure 5.5 An adaptive array signal processing system.

The foregoing examples show the scope of adaptive signal processing to be considered in this chapter. We proceed to some basic concepts, adaptive algorithms, and structures that apply to all of the examples and include some specific illustrations of adaptation. Readers interested in more details can refer to a recent text [5,23,24] or a special issue of the IEEE *Proceedings* or *Transactions* devoted to adaptive signal processing, which are listed in the Reference section.

5.1 THE MEAN-SQUARE-ERROR PERFORMANCE SURFACE

Now that we have seen how Fig. 5.1 illustrates the principal signals involved in different forms of adaptation, we can discuss some basic theory. Our first objective is to see how the adaptive process, which involves minimizing the mean-square error

(MSE), $E[\epsilon_k^2]$, amounts to a search for the lowest point on a "performance surface," or "error surface," in multidimensional space.

The signal environment of an adaptive system is usually nonstationary, with the system adapting to changing signal conditions as to a slowly varying plant in Fig. 5.3 or to a drifting noise spectrum in Fig. 5.4. Changing signal properties are indeed often the principal justification for the use of adaptive processing in a given application. However, to develop basic theory, it is useful to assume temporarily that the signals are stationary and then later to study nonstationary operation [25]. We assume for now that all of the signals in Fig. 5.1 are stationary and have finite correlation functions. We also assume that the adaptive processor is a linear filter, as shown in Fig. 5.6. (Nonlinear adaptive processors [26,27] are an interesting possibility but are beyond the scope of this chapter.) The linear system has a transfer function, $W(z)$, which is adapted by adjusting weights using the adaptive algorithms described below.

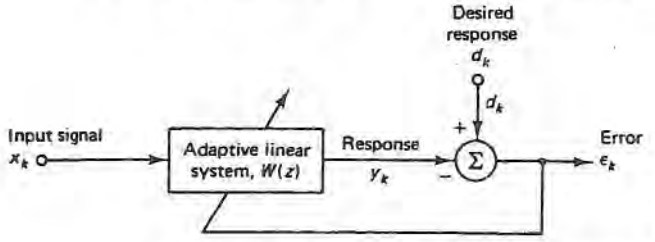

Figure 5.6 Basic adaptive system in Fig. 5.1 with a linear system as the adaptive processor.

We define the correlation functions of the signals in Fig. 5.6 as follows:†

$$\textit{Cross-correlation:} \qquad r_{dx}(n) = E[d_k x_{k+n}] \tag{5.1}$$

$$\textit{Autocorrelation:} \qquad r_{xx}(n) = E[x_k x_{k+n}] \tag{5.2}$$

With stationary signals we can say that the expectation ranges over the time index k. The remaining correlation functions of interest, $r_{dd}(n)$, $r_{yy}(n)$, and $r_{dy}(n)$, are defined similarly to Eqs. (5.1) and (5.2), and we note also that, by definition,

$$r_{xd}(n) = r_{dx}(-n) \tag{5.3}$$

Referring to Fig. 5.6 and assuming fixed filter weights for the moment so that y_k is stationary, we find that the MSE is given by

$$\begin{aligned} \text{MSE} = E[\epsilon_k^2] &= E[(d_k - y_k)^2] \\ &= E[d_k^2] + E[y_k^2] - 2E[d_k y_k] \\ &= r_{dd}(0) + r_{yy}(0) - 2r_{dy}(0) \end{aligned} \tag{5.4}$$

The z-transform of any correlation function, say $r_{uv}(n)$, is the corresponding power spectrum [28] $G_{uv}(z)$; that is,

$$G_{uv}(z) = \sum_{n=-\infty}^{\infty} r_{uv}(n) z^{-n} \tag{5.5}$$

†The notation used here, although it differs slightly from that of previous chapters, is preferred for the present development.

The inverse z-transform relationship [5] also gives

$$r_{uv}(n) = \frac{1}{2\pi j} \oint G_{uv}(z) z^n \frac{dz}{z} \qquad j \triangleq \sqrt{-1} \tag{5.6}$$

Using Eq. (5.6) with $n = 0$ in Eq. (5.4), we obtain

$$\text{MSE} = r_{dd}(0) + \frac{1}{2\pi j} \oint [G_{yy}(z) - 2G_{dy}(z)] \frac{dz}{z} \tag{5.7}$$

Assuming that $z = e^{j\omega}$ is a point on the unit circle on the z-plane corresponding to a frequency of ω rad, the power spectral relationships in Fig. 5.6 are

$$G_{yy}(z) = W(z)W(z^{-1})G_{xx}(z) \tag{5.8}$$

and

$$G_{dy}(z) = W(z)G_{dx}(z) \tag{5.9}$$

Using Eqs. (5.8) and (5.9) in Eq. (5.7), we obtain the general expression for the MSE in terms of $W(z)$:

$$\boxed{\text{MSE} = r_{dd}(0) + \frac{1}{2\pi j} \oint [W(z^{-1})G_{xx}(z) - 2G_{dx}(z)]W(z) \frac{dz}{z}} \tag{5.10}$$

In this general expression we have not specified the form of the adaptive system, $W(z)$. We have only stated that $W(z)$ is linear and has adjustable weights, which for the moment are fixed. Let

$$L = \text{Number of weights in } W(z) \tag{5.11}$$

Then Eq. (5.10) describes the MSE as an L-dimensional surface in $L + 1$-dimensional space, and adaptation becomes the process of seeking the minimum point on this surface. If we allow the weights of $W(z)$ to vary, the task is to find the minimum MSE. This is essentially the unconstrained Wiener least-squares design problem [29]. In adaptive signal processing, this task becomes a continual process of updating the weights of $W(z)$, which may be constrained, in situations where the other quantities in Eq. (5.10), $r_{dd}(0)$, $G_{xx}(z)$, and $G_{dx}(z)$, may be slowly varying.†

We will see shortly that Eq. (5.10) is greatly simplified when $W(z)$ is a finite impulse response (FIR) filter. The integral in Eq. (5.10) can be difficult when $W(z)$ is an infinite impulse response (IIR) filter, and the task of adaptation with IIR filters is in general complicated by two factors:

1. Unconstrained poles can move outside the unit circle during adaptation, causing instability.
2. The error surface in Eq. (5.10) can have flat areas and local minima, making gradient search techniques unreliable.

† Strictly speaking, these quantities are defined to be constant over time, but we think of them as varying slowly.

IIR adaptive filters have, however, been analyzed to some extent and applied in situations where poles are especially useful [30–33].

Adaptive systems with FIR filters do not have the two complications mentioned. FIR filters are also simpler and more widely applicable in adaptive processing, so we will proceed now with the assumption that the single-input adaptive filter is an FIR filter. With this assumption, we will be able to include multiple-input adaptive systems in the same analysis.

5.2 THE QUADRATIC PERFORMANCE SURFACE OF THE ADAPTIVE LINEAR COMBINER

We now assume that $W(z)$ represents a causal FIR transversal filter with L weights:

$$W(z) = \sum_{i=0}^{L-1} w_i z^{-i} \tag{5.12}$$

The causal FIR filter is applicable in real-time situations and is generally suited to a wide variety of adaptive signal processing systems. Inserting Eq. (5.12) into Eq. (5.10), we obtain

$$\text{MSE} = r_{dd}(0) + \sum_{i=0}^{L-1}\sum_{m=0}^{L-1} \frac{w_i w_m}{2\pi j} \oint G_{xx}(z)\, z^{i-m} \frac{dz}{z} - 2\sum_{i=0}^{L-1} \frac{w_i}{2\pi j} \oint G_{dx}(z) z^{-i} \frac{dz}{z} \tag{5.13}$$

Using Eq. (5.6) and also Eq. (5.3) in Eq. (5.13), we obtain the MSE for the causal FIR adaptive filter in terms of correlation functions:

$$\boxed{\text{MSE} = r_{dd}(0) + \sum_{i=0}^{L-1}\sum_{m=0}^{L-1} w_i w_m r_{xx}(i-m) - 2\sum_{i=0}^{L-1} w_i r_{xd}(i)} \tag{5.14}$$

This is the general expression for the performance surface for a causal FIR adaptive filter with given weights. We note that the MSE is a quadratic surface because the weights appear only to first and second degrees in Eq. (5.14).

We can include a class of multiple-input adaptive systems in Eq. (5.14) by using the notion of the adaptive linear combiner, illustrated in Fig. 5.7. With a single input signal, the adaptive linear combiner is an FIR filter with adjustable weights. With multiple inputs, each input signal is multiplied by its own weight. We can also imagine extending Fig. 5.7 in the multiple-input case by making each weight an adaptive FIR filter, i.e., a single-input adaptive linear combiner. This type of extension is useful, for example, in wideband adaptive array processing [5,23]. The theory covering this type of system is a fairly straightforward extension of the theory given in this chapter covering Fig. 5.7, where we use the vector $\mathbf{X}_k$ to represent either input signal at time k:

$$\textit{Single input:} \qquad \mathbf{X}_k = [x_k \quad x_{k-1} \quad \cdots \quad x_{k-L+1}]^T \tag{5.15}$$

$$\textit{Multiple inputs:} \qquad \mathbf{X}_k = [x_{0k} \quad x_{1k} \quad \cdots \quad x_{L-1,k}]^T \tag{5.16}$$

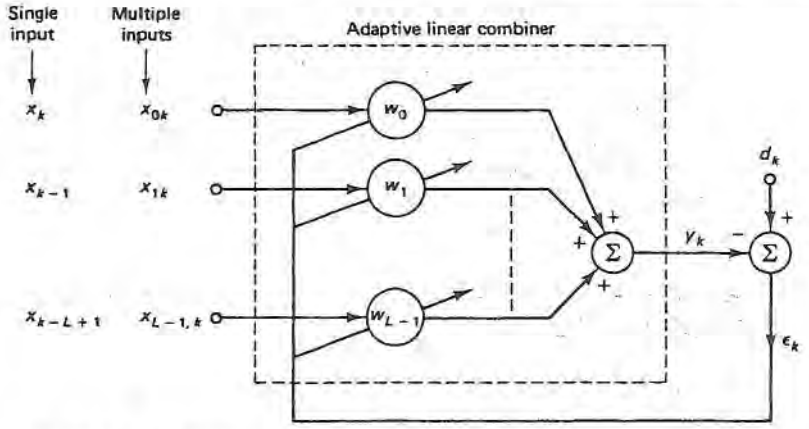

Figure 5.7 The adaptive linear combiner with single- or multiple-input signals.

To simplify Eq. (5.14) we use the symmetric input correlation matrix, or **R** matrix, given by

$$\mathbf{R} = E[\mathbf{X}_k \mathbf{X}_k^T] \tag{5.17}$$

We see from Eqs. (5.15) and (5.16) that **R** is a symmetric $L \times L$ correlation matrix and, in the single-input case, which is more widely applicable, that **R** is given by

$$\mathbf{R} = \begin{bmatrix} r_{xx}(0) & r_{xx}(1) & \cdots & r_{xx}(L-1) \\ r_{xx}(1) & r_{xx}(0) & \cdots & r_{xx}(L-2) \\ \vdots & \vdots & & \vdots \\ r_{xx}(L-1) & r_{xx}(L-2) & \cdots & r_{xx}(0) \end{bmatrix} \quad \text{(single input)} \tag{5.18}$$

The single-input **R** matrix is not only symmetric; it is also a Toeplitz matrix and relatively easy to invert if not singular [34,35]. We also need a cross-correlation vector, or **P** vector:

$$\mathbf{P} = E[d_k \mathbf{X}_k] = E[\mathbf{X}_k d_k] \tag{5.19}$$

Again, in the more common single-input case, the **P** vector is given by

$$\mathbf{P} = [r_{xd}(0) \quad r_{xd}(1) \quad \cdots \quad r_{xd}(L-1)]^T \quad \text{(single input)} \tag{5.20}$$

Finally, we need the weight vector, **W**, given by

$$\mathbf{W} = [w_0 \quad w_1 \quad \cdots \quad w_{L-1}]^T \tag{5.21}$$

With these definitions we can simplify the expression in Eq. (5.14) for the quadratic performance surface and also extend the expression to cover multiple inputs. Thus, for the adaptive linear combiner in Fig. 5.7, we have

$$\boxed{\text{MSE} = r_{dd}(0) + \mathbf{W}^T\mathbf{R}\mathbf{W} - 2\mathbf{P}^T\mathbf{W}} \tag{5.22}$$

The reader should be satisfied that Eqs. (5.22) and (5.14) are equivalent in the single-input case and that in any case Eq. (5.22) describes a quadratic performance surface. Since the MSE is always positive, we also know that the performance surface, being quadratic, must be parabolic and "concave upward," that is, extending toward a positively increasing MSE. With stationary signals, the performance surface is thus a parabolic "bowl" in $L + 1$-dimensional space. In typical applications with non-stationary, slowly varying signal statistics, we think of the bowl as being ill defined, or perhaps drifting in space as the signal properties change slowly with time, and of adaptation as the process of searching for and continuously tracking the bottom of the bowl.

When searching for the bottom of the bowl we find generally that a knowledge or estimate of the gradient is useful. The gradient vector is the column vector obtained by differentiating Eq. (5.22) with respect to the weight vector $\mathbf{W}$:

$$\begin{aligned} \boldsymbol{\nabla} &= \frac{\partial(\text{MSE})}{\partial \mathbf{W}} \\ &= \left[\frac{\partial(\text{MSE})}{\partial w_0} \; \frac{\partial(\text{MSE})}{\partial w_1} \cdots \frac{\partial(\text{MSE})}{\partial w_{L-1}} \right]^T \\ &= 2\mathbf{R}\mathbf{W} - 2\mathbf{P} \end{aligned} \tag{5.23}$$

Since the bowl is quadratic, we know that the global minimum MSE is obtained where $\boldsymbol{\nabla} = \mathbf{0}$. Setting $\boldsymbol{\nabla} = \mathbf{0}$ in Eq. (5.23) gives the optimum weight vector, $\mathbf{W}^*$, as

$$\boxed{\mathbf{W}^* = \mathbf{R}^{-1}\mathbf{P}} \tag{5.24}$$

This is the Wiener solution for the optimum weight vector [34] for the adaptive linear combiner. The corresponding minimum mean-square error is found by substituting Eq. (5.24) into Eq. (5.22) in the following manner:

$$\begin{aligned} (\text{MSE})_{\min} &= r_{dd}(0) + (\mathbf{R}^{-1}\mathbf{P})^T\mathbf{R}\mathbf{W}^* - 2\mathbf{P}^T\mathbf{W}^* \\ &= r_{dd}(0) - \mathbf{P}^T\mathbf{W}^* \end{aligned} \tag{5.25}$$

We are now ready to discuss adaptive algorithms as procedures for searching the quadratic bowl in Eq. (5.22), but first a simple two-weight example may help to clarify the nature of Eq. (5.22). We will introduce the example here and then use it throughout the remainder of this chapter in the discussion of adaptive algorithms.

5.3 AN EXAMPLE OF A QUADRATIC PERFORMANCE SURFACE

To provide an example of the quadratic performance surface of an adaptive FIR filter, we take the simple interference-canceling case illustrated in Fig. 5.8. The wideband signal and sinusoidal interference are described in the figure. We might, for instance, be trying to use this system to cancel power line interference in the signal from a biological sensor or some other type of sensor producing a low-level signal. (In a real

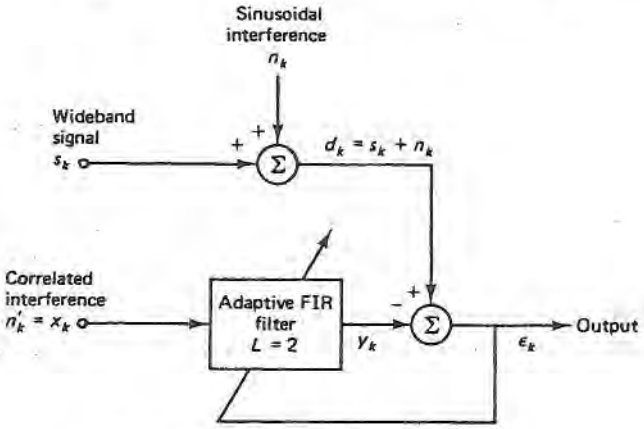

Figure 5.8 A simple interference-canceling example used to illustrate the quadratic performance surface.

situation, we would not need to know the correlation properties of s_k, n_k, or n'_k exactly.) In this simple example, the signal and interference are uncorrelated. The correlation functions used to obtain the MSE are found by averaging over one period (16 samples) of the sinusoids:

$$r_{xx}(i) = \frac{1}{16}\sum_{k=0}^{15}\left(\sqrt{2}\sin\frac{2\pi k}{16}\right)\left(\sqrt{2}\sin\frac{2\pi(k+i)}{16}\right) = \cos\frac{2\pi i}{16} \tag{5.26}$$

$$\begin{aligned} r_{xd}(i) &= \frac{1}{16}\sum_{k=0}^{15}\left(\sqrt{2}\sin\frac{2\pi k}{16}\right)\left[\sin\left(\frac{2\pi(k+i)}{16} + \frac{\pi}{10}\right)\right] \\ &= \frac{1}{\sqrt{2}}\cos\left(\frac{2\pi i}{16} + \frac{\pi}{10}\right) \end{aligned} \tag{5.27}$$

$$r_{dd}(0) = r_{ss}(0) + r_{nn}(0) = 0.55 \tag{5.28}$$

We note in these results that $r_{dd}(0)$ is just the power in s_k plus the power in n_k, i.e., $E[s_k^2] + E[n_k^2]$, and that these two quantities are 0.05 and 0.5, respectively, so the input signal-to-noise ratio is 0.1. Using Eqs. (5.26)–(5.28) in Eq. (5.14) with $L = 2$, we have the MSE in this case as follows:

$$\text{MSE} = 0.55 + w_0^2 + w_1^2 + 2w_0w_1\cos\frac{\pi}{8} - \sqrt{2}\,w_0\cos\frac{\pi}{10} - \sqrt{2}\,w_1\cos\frac{9\pi}{40} \tag{5.29}$$

A three-dimensional plot of a part of this performance surface is shown in Fig. 5.9. The surface cannot be seen very accurately, but we can note its quadratic form and also that the "bowl" in this case is very noncircular in cross section. The noncircular cross section is due to the dissimilar eigenvalues of the **R** matrix, which in this case are 1.92 and 0.08. In general, when the eigenvalues of the **R** matrix are dissimilar,

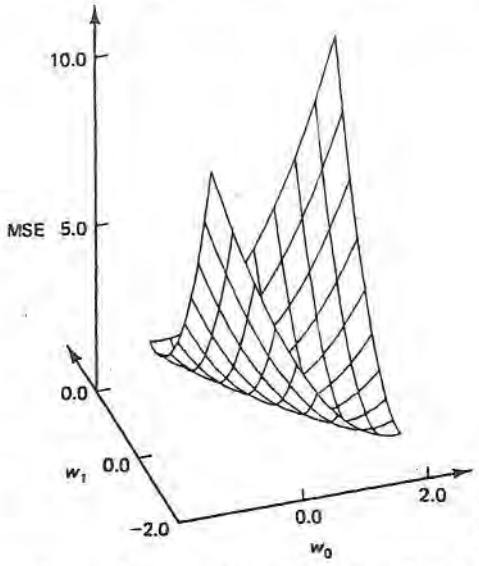

Figure 5.9 Part of the two-dimensional performance surface in Eq. (5.29) for the interference-canceling example in Fig. 5.8.

the quadratic bowl tends to be elliptical so that the gradient varies as we move around the bowl at constant MSE [5]. This property has important consequences for gradient search algorithms, as we will see.

We can view the parabolic surface in Fig. 5.9 in a different way by looking at projections of constant-MSE contours onto the w_0w_1-plane, as in Fig. 5.10. Here we see that the contours are all ellipses, showing more clearly the quadratic form of Eq. (5.29). From Eq. (5.24) we can see that the optimum weight values in this example are

$$\begin{aligned}\begin{bmatrix} w_0^* \\ w_1^* \end{bmatrix} &= \begin{bmatrix} r_{xx}(0) & r_{xx}(1) \\ r_{xx}(1) & r_{xx}(0) \end{bmatrix}^{-1} \begin{bmatrix} r_{xd}(0) \\ r_{xd}(1) \end{bmatrix} \\ &= \begin{bmatrix} 1 & \cos\dfrac{\pi}{8} \\ \cos\dfrac{\pi}{8} & 1 \end{bmatrix}^{-1} \begin{bmatrix} \dfrac{1}{\sqrt{2}}\cos\dfrac{\pi}{10} \\ \dfrac{1}{\sqrt{2}}\cos\dfrac{9\pi}{40} \end{bmatrix} = \begin{bmatrix} 1.20000 \\ -0.57099 \end{bmatrix}\end{aligned} \tag{5.30}$$

These weight values are seen at the projection of the "bottom of the bowl" in Fig. 5.10. From Eq. (5.25), the minimum MSE when the weights are optimized as in Eq. (5.30) is

$$\begin{aligned}(\text{MSE})_{\min} &= r_{dd}(0) - [r_{xd}(0) \quad r_{xd}(1)] \begin{bmatrix} w_0^* \\ w_1^* \end{bmatrix} \\ &= 0.55 - \frac{1.20000}{\sqrt{2}}\cos\frac{\pi}{10} + \frac{0.57099}{\sqrt{2}}\cos\frac{9\pi}{40} = 0.05\end{aligned} \tag{5.31}$$

Thus, when the weights are optimized, y_k exactly cancels n_k as we might have anticipated from Fig. 5.8, and the output, $\epsilon_k = s_k$, is free of sinusoidal interference. We note, in this ideal example, that the signal-to-noise ratio improves from 0.1 to infinity.

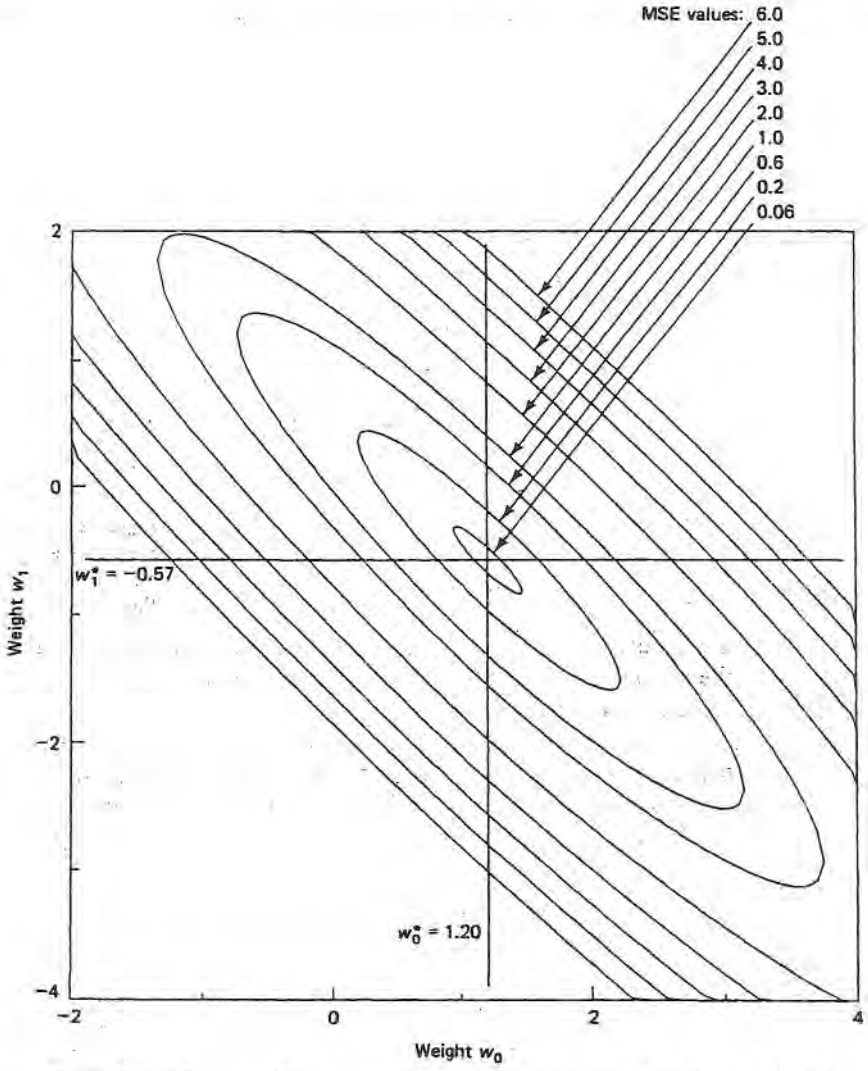

Figure 5.10 Contours of constant MSE on the performance surface in Fig. 5.9. The minimum MSE is 0.05 at $w_0^* = 1.20$ and $w_1^* = -0.57$.

5.4 SEARCH METHODS AND ADAPTIVE ALGORITHMS

So far we have viewed the adaptive process as that of searching the MSE performance surface for its global minimum or of tracking this global minimum as it changes with time. In Eq. (5.10) we saw that the performance surface could, in general, have local minima. Thus the most general adaptive algorithms for searching the performance surface under unknown signaling conditions are random search algorithms. One type of random search algorithm [36] selects a random direction in which to "look" at each iteration. An even more general random search algorithm [37] selects points at random

locations on the weight plane (or hyperplane), estimates the MSE at these points, and retains points with the lowest MSE from one iteration to the next.

However, as we have stated, we are limiting our discussion in this chapter mainly to the quadratic performance surface of the adaptive linear combiner used in FIR adaptive filters. With this type of surface we can implement powerful, deterministic search methods. These methods generally are based on making local estimates of the gradient, $\hat{\nabla}$, and moving incrementally downward toward the bottom of the bowl. Thus, one adaptive cycle or adaptive iteration consists essentially of determining $\hat{\nabla}_k$ at time k and then making an appropriate adjustment in the weight vector from $\mathbf{W}_k$ to $\mathbf{W}_{k+1}$ in order to move toward $\mathbf{W}^*$ at the bottom of the bowl.

Our adaptive algorithms will be versions of two basic search techniques, steepest descent and Newton's method, which in turn result from a basic property of any parabolic surface. The property can be seen by multiplying the gradient formula in Eq. (5.23) by $\frac{1}{2}\mathbf{R}^{-1}$ to obtain

$$\tfrac{1}{2}\mathbf{R}^{-1}\nabla = \mathbf{W} - \mathbf{R}^{-1}\mathbf{P} \tag{5.32}$$

We can now combine this result with Eq. (5.24) to obtain

$$\boxed{\mathbf{W}^* = \mathbf{W} - \tfrac{1}{2}\mathbf{R}^{-1}\nabla} \tag{5.33}$$

This result is essentially Newton's root-finding method [38] applied to find the zero of a linear function (∇ in this case). It can also be derived by writing the MSE in the form of a Taylor series, as in [39]. Given any weight vector $\mathbf{W}$ along with $\mathbf{R}$ and the corresponding gradient ∇, we can move from $\mathbf{W}$ to the optimum weight vector $\mathbf{W}^*$ in a single step by adjusting $\mathbf{W}$ in accordance with Eq. (5.33).

If we could apply Eq. (5.33) in practical situations such as those illustrated in Figs. 5.2–5.5, we would always be able to adapt in one iteration to the optimal weight vector, and the adaptive process for the quadratic performance surface would be simple (but uninteresting). As in nature, however, practical adaptive signal processing systems do not have enough information to adapt perfectly in just one iteration. There are two specific problems with Eq. (5.33) in practice, under nonstationary conditions:

1. The $\mathbf{R}$ matrix is unknown and can at best only be estimated.
2. The gradient must be estimated at each adaptive iteration, using local statistics.

From Eqs. (5.17) and (5.19), we note that these two problems are directly related to the fact that the signal correlation functions are unknown and must be estimated.

To anticipate the use of noisy estimates of $\mathbf{R}^{-1}$ and ∇ in Eq. (5.33), we can modify Eq. (5.33) to obtain an algorithm that adjusts $\mathbf{W}$ in smaller increments and converges to $\mathbf{W}^*$ after many iterations. The small increments will have the effect of smoothing the noise in the estimates of $\mathbf{R}^{-1}$ and ∇, thus allowing the adaptive system to have stable behavior. The modified version of Eq. (5.33) is

$$\boxed{\mathbf{W}_{k+1} = \mathbf{W}_k - \mu\mathbf{R}^{-1}\nabla_k} \tag{5.34}$$

Under noise-free conditions with $\mathbf{R}$ and ∇_k known exactly, the convergence of Eq. (5.34) depends only on the scalar convergence factor, u. We can see this by substituting Eq. (5.23) and then Eq. (5.24) into Eq. (5.34):

$$\begin{aligned} \mathbf{W}_{k+1} &= \mathbf{W}_k - u\mathbf{R}^{-1}(2\mathbf{R}\mathbf{W}_k - 2\mathbf{P}) \\ &= (1 - 2u)\mathbf{W}_k + 2u\mathbf{W}^* \end{aligned} \tag{5.35}$$

Using Eq. (5.35) in itself, we arrive by induction at an expression for the relaxation of Eq. (5.34) from $\mathbf{W}_0$ toward the optimum weight vector, $\mathbf{W}^*$:

$$\begin{aligned} \mathbf{W}_{k+1} &= (1 - 2u)\mathbf{W}_k + 2u\mathbf{W}^* \\ &= (1 - 2u)^2\mathbf{W}_{k-1} + 2u\mathbf{W}^*[1 + (1 - 2u)] \\ &= (1 - 2u)^3\mathbf{W}_{k-2} + 2u\mathbf{W}^*[1 + (1 - 2u) + (1 - 2u)^2] \\ &\vdots \\ &= (1 - 2u)^{k+1}\mathbf{W}_0 + 2u\mathbf{W}^* \sum_{i=0}^{k} (1 - 2u)^i \end{aligned} \tag{5.36}$$

Summing the series in Eq. (5.36) and using k in place of $k + 1$, we obtain

$$\begin{aligned} \mathbf{W}_k &= (1 - 2u)^k\mathbf{W}_0 + \frac{2u[1 - (1 - 2u)^k]}{1 - (1 - 2u)}\mathbf{W}^* \\ &= (1 - 2u)^k\mathbf{W}_0 + [1 - (1 - 2u)^k]\mathbf{W}^* \end{aligned} \tag{5.37}$$

In this result we can see that $\mathbf{W}_k$ will converge to $\mathbf{W}^*$ at $k = \infty$ in general only if the geometric ratio $1 - 2u$ is less than 1 in magnitude. In other words, the rule is

$$\boxed{\text{For noise-free convergence:} \quad 0 < u < 1} \tag{5.38}$$

In Eq. (5.33) the one-step, noise-free solution is found with $u = 1/2$. With $u > 1/2$ the convergence is oscillatory, with $\mathbf{W}_k$ jumping "back and forth" across the bowl and converging toward $\mathbf{W}^*$. In practical adaptive systems with noise, values of u well below $\frac{1}{2}$, typically on the order of 0.01, are used, giving a convergence time constant of many iterations. We define the convergence time constant to be τ iterations, where the relaxation toward $\mathbf{W}^*$ goes in proportion to $(1 - e^{-k/\tau})$. Then, using the linear approximation $(1 - 1/\tau)$ for $e^{-1/\tau}$ when τ is large and comparing with Eq. (5.37), we have

$$\begin{aligned} 1 - e^{-k/\tau} &\approx 1 - (1 - 1/\tau)^k \\ &\approx 1 - (1 - 2u)^k \end{aligned} \tag{5.39}$$

From this we have the weight-vector convergence time constant for Newton's method in terms of the adaptive gain parameter u:

Time constant for weight-vector convergence under noise-free conditions:

$$\tau_w \approx \frac{1}{2u}, \qquad 0 < u \ll 1/2 \tag{5.40}$$

In addition to the weight-vector convergence described in Eq. (5.37), the convergence of the MSE toward its minimum value is another process commonly used as a performance measure in adaptive systems. This convergence is known euphemistically as "learning," and a plot of the MSE versus the iteration number k is called a "learning curve" [4]. To derive a formula similar to Eq. (5.37) for the learning curve, it is convenient to use translated weight-vector coordinates, so we define the translated weight vector $\mathbf{V}$ as the deviation from the optimum weight vector:

$$\mathbf{V} = \mathbf{W} - \mathbf{W}^* \tag{5.41}$$

The minimum MSE is seen to be at $\mathbf{V} = \mathbf{0}$ in this translated system. (For example, the centered axes in Fig. 5.10 are the v_0 and v_1 axes.) Using Eq. (5.41) in Eq. (5.22) for the MSE, we obtain

$$\begin{aligned} \text{MSE} &= r_{dd}(0) + (\mathbf{V} + \mathbf{W}^*)^T\mathbf{R}(\mathbf{V} + \mathbf{W}^*) - 2\mathbf{P}^T(\mathbf{V} + \mathbf{W}^*) \\ &= r_{dd}(0) + \mathbf{W}^{*T}\mathbf{R}\mathbf{W}^* - 2\mathbf{P}^T\mathbf{W}^* + \mathbf{V}^T\mathbf{R}\mathbf{V} + \mathbf{V}^T\mathbf{R}\mathbf{W}^* + \mathbf{W}^{*T}\mathbf{R}\mathbf{V} - 2\mathbf{P}^T\mathbf{V} \end{aligned} \tag{5.42}$$

From Eq. (5.24) and the first line of Eq. (5.25) we see that $(\text{MSE})_{\min}$ can be substituted for the first three terms in Eq. (5.42). Also, using Eq. (5.24) plus the fact that any scalar is equal to its own transpose, we see that the last three terms in Eq. (5.42) cancel to zero. Hence, Eq. (5.42) reduces to

$$\text{MSE} = (\text{MSE})_{\min} + \mathbf{V}^T\mathbf{R}\mathbf{V} \tag{5.43}$$

This result is often preferred over Eq. (5.22) as a general expression for the quadratic performance surface. To obtain the learning curve expression from Eq. (5.43), we use Eq. (5.41) to translate Eq. (5.37) and obtain a simpler result:

$$\mathbf{V}_k = (1 - 2u)^k\mathbf{V}_0 \tag{5.44}$$

Substituting Eq. (5.44) into Eq. (5.43) expressed at the kth iteration, we have the learning curve formula for Newton's method:

$$(\text{MSE})_k = (\text{MSE})_{\min} + (1 - 2u)^{2k}\mathbf{V}_0^T\mathbf{R}\mathbf{V}_0 \tag{5.45}$$

This result gives us a closed expression for the learning curve with Newton's method under noise-free conditions. Comparing it with Eq. (5.37) or (5.44), we see that the

geometric convergence ratio is now $(1 - 2u)^2$ instead of $(1 - 2u)$. Thus, in a development similar to Eq. (5.39), the approximate geometric ratio is $\exp(-1/\tau_{MSE}) = \exp(-2/\tau_w)$, where τ_{MSE} is the learning curve time constant. So our result similar to Eq. (5.40) for the learning curve is

Time constant for MSE convergence with Newton's method under noise-free conditions:

$$\tau_{MSE} = \tau_w/2 \approx 1/4u, \qquad 0 < u \ll 1/2 \tag{5.46}$$

Examples of weight convergence and a learning curve are given in Figs. 5.11 and 5.12. These examples represent the noise-free application of Eq. (5.34), starting at $w_{00} = 3$ and $w_{10} = -4$ and using $u = 0.04$, well below the one-step convergence value, $u = 0.5$. The examples are for the interference-canceling example of Fig. 5.8, with performance surface illustrated in Figs. 5.9 and 5.10. In Fig. 5.11, note the

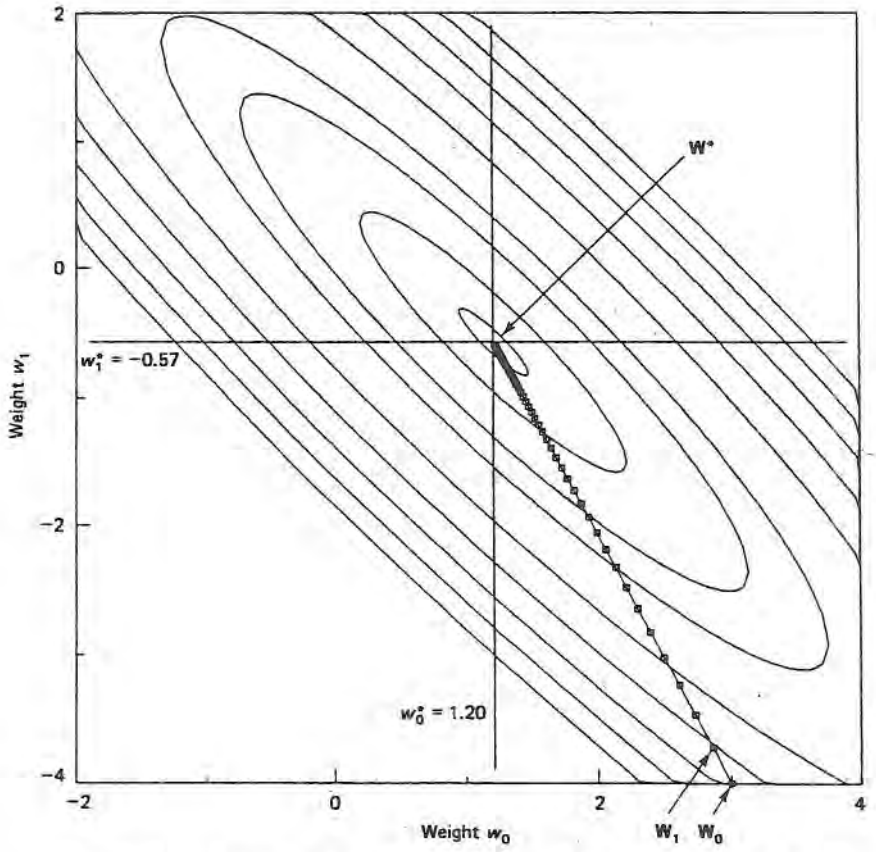

Figure 5.11 Newton weight track, showing noise-free convergence from $\mathbf{W}_0 = (3, -4)^T$ to near $\mathbf{W}^*$ for the interference-canceling system in Fig. 5.8. The adaptive gain in Eq. (5.34) was $u = 0.04$. The weight track is shown for $k = 0, 1, \ldots, 60$.

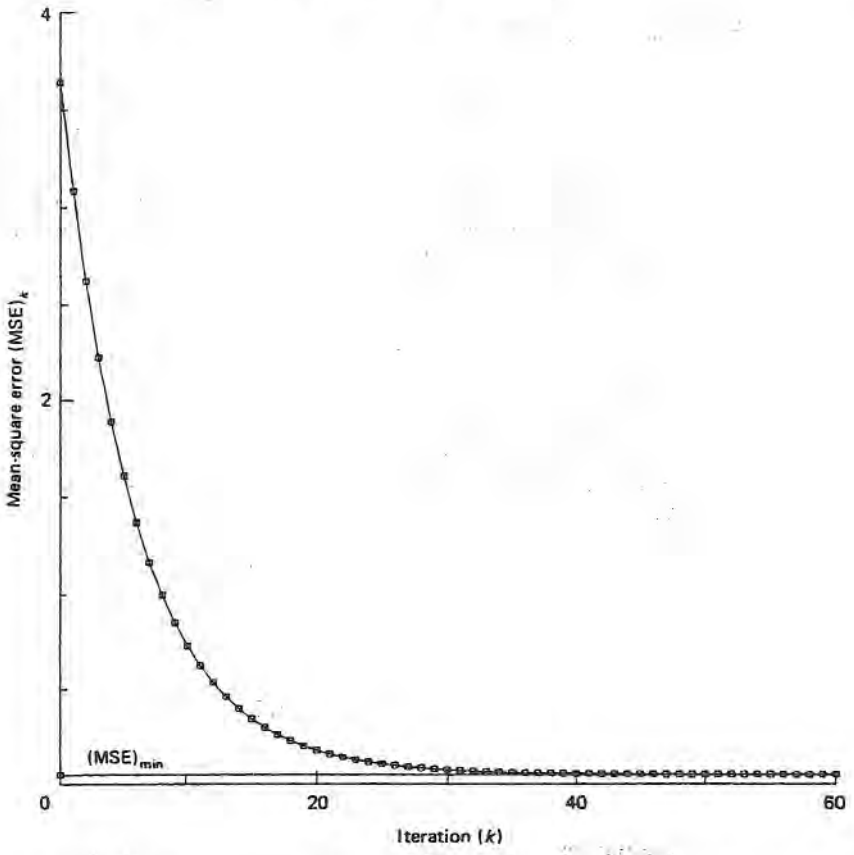

Figure 5.12 Learning curve of MSE versus iteration number, corresponding with the weight track in Fig. 5.11.

straight, noise-free track from $\mathbf{W}_0$ to $\mathbf{W}^*$ and also the translated v_0v_1-coordinate system with its origin at $\mathbf{W}^*$. With $u = 0.04$, the weight convergence time constant is approximately 12 iterations in accordance with Eq. (5.40), and we note in Fig. 5.11 that there is convergence after 5 time constants, or 60 iterations. In Fig. 5.12, we see the corresponding exponential decrease of the MSE from its starting value of approximately 3.64 toward the minimum value, 0.05, in Eq. (5.31). The MSE time constant, τ_{MSE}, should be approximately 6 iterations in accordance with Eq. (5.46), and we observe in Fig. 5.12 that the approximation is valid. After 4 time constants, or 24 iterations in this example, the MSE has relaxed to approximately e^{-4}, or just under 2% of the original excess MSE, given by $(MSE)_0 - (MSE)_{min}$.

In this discussion of adaptive algorithms for quadratic performance surfaces, the principal formula is the Newton type of algorithm in Eq. (5.34). We must now examine how to apply this formula with imperfect knowledge of the gradient and the $\mathbf{R}$ matrix. First, we will assume that we have no knowledge of the off-diagonal elements of the $\mathbf{R}$ matrix. This assumption leads us to the "steepest-descent" class of algorithms and in turn to the widely used Widrow-Hoff LMS algorithm, which is the simplest and most useful algorithm in adaptive signal processing.

5.5 STEEPEST DESCENT AND THE LMS ALGORITHM

The least-mean-square (LMS) algorithm [4] is the simplest and most widely applicable algorithm in adaptive signal processing. It is a steepest-descent type of algorithm, that is, an algorithm whose track follows (on the average) the negative gradient of the performance surface. As mentioned above, such an algorithm does not require the complete $\mathbf{R}$ matrix. To revise Eq. (5.34) into a steepest-descent type of algorithm, we define $\hat{\nabla}_k$ to be an estimate of ∇_k and write

$$\mathbf{W}_{k+1} = \mathbf{W}_k - \mu \hat{\nabla}_k \tag{5.47}$$

Here we can see that the increment from $\mathbf{W}_k$ to $\mathbf{W}_{k+1}$ is in the estimated negative gradient direction, so the weight track will follow approximately a steepest-descent path on the performance surface. The parameter μ in Eq. (5.47) is the parameter used by Widrow and others [2] in the LMS algorithm. It can be related to the parameter u in Eq. (5.34) by noting that the elements of the $\mathbf{R}$ matrix have dimensions of input power, as seen in Eq. (5.17). We can define

$$\begin{aligned} \sigma^2 &= \text{Average signal power to adaptive linear combiner} \\ &= \frac{1}{L} E[\mathbf{X}_k^T \mathbf{X}_k] = \frac{1}{L} Tr[\mathbf{R}] \\ &= E[x_k^2] \qquad \text{(single input)} \\ &= \frac{1}{L} \sum_{i=0}^{L-1} E[x_{ik}^2] \qquad \text{(multiple inputs)} \end{aligned} \tag{5.48}$$

Suppose we have a single white input signal with power σ^2, or multiple white inputs each with power σ^2. Then the $\mathbf{R}$ matrix, $\mathbf{R} = \sigma^2\mathbf{I}$, has all L eigenvalues equal to σ^2, and the constant-MSE contours given by Eq. (5.22) are "circular" in $(L + 1)$-dimensional space. For this case, comparing Eq. (5.47) with Eq. (5.34), we have

$$\textit{Equal eigenvalues:} \qquad \mu = \frac{u}{\sigma^2}, \qquad 0 < u < 1, \; 0 < \mu < \frac{1}{\sigma^2} \tag{5.49}$$

For the general case where the eigenvalues are not equal, it is then reasonable to use the *maximum* eigenvalue, λ_{max}, in place of σ^2 in Eq. (5.49) to give a conservative range on μ corresponding with $0 < u < 1$ and ensure convergence of the steepest-descent algorithm, Eq. (5.47). Widrow [2] has proved this conjecture for convergence in the mean. However, the eigenvalues of $\mathbf{R}$ are not usually known in practice, so the *sum* of eigenvalues, which is the same as the trace of $\mathbf{R}$, is used instead. Thus, from Eq. (5.48), we use $L\sigma^2$ in place of λ_{max} and have for the general case

$$\textit{Unequal eigenvalues:} \qquad \mu = \frac{u}{Tr[\mathbf{R}]} = \frac{u}{L\sigma^2}, \qquad 0 < u < 1, \; 0 < \mu < \frac{1}{L\sigma^2} \tag{5.50}$$

This more general bound on μ has also been derived by Widrow [3]. From Eqs. (5.50) and (5.47) the steepest-descent version of Eq. (5.34) for the general case now becomes

$$\mathbf{W}_{k+1} = \mathbf{W}_k - \frac{u}{L\sigma^2}\hat{\nabla}_k \tag{5.51}$$

We note that the implementation of this result implies knowing the average input signal power σ^2. In practice a system for estimating σ^2 or tracking σ^2 under nonstationary conditions may be required. For example, a simple smoothing filter for estimating σ^2 in the single-input case is shown in Fig. 5.13. If we use the algorithm in the figure recursively, we obtain

$$\hat{\sigma}_k^2 = \alpha x_k^2 + (1 - \alpha)\hat{\sigma}_{k-1}^2 \tag{5.52}$$

$$= \alpha x_k^2 + (1 - \alpha)[\alpha x_{k-1}^2 + (1 - \alpha)\hat{\sigma}_{k-2}^2]$$

$$\vdots$$

$$= \alpha \sum_{i=0}^{k} (1 - \alpha)^i x_{k-i}^2 \tag{5.53}$$

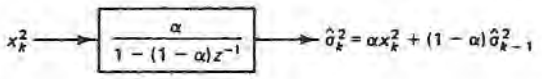

Figure 5.13 Smoothing filter for estimating and tracking the input signal power, $\sigma^2 = E[x_k^2]$.

Using a development similar to Eq. (5.39), we see from Eq. (5.53) that the operation in Fig. 5.13 amounts to "forgetting" the past values of x_k^2 exponentially, with the time constant

$$\tau_{\sigma^2} \approx \frac{1}{\alpha} \text{ iterations}, \qquad 0 < \alpha \ll 1 \tag{5.54}$$

Unless the input signal power is known, some type of estimate like the one in Fig. 5.13 is needed to compensate for the missing $\mathbf{R}$ matrix in the steepest-descent type of algorithm in Eq. (5.51). The initial estimate, $\hat{\sigma}_0^2$, would normally be the best *a priori* estimate of σ^2.

An example of noise-free steepest descent with $\hat{\nabla}_k = \nabla_k$ is shown in Fig. 5.14 for comparison with Fig. 5.11. The only difference between the two figures is the use of Eq. (5.51) for Fig. 5.14 in place of Eq. (5.34) for Fig. 5.11. The terms in Eq. (5.51) for this case are $L = 2$ and $\sigma^2 = 1.0$ from Fig. 5.8, with ∇_k found by taking the gradient of Eq. (5.29). The value of $u = 0.2$ was used in Fig. 5.14. The principle feature of Fig. 5.14 is the direction of the weight track, which proceeds normal to the constant-MSE contours and is consequently of greater length compared with the track in Fig. 5.11. We note that when the error contours are highly elliptical, as they are here, the steepest-descent path can be much longer than the Newton path. However, if the error contours are circles, the two paths are the same. The ellipticity of the contours is determined by the eigenvalues of the $\mathbf{R}$ matrix [5], which, as we have noted, are not usually known in practice.

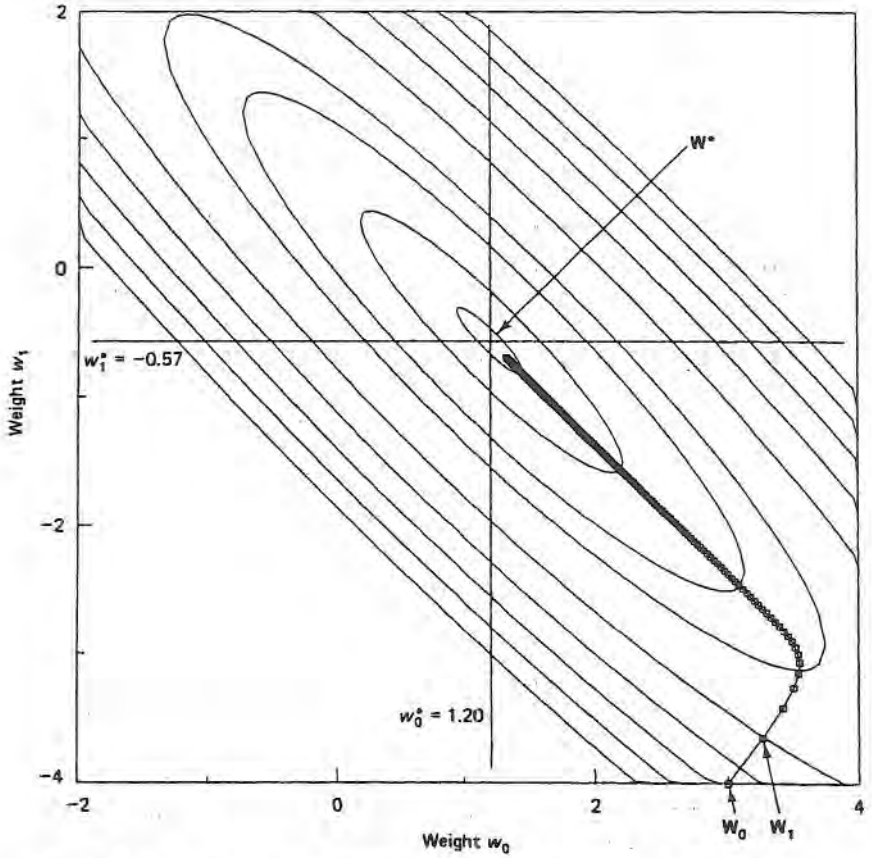

Figure 5.14 Steepest-descent weight track, showing noise-free ($\hat{\nabla}_k = \nabla_k$) convergence from $\mathbf{W}_0 = (3, -4)^T$ to near $\mathbf{W}^*$ for the interference-canceling system in Fig. 5.8. The adaptive gain parameter in Eq. (5.51) was $\mu = 0.2$. The weight track is shown for $k = 0, 1, \ldots, 200$.

We need one final modification of Eq. (5.47) to get from steepest descent to the LMS algorithm. Instead of the gradient at each iteration, ∇_k, which is not usually known in practice, we use a simple estimate, $\hat{\nabla}_k$. This estimate is in turn based on using the instantaneous squared error ϵ_k^2 as an estimate of the mean-square error $E[\epsilon_k^2]$. From the definition of ϵ_k as illustrated in Fig. 5.7, we can see that

$$\epsilon_k = d_k - \mathbf{W}_k^T \mathbf{X}_k \tag{5.55}$$

In Eq. (5.23) we have the gradient given as

$$\nabla_k = \frac{\partial(\text{MSE})}{\partial \mathbf{W}_k} \tag{5.56}$$

Therefore, using ϵ_k^2 as an estimate of the MSE, we have

$$\hat{\nabla}_k = \frac{\partial \epsilon_k^2}{\partial \mathbf{W}_k} = 2\epsilon_k \frac{\partial \epsilon_k}{\partial \mathbf{W}_k} = -2\epsilon_k \mathbf{X}_k \tag{5.57}$$

We can show that Eq. (5.57) is an unbiased gradient estimate by substituting Eq. (5.55) for ϵ_k and taking the expected value. The result is $E[\hat{\nabla}_k] = \nabla$ in Eq. (5.23).

The result of substituting from Eq. (5.57) for $\hat{\nabla}_k$ in Eq. (5.51) is

$$\boxed{\mathbf{W}_{k+1} = \mathbf{W}_k + \frac{2u}{L\sigma^2}\epsilon_k \mathbf{X}_k, \qquad 0 < u < 1} \tag{5.58}$$

This is the LMS algorithm. We prefer the normalized constant u over μ in Eq. (5.47) to emphasize the need to estimate input power and also because the allowable range for u is easy to remember.

An example using the LMS algorithm in the system of Fig. 5.8 is shown in Fig. 5.15 for comparison with Figs. 5.11 and 5.14. The conditions are exactly the same as for Fig. 5.14 except that Eq. (5.58) is used instead of Eq. (5.51). We note in Fig. 5.15 that the weight track is now noisy due to the local gradient estimated in the LMS algorithm.

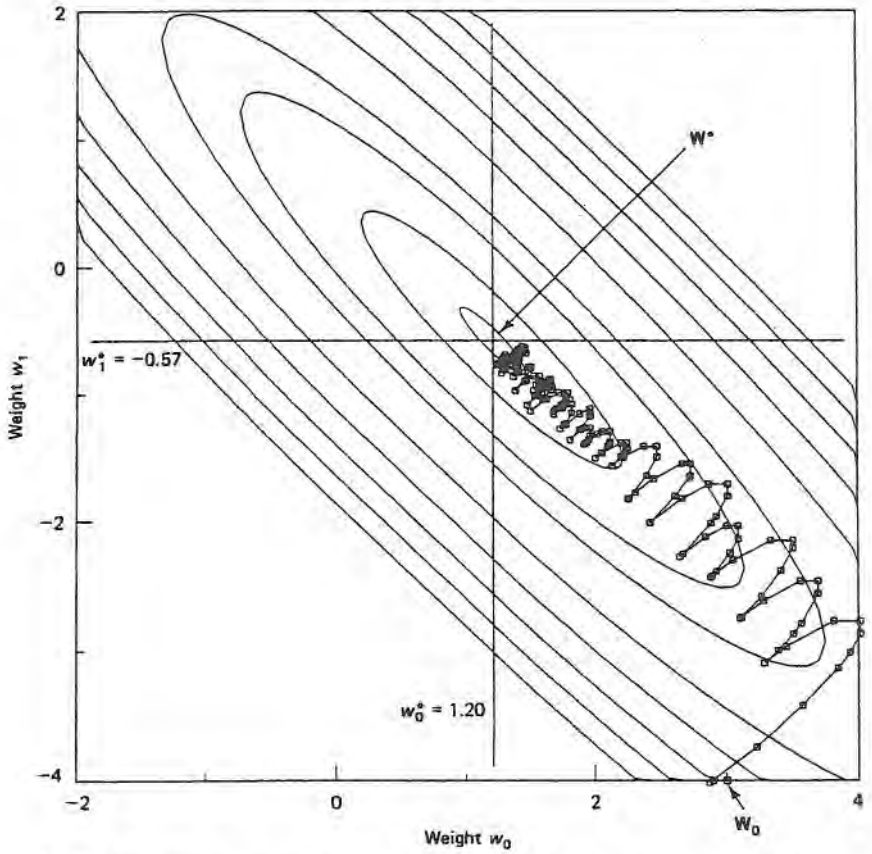

Figure 5.15 LMS weight track, showing convergence from $\mathbf{W}_0 = (3, -4)^T$ to near $\mathbf{W}^*$ for the interference-canceling system in Fig. 5.8. The adaptive gain parameter in Eq. (5.58) was $u = 0.2$. The weight track is shown for $k = 0, 1, \ldots, 200$.

A plot of the output ϵ_k versus k is shown in Fig. 5.16. This plot corresponds with the weight track in Fig. 5.15. In Fig. 5.16 we can see the sinusoidal interference gradually disappearing from the output as k increases, with $\epsilon_k \approx s_k$ when k reaches 200, which is of course the desired result in Fig. 5.8. The effect of this disappearance of the sinusoidal component of ϵ_k can also be seen in the weight track in Fig. 5.15.

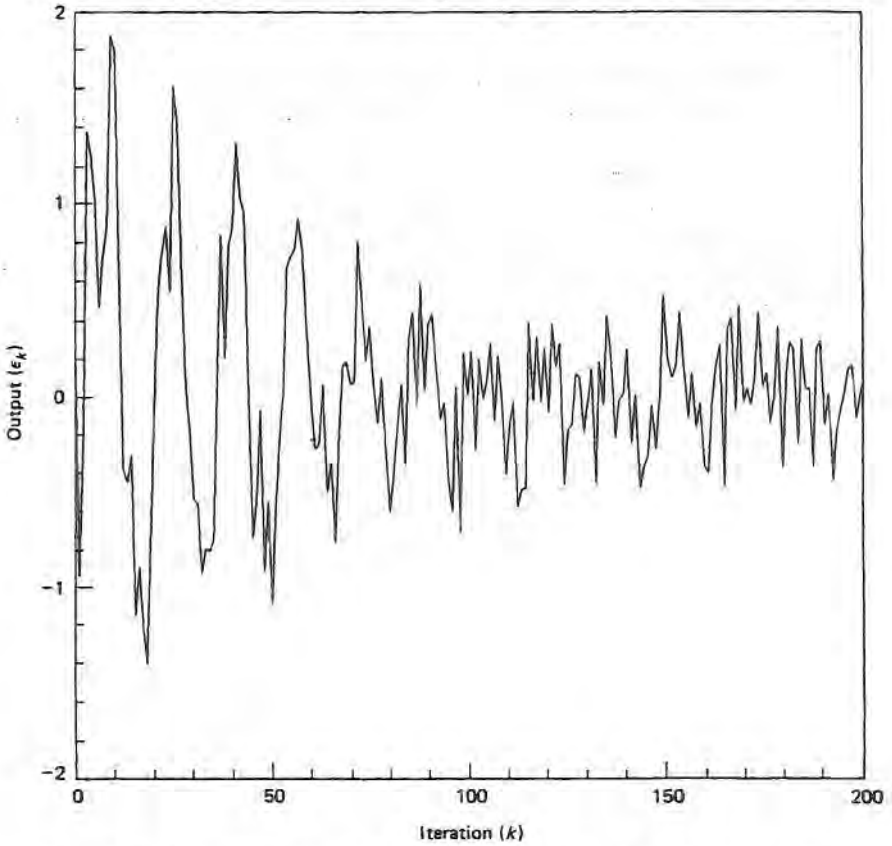

Figure 5.16 Output ϵ_k versus k for the LMS algorithm applied in Fig. 5.8, showing cancellation of the sinusoidal interference component.

The LMS algorithm is the simplest and most widely applicable adaptive algorithm. All of the terms in the weight-vector increment in Eq. (5.58) can be made available in a wide variety of applications. When the input power σ^2 to the adaptive linear combiner varies during adaptation, an algorithm such as Eq. (5.52) must be used together with Eq. (5.58) to provide a current estimate σ_k^2 to use in place of σ^2.

5.6 AN LMS-NEWTON ALGORITHM

We have seen that the LMS algorithm is a steepest-descent type of algorithm that, as in Eq. (5.47), uses an estimate of the gradient vector but no estimate of the $\mathbf{R}$ matrix. A Newton type of algorithm that "steers" the weight-vector increment and generates

a straighter path to $\mathbf{W}^*$ on the performance surface (as in the ideal case of Fig. 5.11 compared with the case of Fig. 5.14) must contain an estimate of $\mathbf{R}^{-1}$ in addition to the gradient estimate. That is, from Eq. (5.34), a Newton type of algorithm using estimates of $\mathbf{R}^{-1}$ and ∇_k would be

$$\mathbf{W}_{k+1} = \mathbf{W}_k - \mu \hat{\mathbf{R}}_k^{-1} \hat{\nabla}_k \tag{5.59}$$

To see how we might compute $\hat{\mathbf{R}}_k^{-1}$ in Eq. (5.59), let us first consider a scheme [5] similar to Fig. 5.13 for estimating the entire $\mathbf{R}$ matrix instead of just the input power. The result is shown in Fig. 5.17. Just as in Eqs. (5.52)–(5.54) we have, from Fig. 5.17,

$$\hat{\mathbf{R}}_k = (1 - \alpha)\hat{\mathbf{R}}_{k-1} + \alpha \mathbf{X}_k \mathbf{X}_k^T \tag{5.60}$$

$$= \alpha \sum_{i=0}^{k} (1 - \alpha)^i \mathbf{X}_{k-i} \mathbf{X}_{k-i}^T \tag{5.61}$$

$$\tau_R \approx \frac{1}{\alpha} \text{ iterations}, \qquad 0 < \alpha \ll 1 \tag{5.62}$$

Here again, we are weighting the past input values exponentially so that they are forgotten approximately with a time constant equal to τ_R iterations. This is a natural way to estimate the $\mathbf{R}$ matrix, at least when the signal $\mathbf{X}_k$ is stationary enough to allow τ_R to be large compared with the periods of the dominant frequency components of $\mathbf{X}_k$. (For an alternative method, see [40].)

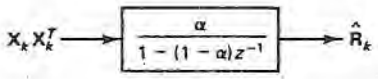

Figure 5.17 Smoothing filter for estimating the $\mathbf{R}$ matrix, similar to the filter shown in Fig. 5.13.

Assuming that Eq. (5.60) gives an acceptable estimate of $\mathbf{R}$, we now proceed to change Eq. (5.60) into an equivalent estimate of $\mathbf{R}^{-1}$ to remove the need for a matrix inversion at each iteration of Eq. (5.59). First we premultiply Eq. (5.60) by $\hat{\mathbf{R}}_k^{-1}$, then postmultiply by $\hat{\mathbf{R}}_{k-1}^{-1}$, and obtain

$$\hat{\mathbf{R}}_{k-1}^{-1} = (1 - \alpha)\hat{\mathbf{R}}_k^{-1} + \alpha \hat{\mathbf{R}}_k^{-1} \mathbf{X}_k \mathbf{X}_k^T \hat{\mathbf{R}}_{k-1}^{-1} \tag{5.63}$$

Postmultiplying this result by $\mathbf{X}_k$ gives

$$\begin{aligned} \hat{\mathbf{R}}_{k-1}^{-1} \mathbf{X}_k &= (1 - \alpha)\hat{\mathbf{R}}_k^{-1} \mathbf{X}_k + \alpha \hat{\mathbf{R}}_k^{-1} \mathbf{X}_k \mathbf{X}_k^T \hat{\mathbf{R}}_{k-1}^{-1} \mathbf{X}_k \\ &= \hat{\mathbf{R}}_k^{-1} \mathbf{X}_k (1 - \alpha + \alpha \mathbf{X}_k^T \hat{\mathbf{R}}_{k-1}^{-1} \mathbf{X}_k) \end{aligned} \tag{5.64}$$

It will now be useful to define the auxiliary vector

$$\hat{\mathbf{S}}_k = \hat{\mathbf{R}}_{k-1}^{-1} \mathbf{X}_k \tag{5.65}$$

Then Eq. (5.64) becomes

$$\hat{\mathbf{S}}_k = \hat{\mathbf{R}}_k^{-1} \mathbf{X}_k (1 - \alpha + \alpha \mathbf{X}_k^T \hat{\mathbf{S}}_k) \tag{5.66}$$

We note that $\hat{\mathbf{R}}_k^{-1}$ is symmetric just as $\hat{\mathbf{R}}_k$ in Eq. (5.60) is symmetric and therefore the transpose of Eq. (5.65) is

$$\hat{\mathbf{S}}_k^T = \mathbf{X}_k^T \hat{\mathbf{R}}_{k-1}^{-1} \tag{5.67}$$

Dividing Eq. (5.66) by the scalar in parentheses and postmultiplying by Eq. (5.67), we obtain

$$\hat{\mathbf{R}}_k^{-1}\mathbf{X}_k\mathbf{X}_k^T\hat{\mathbf{R}}_{k-1}^{-1} = \frac{\hat{\mathbf{S}}_k\hat{\mathbf{S}}_k^T}{1-\alpha+\alpha\mathbf{X}_k^T\hat{\mathbf{S}}_k} \tag{5.68}$$

Finally, we substitute Eq. (5.68) into the last term in Eq. (5.63) and rearrange the terms in the result to obtain

$$\hat{\mathbf{R}}_k^{-1} = \frac{1}{1-\alpha}\left(\hat{\mathbf{R}}_{k-1}^{-1} - \alpha\frac{\hat{\mathbf{S}}_k\hat{\mathbf{S}}_k^T}{1-\alpha+\alpha\mathbf{X}_k^T\hat{\mathbf{S}}_k}\right) \tag{5.69}$$

In Eqs. (5.65) and (5.69) we now have an iterative algorithm for $\hat{\mathbf{R}}_k^{-1}$ equivalent to Eq. (5.60), which is the desired result.

As with σ_0^2 in Eq. (5.52), the initial matrix $\hat{\mathbf{R}}_0^{-1}$ in Eq. (5.69) would normally be our best *a priori* estimate of the inverse $\mathbf{R}$ matrix. If only the signal power is known initially, then $(1/\sigma^2)\mathbf{I}$ could be used for $\hat{\mathbf{R}}_0^{-1}$.

Having the algorithm for $\hat{\mathbf{R}}_k^{-1}$ and using Eq. (5.57) as before for the gradient estimate $\hat{\nabla}_k$, we obtain for the LMS-Newton weight update formula in Eq. (5.59)

$$\boxed{\mathbf{W}_{k+1} = \mathbf{W}_k + 2u\epsilon_k\hat{\mathbf{R}}_k^{-1}\mathbf{X}_k, \qquad 0 < u < 1} \tag{5.70}$$

Just as we needed Eq. (5.52) at each iteration if σ was changing in the LMS algorithm of Eq. (5.58), we now need Eqs. (5.65) and (5.69) at each iteration when $\mathbf{R}^{-1}$ is changing in the LMS-Newton algorithm of Eq. (5.70). In contrast with Eq. (5.58), we use $\hat{\mathbf{R}}_k^{-1}$ explicitly in Eq. (5.70) instead of $\mathbf{R}^{-1}$ because $\mathbf{R}$ is usually not known accurately in adaptive situations.

A weight track produced by the LMS-Newton algorithm in Eqs. (5.65), (5.69), and (5.70) is illustrated in Fig. 5.18 for comparison with the LMS weight track in Fig. 5.15. The starting weight vector is $\mathbf{W}_0 = (3, -4)^T$ as before and the signals are as in Fig. 5.8, so this weight track is comparable with all of the previous weight tracks. For Fig. 5.18, we set $\hat{\mathbf{R}}_0^{-1} = (1/\sigma^2)\mathbf{I}$ and let the inverse-$\mathbf{R}$ algorithm in Eqs. (5.65) and (5.69) run for 500 iterations with $\alpha = 0.01$ *prior* to running the full weight-adjusting algorithm in Eqs. (5.65), (5.69), and (5.70). We did this to illustrate a "tracking" situation where $\hat{\mathbf{R}}^{-1}$ remains near $\mathbf{R}^{-1}$ during adaptation and to show a weight track that is clearly more like the Newton weight track in Fig. 5.11 and less like the steepest-descent tracks in Figs. 5.14 and 5.15. Comparing Figs. 5.15 and 5.18, we note that the latter (LMS-Newton) weights converge in fewer iterations, just as Fig. 5.11 (Newton) showed faster convergence than Fig. 5.14.

5.7 PERFORMANCE CHARACTERISTICS

We have seen examples showing how the algorithms described in the two preceding sections perform in the simple interference-canceling system of Fig. 5.8. Now we will discuss some basic performance characteristics illustrated in these examples.

First we note that considerably more computing per iteration is required for the

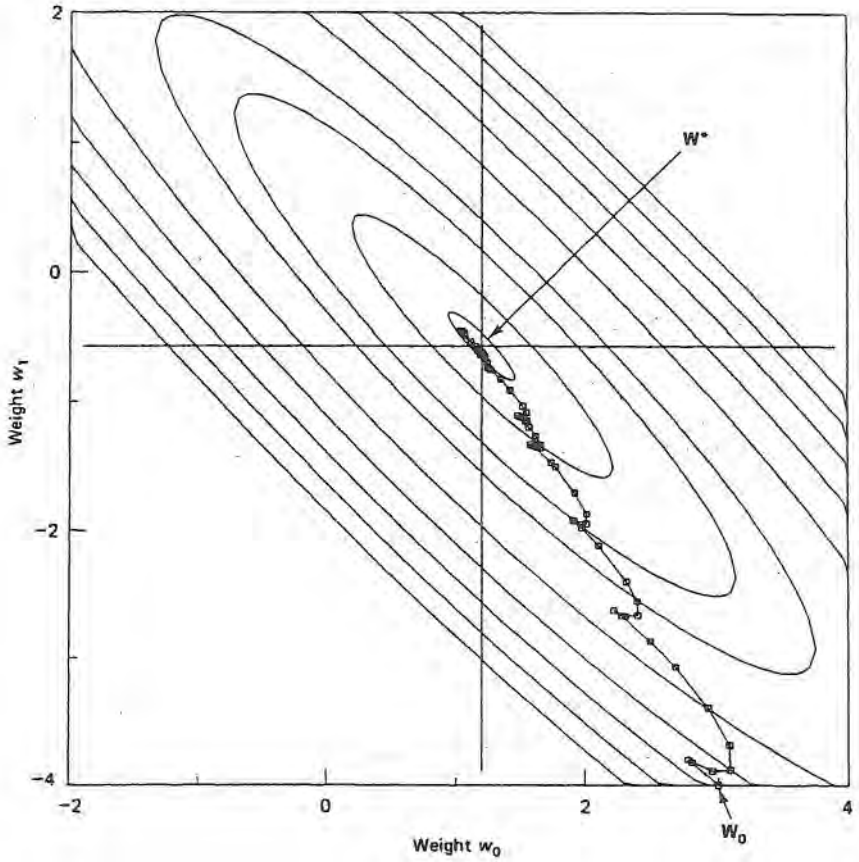

Figure 5.18 LMS-Newton weight track, showing convergence from $\mathbf{W}_0 = (3, -4)^T$ to near $\mathbf{W}^*$ for the interference-canceling system in Fig. 5.8. The adaptive gain parameter in Eq. (5.70) was $u = 0.03$, α in Eq. (5.69) was 0.01, and $\hat{\mathbf{R}}_0^{-1}$ in Eq. (5.69) was near $\mathbf{R}^{-1}$, as explained in the text. The weight track is shown for $k = 0, 1, \ldots, 60$.

Newton type of algorithm because the inverse-$\mathbf{R}$ estimate must be updated at each iteration, as in Eqs. (5.69) and (5.65). Also methods exist for constructing fast frequency-domain implementations of the algorithms. These implementations are called *frequency-domain adaptive filters* or *block adaptive filters* [41–46]. They are capable of producing a significant decrease in the computational burden of an adaptive processor and have been shown to have essentially the same convergence properties as the time-domain algorithm described above.

Concerning convergence, we have seen that under noise-free conditions the adaptive algorithms converge to the minimum MSE and then remain there, where the gradient of the performance surface is zero. We saw this in Eq. (5.37), for example, as long as the adaptive gain u was within its correct range from 0 to 1. We also saw in Eqs. (5.40) and (5.46) that the convergence time constants are inversely proportional to u. Convergence is faster when u is increased, at least up to $u = \frac{1}{2}$.

So why not increase u toward $\frac{1}{2}$ and have a more responsive, faster-converging adaptive system? The answer is that, in practical systems with noisy estimates of the gradient and/or the $\mathbf{R}$ matrix, increasing u means increasing the *excess MSE* after convergence. The excess MSE is the increase in the MSE over $(\text{MSE})_{\text{min}}$ after convergence, due to these noisy estimates. From Eq. (5.43), the excess MSE may be written as

$$\begin{aligned} \text{Excess MSE} &= (\text{MSE})_{\text{actual}} - (\text{MSE})_{\text{min}} \\ &= E[\mathbf{V}_k^T \mathbf{R} \mathbf{V}_k] \qquad \text{(after convergence)} \end{aligned} \tag{5.71}$$

For example, with the noisy gradient estimate given by $\hat{\nabla}_k$ in Eq. (5.57), the LMS algorithm adjusts the weight vector (and hence $\mathbf{V}_k$) at each iteration, causing the adaptive linear combiner to climb up the sides of its bowl-shaped performance surface in $L + 1$ dimensions. Of course, when the system discovers at the next iteration that it is at a point where the gradient is nonzero it will, on the average, adjust back toward the bottom of the bowl. With this process going on continually in a stochastic setting, the result is a nonzero, positive value for the excess MSE in Eq. (5.71).

For the LMS algorithm, assuming that $\mathbf{R}$ has equal eigenvalues and using Eq. (5.49) to relate μ and u, the excess MSE has been shown for small values of u to be approximately

$$\begin{aligned} \text{Excess MSE} &\approx \mu (\text{MSE})_{\text{min}} Tr[\mathbf{R}] \\ &\approx uL(\text{MSE})_{\text{min}}, \qquad 0 < u \ll 1 \end{aligned} \tag{5.72}$$

The derivation of this useful approximation is rather lengthy and therefore not included here. The interested reader may refer to [25] or to Chapters 5 and 6 in [5]. A related measure is the *misadjustment* M, which is a normalized version of the excess MSE:

$$\boxed{\begin{aligned} M &\triangleq \frac{\text{Excess MSE}}{(\text{MSE})_{\text{min}}} \\ &\approx uL, \qquad 0 < u \ll 1 \end{aligned}} \tag{5.73}$$

The approximation in Eq. (5.73) appears to be good for misadjustments up to about 0.1, even in cases where the eigenvalues of $\mathbf{R}$ are unequal, as in the example of Fig. 5.8 that we have been following in this chapter. For that example we set $\mathbf{W}_0 = \mathbf{W}^*$ and then ran the LMS algorithm of Eq. (5.58) for 15 values of u between 0.00 and 0.06, which, with $L = 2$, give theoretical misadjustments between 0.0 and 0.12 in Eq. (5.73). Each "run" actually consisted of 10 runs with $\mathbf{W}_0 = \mathbf{W}^*$ and $0 \le k \le 10^4$, using 10 different starting points on the signal waveforms. The result is plotted in Fig. 5.19, and we can see that Eq. (5.73) is a good approximation for values of M below $M = 0.1$. Above $M = 0.1$, the approximation gets worse with increasing M.

The time constants derived in Eqs. (5.40) and (5.46) were applicable to the noise-free convergence of the Newton algorithm of Eq. (5.34) and are therefore valid only for "ideal" operation of the algorithms described in this section, which use

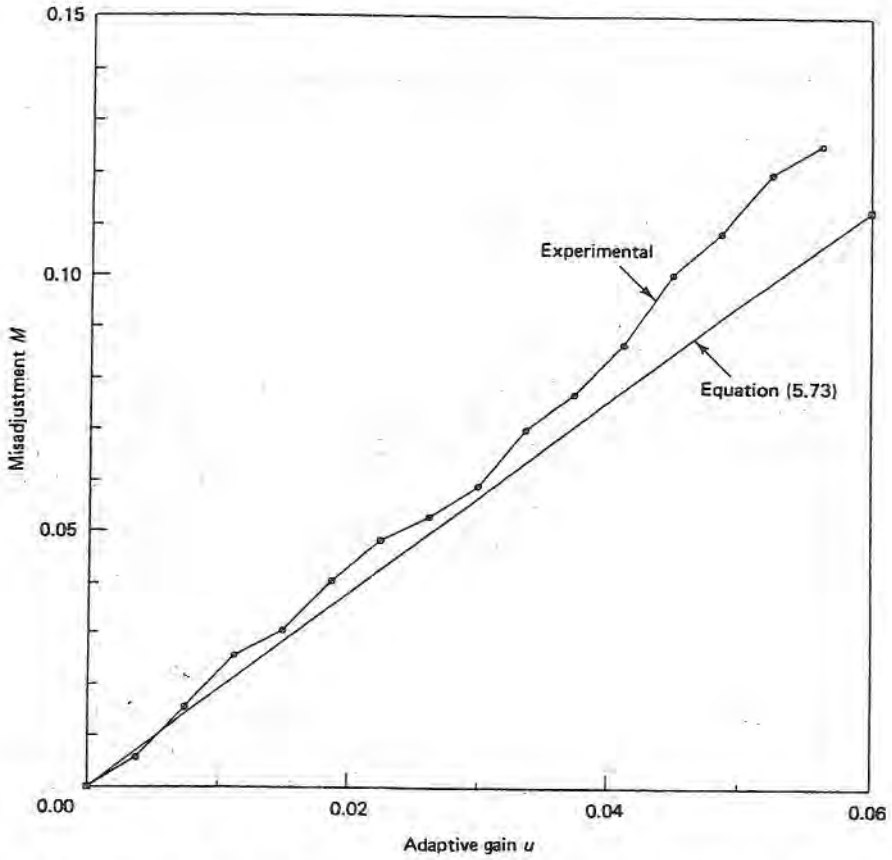

Figure 5.19 Misadjustment versus u after convergence of the adaptive filter in Fig. 5.8. Each experimental point is the average of 10^5 values of the squared error ϵ_k^2. The theoretical approximation in Eq. (5.73) is considered good up to around $M = 0.1$.

generally noisy estimates for $\mathbf{R}$ and $\overline{\nabla}_k$. If we assume that the noise-free formula of Eq. (5.46) is approximately correct for the learning time constant τ_{MSE}, we can combine Eqs. (5.46) and (5.73) to obtain

$$\boxed{M \approx \frac{L}{4\tau_{\text{MSE}}}, \qquad 0 < u \ll 1} \tag{5.74}$$

The time constant formula, and therefore Eq. (5.74), is valid for the LMS algorithm only with circular error contours (equal eigenvalues of $\mathbf{R}$) and noise-free convergence. However, Eq. (5.74) has been found to be a good approximation with the LMS algorithm "when the eigenvalues are sufficiently similar for the learning curve to be approximately fitted by a single exponential" [25].

The main point in this discussion of convergence is summarized in Eq. (5.74). In adaptive systems, low misadjustment and fast convergence are desirable but conflicting requirements. Faster convergence means noisier performance, with greater

excess MSE. In fact, noting that approximately four time constants, or $4\tau_{MSE}$, are required for convergence from an arbitrary MSE to $(MSE)_{min}$, we can restate Eq. (5.74) as follows [25]:

$$\boxed{\text{Misadjustment } (M) \approx \frac{\text{Number of weights } (L)}{\text{Convergence time } (4\tau_{MSE})}, \qquad 0 < u \ll 1} \tag{5.75}$$

This is perhaps the main performance characteristic of adaptive systems using the adaptive linear combiner structure.

Another factor affecting the stability and performance of the adaptive algorithms is the accuracy of the power and inverse-$\mathbf{R}$ estimates, $\hat{\sigma}_k^2$ and $\hat{\mathbf{R}}_k^{-1}$, in nonstationary situations. If $\hat{\mathbf{R}}_k^{-1}$ is accurate, the LMS-Newton algorithm will have superior convergence and tracking performance, as illustrated above. However, the smoothing filters in Figs. 5.13 and 5.17 for producing the estimates are simple lowpass filters, and they produce estimates of varying accuracy depending on the value of α and the frequency content of x_k. Suppose x_k is a sinusoid at frequency ω_0 radians, for example,

$$\begin{aligned} x_k &= \sin k\omega_0 \\ x_k^2 &= \sin^2 k\omega_0 = \tfrac{1}{2}(1 - \cos 2k\omega_0) \end{aligned} \tag{5.76}$$

After the initial transient, the filter in Fig. 5.13 will pass the dc component in Eq. (5.76), which is the correct power estimate, and will reject the component at $2\omega_0$, provided that ω_0 is large enough compared with α. The power gain of the filter is

$$\begin{aligned} |H(e^{j\omega})|^2 &= \left| \frac{\alpha}{1 - (1 - \alpha)e^{-j\omega}} \right|^2 \\ &= \frac{\alpha^2}{\alpha^2 + 4(1 - \alpha)\sin^2(\omega/2)} \end{aligned} \tag{5.77}$$

The gain is plotted in Fig. 5.20 for $\alpha = 0.1$, 0.01, and 0.001, and we can see how the rejection of the undesirable component improves as α decreases. On the other hand, the transient response of the filter decreases in duration as α increases, and α must be chosen with these factors in mind. The value 0.01 used in Fig. 5.18 is typical.

5.8 ADAPTIVE LATTICE STRUCTURE

In most of our discussion so far we have assumed that the single-input adaptive system is a transversal FIR filter. The lattice structure is an alternative to the transversal FIR filter in adaptive systems and is also potentially useful as a signal conditioner ahead of an adaptive linear combiner, as explained below. In this short section we will introduce the basic all-zero lattice structure that has been used adaptively and will briefly describe its important properties, referring to the literature for proofs and further description.

The lattice predictor structure proposed by Itakura and Saito [47] for speech analysis is shown in Fig. 5.21. We will use this structure as a basis for our discussion

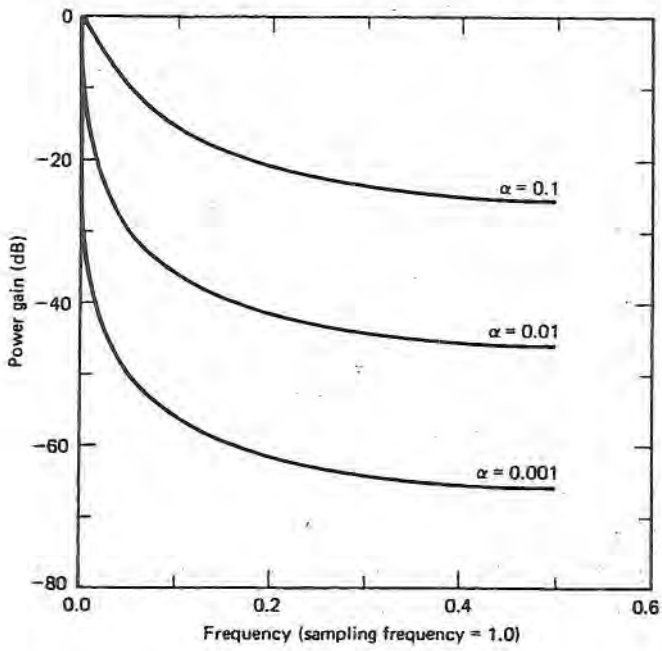

Figure 5.20 Power gain versus frequency for the smoothing filters in Figs. 5.13 and 5.17.

here. Before showing how the lattice can be made adaptive, we will discuss some of its important characteristics. Makhoul has provided an excellent basis for this discussion [48] and has also described the use of adaptive lattices in the analysis of speech [49].

A *stage* of the lattice in Fig. 5.21 is a single section with a delay, two weights, and two summing units. Each stage has two input signals and two output signals, and the inputs to the first stage are tied together as shown. In Fig. 5.21 we can see that the signals in the ith stage are related as follows:

$$\begin{bmatrix} f_{i+1}(k) \\ b_{i+1}(k) \end{bmatrix} = \begin{bmatrix} 1 & \kappa_i \\ \kappa_i & 1 \end{bmatrix} \begin{bmatrix} f_i(k) \\ b_i(k-1) \end{bmatrix}, \quad i = 0, 1, \ldots, L-1 \tag{5.78}$$

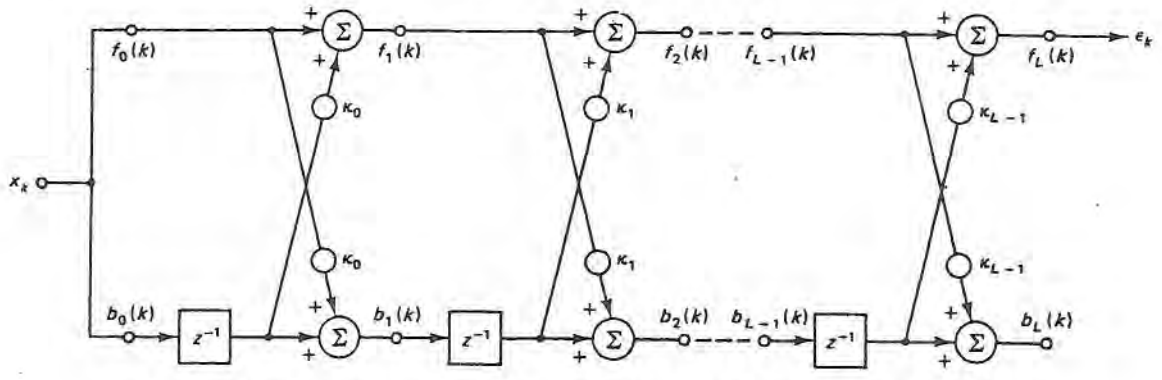

Figure 5.21 The all-zero two-multiplier lattice of Itakura and Saito. The lattice functions as a one-step predictor of x_k, and ϵ_k is the prediction error.

The weights $[\kappa_i]$ are known as reflection coefficients, or partial correlation ("parcor") coefficients. At stage 0 the inputs are $f_0(k) = b_0(k) = x_k$, and at stage $L - 1$ the outputs are $f_L(k) = \epsilon_k$ and $b_L(k)$. Taking the z-transform of Eq. (5.78) gives

$$\begin{bmatrix} F_{i+1}(z) \\ B_{i+1}(z) \end{bmatrix} = \begin{bmatrix} 1 & z^{-1}\kappa_i \\ \kappa_i & z^{-1} \end{bmatrix} \begin{bmatrix} F_i(z) \\ B_i(z) \end{bmatrix}, \qquad i = 0, 1, \ldots, L-1 \tag{5.79}$$

We can use Eq. (5.79) to find the overall transfer functions in Fig. 5.21. At the output of the ith stage we define

$$H_i(z) \triangleq \frac{F_i(z)}{F_0(z)} = \frac{F_i(z)}{X(z)}, \qquad G_i(z) \triangleq \frac{B_i(z)}{B_0(z)} = \frac{B_i(z)}{X(z)}, \qquad i = 0, 1, \ldots, L \tag{5.80}$$

Using these definitions in Eq. (5.79) and using Eq. (5.79) recursively, we can write

$$\begin{bmatrix} H_i(z) \\ G_i(z) \end{bmatrix} = \begin{bmatrix} 1 & z^{-1}\kappa_{i-1} \\ \kappa_{i-1} & z^{-1} \end{bmatrix} \begin{bmatrix} 1 & z^{-1}\kappa_{i-2} \\ \kappa_{i-2} & z^{-1} \end{bmatrix} \cdots \begin{bmatrix} 1 & z^{-1}\kappa_0 \\ \kappa_0 & z^{-1} \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{5.81}$$

If we set $i = L$, Eq. (5.81) gives the overall transfer functions. Thus we see that both overall transfer functions could be written as polynomials in z^{-1}. In fact, at the ith stage, we could write

$$H_i(z) = 1 + \sum_{m=1}^{i} a_{im} z^{-m}, \qquad i = 1, 2, \ldots, L \tag{5.82}$$

The leading coefficient is seen in Eq. (5.81) to be 1, and the rest of the a's in Eq. (5.82) could be found by multiplying the terms in Eq. (5.81). In particular, if we imagine multiplying the i square matrices together in Eq. (5.81), the coefficient of z^{-i}, which is equal to a_{ii} in Eq. (5.82), is seen to be κ_{i-1}. Thus,

$$a_{ii} = \kappa_{i-1} \tag{5.83}$$

To see how $G_i(z)$ is related to $H_i(z)$, we next revise Eq. (5.81) by replacing z with z^{-1} in the top equation to obtain

$$\begin{bmatrix} H_i(z^{-1}) \\ G_i(z) \end{bmatrix} = \begin{bmatrix} 1 & z\kappa_{i-1} \\ \kappa_{i-1} & z^{-1} \end{bmatrix} \begin{bmatrix} 1 & z\kappa_{i-2} \\ \kappa_{i-2} & z^{-1} \end{bmatrix} \cdots \begin{bmatrix} 1 & z\kappa_0 \\ \kappa_0 & z^{-1} \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{5.84}$$

If we multiply the top equation in Eq. (5.84) by z^{-i}, we can write

$$\begin{bmatrix} z^{-i}H_i(z^{-1}) \\ G_i(z) \end{bmatrix} = \begin{bmatrix} z^{-1} & \kappa_{i-1} \\ \kappa_{i-1} & z^{-1} \end{bmatrix} \begin{bmatrix} z^{-1} & \kappa_{i-2} \\ \kappa_{i-2} & z^{-1} \end{bmatrix} \cdots \begin{bmatrix} z^{-1} & \kappa_0 \\ \kappa_0 & z^{-1} \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{5.85}$$

If we think of carrying out the product in Eq. (5.85) from right to left, we can see that the two rows are identical, and thus the relation of $G_i(z)$ to $H_i(z)$ is

$$\boxed{G_i(z) = z^{-i}H_i(z^{-1}), \qquad i = 0, 1, \ldots, L} \tag{5.86}$$

(For $i = 0$, this result is understood to be $G_0(z) = H_0(z) = 1$.) We can now relate the lattice in Fig. 5.21 to a specific transversal filter. We first write Eq. (5.81) in a recursive form similar to Eq. (5.79):

$$\begin{bmatrix} H_i(z) \\ G_i(z) \end{bmatrix} = \begin{bmatrix} 1 & z^{-1}\kappa_{i-1} \\ \kappa_{i-1} & z^{-1} \end{bmatrix} \begin{bmatrix} H_{i-1}(z) \\ G_{i-1}(z) \end{bmatrix}, \qquad i = 0, 1, \ldots, L-1 \tag{5.87}$$

We next multiply the lower equation in Eq. (5.87) by κ_{i-1}, subtract the two equations, and solve for $H_{i-1}(z)$ to obtain

$$H_{i-1}(z) = \frac{H_i(z) - \kappa_{i-1} G_i(z)}{1 - \kappa_{i-1}^2}, \qquad i = L, L-1, \ldots, 1 \tag{5.88}$$

With this result we can write an algorithm for changing a transversal filter into a lattice. We start with a filter in the form of Eq. (5.82) with $i = L$:

$$H_L(z) = 1 + \sum_{m=1}^{L} a_{Lm} z^{-m} \tag{5.89}$$

$$= 1 - \sum_{m=1}^{L} w_m z^{-m} \tag{5.90}$$

This we recognize as the adaptive predictor in Fig. 5.2(b) with $\Delta = 1$ and with the processor (FIR filter) weights given by $[w_m] = [-a_{Lm}]$. Now we combine Eqs. (5.82), (5.83), (5.86), and (5.88) to form the algorithm that uses Eq. (5.89) as its starting point and generates each lattice weight κ_i for $i = L-1$ down through $i = 0$:

$$\begin{aligned} &\text{for } i = L, L-1, \ldots, 1: \\ &\quad \kappa_{i-1} = a_{ii} \\ &\quad G_i(z) = z^{-i} H_i(z^{-1}) \\ &\quad H_{i-1}(z) = \frac{H_i(z) - \kappa_{i-1} G_i(z)}{1 - \kappa_{i-1}^2} \end{aligned} \tag{5.91}$$

Thus we have a method for transforming a given transversal predictor, or FIR filter in the form of Eq. (5.89), into a lattice in the form of Fig. 5.21. In other words, we have shown that the lattice in Fig. 5.21 is a one-step ($\Delta = 1$) predictor and we have justified the labeling of the *forward prediction error* $f_L(k) = \epsilon_k$ as the one-step prediction error signal in Fig. 5.21.

We also note from Eqs. (5.86) and (5.89) that the transfer function from x_k to the lower output $b_L(k)$ in Fig. 5.21 is

$$\begin{aligned} G_L(z) &= z^{-L}\left[1 - \sum_{m=1}^{L} w_m z^{+m}\right] \\ &= z^{-L} - \sum_{m=0}^{L-1} w_{L-m} z^{-m} \end{aligned} \tag{5.92}$$

From this result, $b_L(k)$ is called the *backward prediction error* because the samples x_k through x_{k-L+1} are being used to predict the sample x_{k-L}. The forward and backward prediction errors are illustrated in Fig. 5.22, where $H_L(z)$ and $G_L(z)$ are shown in transversal form.

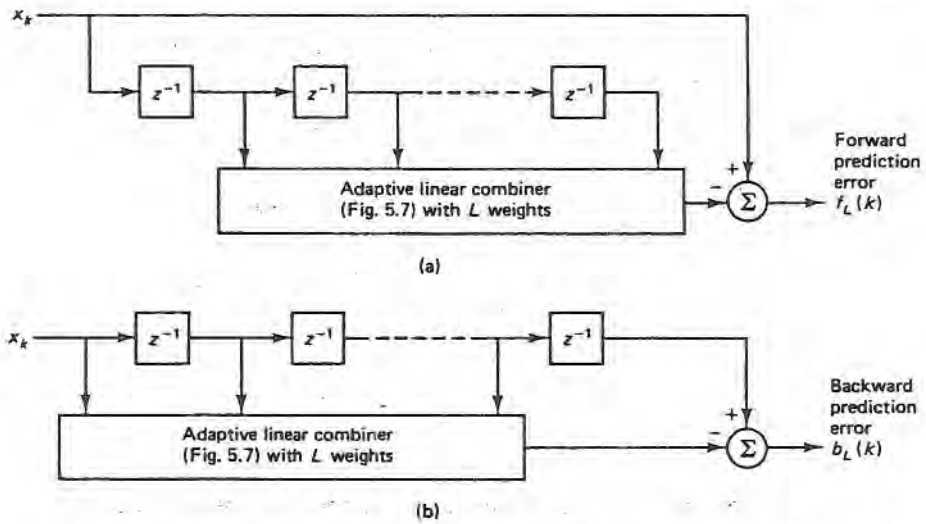

Figure 5.22 The two lattice transfer functions: (a) $H_L(z)$ in Eq. (5.89) and (b) $G_L(z)$ in Eq. (5.92), illustrated in the form of transversal filters to show the forward and backward prediction errors.

Griffiths [50] has shown how to make the lattice in Fig. 5.21 adaptive using the LMS algorithm. Griffiths allows the weights to adapt separately instead of in pairs as in Fig. 5.21, but if we keep the weights equal in pairs the results are very similar. We have seen that the output $f_{i+1}(k)$ of the ith lattice stage is the one-step prediction error at that stage. In adaptation we therefore adjust the weight value κ_i to minimize the expected value of $f_{i+1}^2(k)$. The steepest-descent formula, similar to Eq. (5.47), is

$$\kappa_i(k+1) = \kappa_i(k) - \mu_i \hat{\nabla}_k \tag{5.93}$$

where k is the time index as usual and, similar to Eq. (5.57),

$$\begin{aligned} \hat{\nabla}_k &= \text{Estimate of } \frac{\partial E[f_{i+1}^2(k)]}{\partial \kappa_i} \\ &= \frac{\partial f_{i+1}^2(k)}{\partial \kappa_i} = 2f_{i+1}(k)\frac{\partial f_{i+1}(k)}{\partial \kappa_i} \end{aligned} \tag{5.94}$$

The adaptive gain for the ith stage (μ_i) is similar to the gain μ in Eq. (5.47), and the range in Eq. (5.49) applies here, with σ_i^2 the input power to the ith stage:

$$\boxed{0 < \mu_i < \frac{1}{\sigma_i^2}} \tag{5.95}$$

During adaptive processing, the signal power changes throughout the lattice, and so the power (and hence μ_i) at each stage may require continual adjustment using a method such as the one in Eq. (5.52) as a part of the adaptive process.† Using the top

†The subscript i in Eq. (5.95) denotes the lattice stage, whereas the subscript k in Eq. (5.52) is the time index.

equation in Eq. (5.78) in Eq. (5.94) and then using Eq. (5.94) in Eq. (5.93), we have the LMS lattice algorithm:

$$\boxed{\kappa_i(k+1) = \kappa_i(k) - 2\mu_i f_{i+1}(k) b_i(k-1), \qquad i = 0, 1, \ldots, L-1} \tag{5.96}$$

As with the LMS algorithm discussed previously, acceptable misadjustment levels at each lattice stage are generally attained by keeping μ_i to a small fraction of its upper limit in Eq. (5.95).

An important property of the lattice shown by Jury [51] is the following:

$$\boxed{\begin{array}{c}\text{If } |\kappa_i| < 1 \text{ for } i = 0, 1, \ldots, L-1, \\ \text{then } H_L(z) \text{ is minimum phase}\end{array}} \tag{5.97}$$

That is, the zeros of $H_L(z)$ are all inside the unit circle if the lattice weights are all less than 1 in magnitude. Therefore, the all-pole inverse, $1/H_L(z)$, is stable under these conditions. This property is very useful when applying adaptive lattices in speech communications. The latter subject has been discussed by Itakura and Saito [47] and others. In particular, Makhoul and Cosell [49] present a general class of adaptive algorithms for lattice predictors in speech analysis that goes beyond the simple LMS algorithm in Eq. (5.96). Lee, Morf, and Friedlander [52] have also presented improved adaptive algorithms for lattices.

When the adaptive lattice has converged, the optimum weight values are denoted $[\kappa_i^*]$. With stationary signals, these values turn out to be the same whether we minimize the forward or the backward prediction error. Taking the expected squared value of Eq. (5.78) and noting that $E[b_i^2(k)] = E[b_i^2(k-1)]$ under stationary conditions, we have

$$E[f_{i+1}^2(k)] = E[f_i^2(k)] + \kappa_i^2 E[b_i^2(k)] + 2\kappa_i E[f_i(k) b_i(k-1)] \tag{5.98}$$

$$E[b_{i+1}^2(k)] = \kappa_i^2 E[f_i^2(k)] + E[b_i^2(k)] + 2\kappa_i E[f_i(k) b_i(k-1)] \tag{5.99}$$

At stage $i = 0$, $f_0(k) = b_0(k) = x_k$, and so Eqs. (5.98) and (5.99) are equal, and $E[f_1^2(k)] = E[b_1^2(k)]$. But then at stage 1, Eqs. (5.98) and (5.99) are equal again, and so on, and therefore we have

$$E[f_i^2(k)] = E[b_i^2(k)], \qquad i = 0, 1, \ldots, L-1 \tag{5.100}$$

Setting the derivative of Eq. (5.98) or (5.99) with respect to κ_i equal to zero, we have the optimum weight that minimizes both prediction MSEs simultaneously:

$$\boxed{\begin{aligned}\kappa_i^* &= -\frac{E[f_i(k) b_i(k-1)]}{E[f_i^2(k)]} \\ &= -\frac{E[f_i(k) b_i(k-1)]}{E[b_i^2(k)]}, \qquad i = 0, 1, \ldots, L-1\end{aligned}} \tag{5.101}$$

Makhoul [53] has derived the formulas in Eq. (5.101) as well as some similar formulas optimized on different criteria.

When the weights are optimized, an adaptive processor, either transversal or lattice in structure, has the property that the error and input signals are orthogonal. This property can be derived for the transversal structure from Fig. 5.7, where we have

$$\epsilon_k = d_k - \mathbf{X}_k^T \mathbf{W} \tag{5.102}$$

We multiply each scalar term in Eq. (5.102) by $\mathbf{X}_k$, take the expected value, and use Eqs. (5.17) and (5.19) to obtain

$$E[\epsilon_k \mathbf{X}_k] = \mathbf{P} - \mathbf{R}\mathbf{W} \tag{5.103}$$

But at convergence, Eq. (5.24) gives $\mathbf{W}^* = \mathbf{R}^{-1}\mathbf{P}$, and so Eq. (5.103) becomes

$$E[\epsilon_k \mathbf{X}_k]_{\mathbf{W}=\mathbf{W}^*} = \mathbf{0} \tag{5.104}$$

Thus the error and input signals are orthogonal.

Similarly, in the lattice of Fig. 5.21 with optimal weights, Makhoul [48] has shown that the backward prediction errors (b's) are mutually orthogonal and that the set of optimal weights in Eq. (5.101), each computed using local statistics at each stage, minimizes the MSE at each stage as well as the overall MSE, i.e., the MSE at the final stage.

This mutual orthogonality of the signals in a lattice opens the possibility suggested by Griffiths [54] of using the lattice as a signal conditioner ahead of an adaptive processor, as illustrated in Fig. 5.23. Instead of feeding the signal vector $\mathbf{X}_k$ directly to the adaptive processor, we first process $\mathbf{X}_k$ with a lattice to produce a dependent signal vector $\mathbf{B}_k$, whose elements (after convergence) are mutually orthogonal. Looking again at Eq. (5.17), but now with $\mathbf{B}_k$ in place of $\mathbf{X}_k$, we see that the correlation matrix $\mathbf{R}$ is *diagonal* after the lattice has converged. That is,

$$\mathbf{R}_{(\text{Fig. 5.23})} = \text{Diag.}[r_{b_0 b_0} \quad r_{b_1 b_1} \quad \cdots \quad r_{b_{L-1} b_{L-1}}] \tag{5.105}$$

We also have, from Eq. (5.19) or (5.20),

$$\mathbf{P}_{(\text{Fig. 5.23})} = [r_{db_0} \quad r_{db_1} \quad \cdots \quad r_{db_{L-1}}]^T \tag{5.106}$$

Therefore, from Eq. (5.24), the optimum weight vector in Fig. 5.23 is

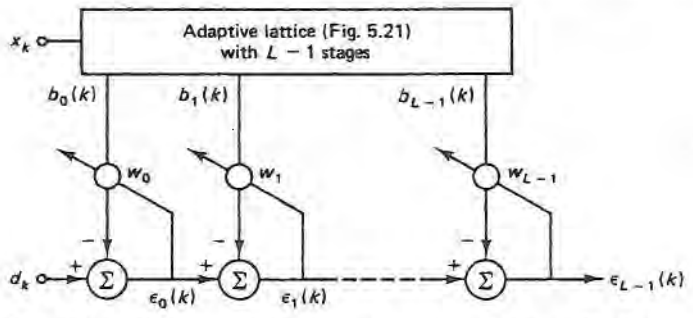

Figure 5.23 Using an adaptive lattice is a signal conditioner to provide orthogonal inputs to an adaptive linear combiner, allowing local adjustment of the combiner weights.

$$\mathbf{W}^*_{(\text{Fig. 5.23})} = \mathbf{R}^{-1}_{(\text{Fig. 5.23})}\mathbf{P}_{(\text{Fig. 5.23})} = \begin{bmatrix} \dfrac{r_{db_0}}{r_{b_0b_0}} & \dfrac{r_{db_1}}{r_{b_1b_1}} & \cdots & \dfrac{r_{db_{L-1}}}{r_{b_{L-1}b_{L-1}}} \end{bmatrix}^T \tag{5.107}$$

This result justifies the local weight-adjustment scheme shown in Fig. 5.23. At each stage in the adaptive linear combiner, the optimal weight is seen in Eq. (5.107) to depend only on local signals. The weight w_i^* is determined only by b_i and ϵ_{i-1}, not by the other b's. The LMS algorithm or any other desired scheme can be used to estimate continually the ratios in Eq. (5.107).

The advantage of the local weight-adjustment scheme in Fig. 5.23 is that a local adaptive gain can now be used at each stage, just as in the lattice. If $w_i(k)$ is the kth value of the ith weight in Fig. 5.21, the LMS algorithm, similar to Eq. (5.58), is

$$w_i(k+1) = w_i(k) + \frac{2u}{\sigma_i^2}\epsilon_i(k)f_i(k), \qquad \begin{matrix} i = 0, 1, \ldots, L-1, \\ 0 < u < 1 \end{matrix} \tag{5.108}$$

At each stage, σ_i^2 is now $E[b_i^2]$, which must be either known or estimated as discussed previously, as in Eq. (5.52). But the steepest-descent convergence problems seen previously, in Fig. 5.14 for example, caused by differing time constants due in turn to disparate eigenvalues of the input correlation matrix, do not occur here. Thus the convergence of Eq. (5.108) is potentially faster than that of Eq. (5.58) for such cases.

A further analysis of the "orthogonalized LMS" algorithm is provided by Widrow and Walach [55]. Other methods besides the adaptive lattice, such as the discrete Fourier transform, can be used to orthogonalize the input signal and produce essentially the same effect on the performance of the adaptive linear combiner. We will conclude by showing how the application of Fig. 5.23 alters the performance surface of the interference-canceling system in Fig. 5.8.

5.9 AN EXAMPLE OF AN ADAPTIVE LATTICE

In this example we use the adaptive lattice with the interference-canceling system of Fig. 5.8 to obtain a comparison with the performances shown in Figs. 5.11, 5.14, 5.15, and 5.18. The lattice is used as in Fig. 5.23; that is, it is used to orthogonalize the signal x_k to the adaptive FIR filter in Fig. 5.8. Since the adaptive filter has only two weights, a single lattice stage suffices to provide the two orthogonal data sequences.

The system resulting from the application of Fig. 5.23 to Fig. 5.8 is shown in Fig. 5.24. The adaptive lattice has the effect of altering the R matrix and the P vector and thus the performance surface of the adaptive filter. Let us assume that the lattice is converged as in Eq. (5.101), with

$$\kappa_0 = \kappa_0^* = -\frac{E[f_0(k)b_0(k-1)]}{E[f_0^2(k)]} = -\frac{E[x_kx_{k-1}]}{E[x_k^2]} = -\cos\frac{\pi}{8} \tag{5.109}$$

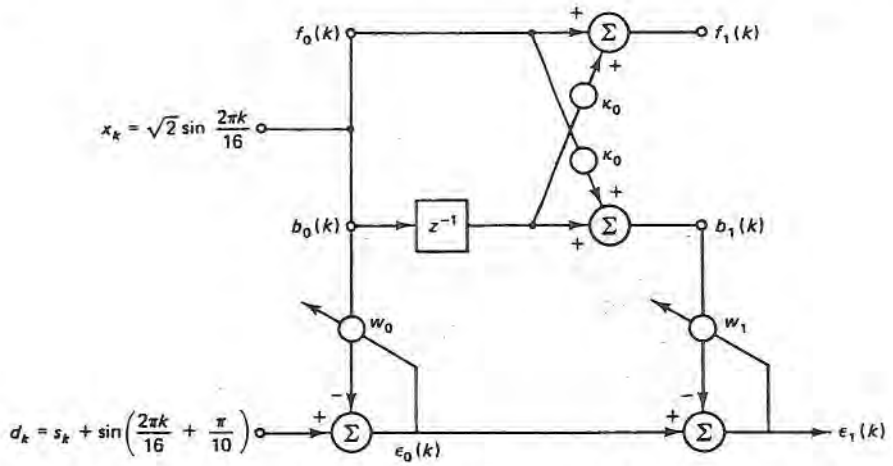

Figure 5.24 An example in which Fig. 5.23 is applied to the interference-canceling system in Fig. 5.8.

Then, with x_k as given in Fig. 5.24, the lattice signal $b_1(k)$ is

$$\begin{aligned} b_1(k) &= \kappa_0 f_0(k) + b_0(k-1) \\ &= \kappa_0 x_k + x_{k-1} \\ &= -\sqrt{2}\sin\frac{\pi}{8}\cos\frac{2\pi k}{16} \end{aligned} \tag{5.110}$$

We note that $b_1(k)$, a cosine function of k, is orthogonal to $b_0(k) = x_k$, a sine function of k. We also have the following correlation functions:

$$r_{b_0 b_0} = E[x_k^2] = 1 \tag{5.111}$$

$$r_{b_1 b_1} = E[b_1^2(k)] = \sin^2\frac{\pi}{8} \tag{5.112}$$

$$r_{db_0} = \sqrt{2}\,E\left[\sin\left(\frac{2\pi k}{16} + \frac{\pi}{10}\right)\sin\frac{2\pi k}{16}\right] = \frac{1}{\sqrt{2}}\cos\frac{\pi}{10} \tag{5.113}$$

$$\begin{aligned} r_{db_1} &= -\sqrt{2}\sin\frac{\pi}{8}\,E\left[\sin\left(\frac{2\pi k}{16} + \frac{\pi}{10}\right)\cos\frac{2\pi k}{16}\right] \\ &= -\frac{1}{\sqrt{2}}\sin\frac{\pi}{8}\sin\frac{\pi}{10} \end{aligned} \tag{5.114}$$

Using these results in Eqs. (5.105) and (5.106), we have

$$\mathbf{R} = \begin{bmatrix} 1 & 0 \\ 0 & \sin^2\frac{\pi}{8} \end{bmatrix} \tag{5.115}$$

$$\mathbf{P} = \frac{1}{\sqrt{2}} \begin{bmatrix} \cos \dfrac{\pi}{10} \\ -\sin \dfrac{\pi}{8} \sin \dfrac{\pi}{10} \end{bmatrix} \tag{5.116}$$

Using these results in Eq. (5.14) or (5.22), we have now an error surface for the adaptive linear combiner that differs from Eq. (5.29):

$$\text{MSE} = 0.55 + w_0^2 + w_1^2 \sin^2 \frac{\pi}{8} - \sqrt{2}\left(w_0 \cos \frac{\pi}{10} - w_1 \sin \frac{\pi}{8} \sin \frac{\pi}{10}\right) \tag{5.117}$$

Contours of this MSE, which may be compared with those in Fig. 5.10, are shown in Fig. 5.25. Figure 5.25 also shows a weight track, illustrating convergence of the adaptive filter with signals the same as in Figs. 5.11, 5.14, 5.15, and 5.18.

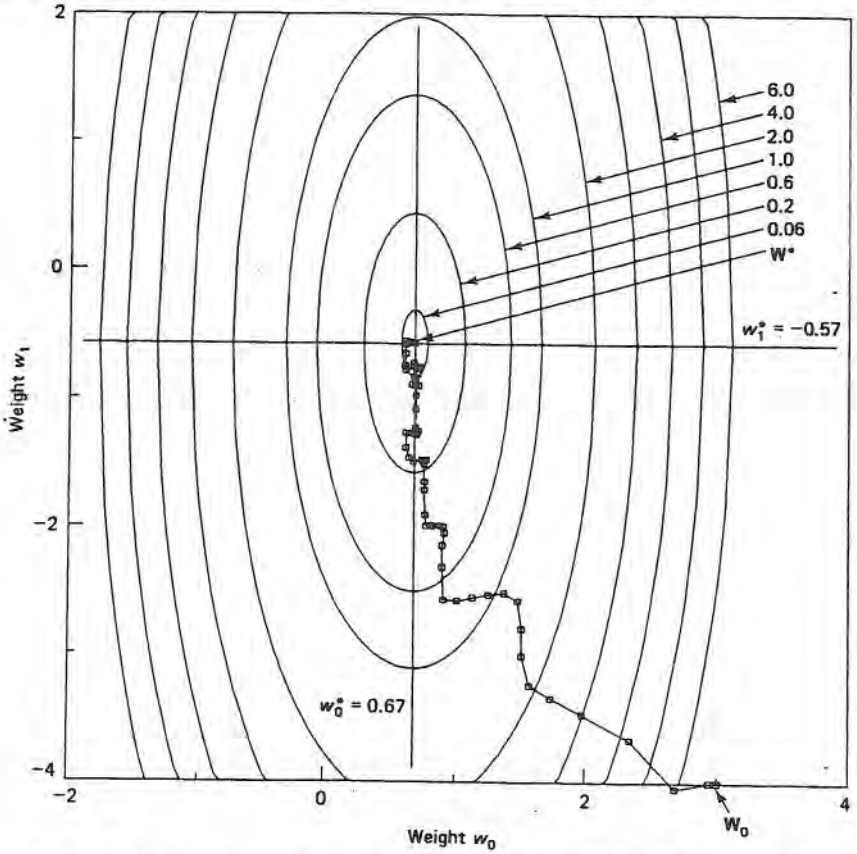

Figure 5.25 Error contours and LMS weight track for Fig. 5.24, showing convergence from $\mathbf{W}_0 = (3, -4)^T$ to near $\mathbf{W}^*$. The adaptive gain parameter was $\mu = 0.03$ and $k = 0, 1, \ldots, 60$ as in Fig. 5.18. MSE values from Eq. (5.117) are shown as error contours, for comparison with Fig. 5.10.

The weight track in Fig. 5.25 is not computed with a converged lattice coefficient, κ_0^*. The lattice is also allowed to converge from $\kappa_0 = 0$ at $k = 0$ to produce a more realistic example. If κ_0^* had been used instead, convergence to W* would have been faster than in Fig. 5.25.

Convergence plots are shown in Fig. 5.26 for κ_0, w_0, and w_1. The adaptive gain was relatively high ($\mu = 0.1$) for κ_0, thus allowing the lattice to converge rapidly and produce near-orthogonal inputs to the adaptive filter. Note that while κ_0 is converging, and also after κ_0 is converged but misadjusted, the weights w_0 and w_1 are converging to their optimal values. Thus the dynamic behavior of the adaptive lattice combined with the FIR adaptive filter, although rather complex to analyze, appears to be satisfactory, and in this particular example better than that of the adaptive filter without the lattice.

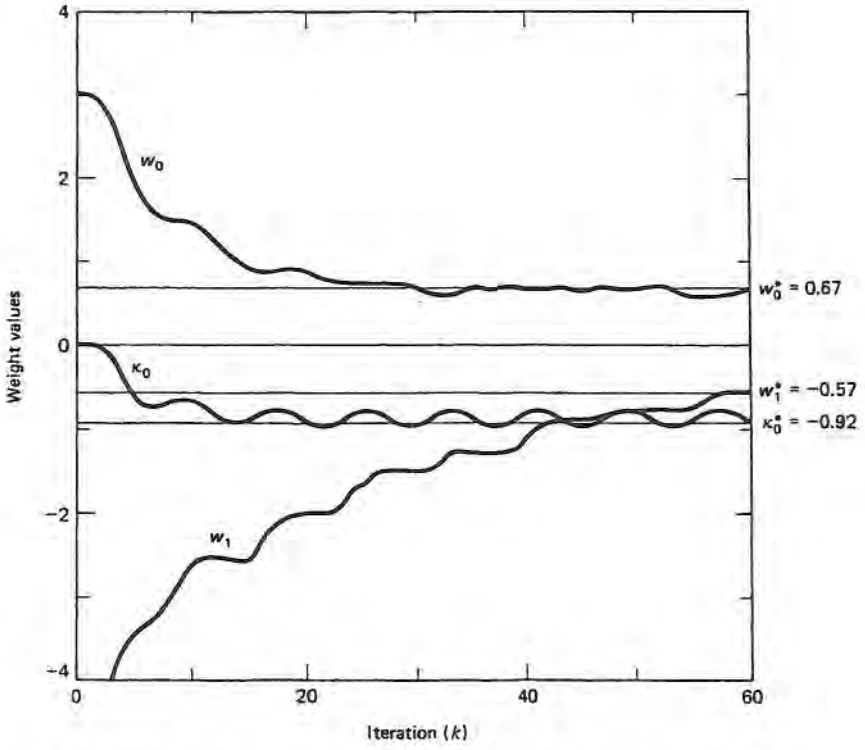

Figure 5.26 Convergence of the lattice weight κ_0 and the adaptive filter weights w_0 and w_1 in Fig. 5.24. Initial values at $k = 0$ are $\kappa_0 = 0$, $w_0 = 3$, and $w_1 = -4$. Convergence parameters (μ) are 0.1 for the lattice and 0.03 for the adaptive filter.

21. R. W. Lucky, "Techniques for Adaptive Equalization of Digital Communication Systems," *Bell System Tech. J.*, pp. 255–286, Feb. 1966.
22. L. J. Griffiths, F. R. Smolka, and L. D. Trembly, "Adaptive Deconvolution: A New Technique for Processing Time-Varying Seismic Data," *Geophysics*, June 1977.
23. R. A. Monzingo and T. W. Miller, *Introduction to Adaptive Arrays*, Wiley, New York, 1980.
24. C. F. N. Cowan and P. M. Grant, Eds., *Adaptive Filters*, Prentice-Hall, Englewood Cliffs, NJ, 1985.
25. B. Widrow, J. M. McCool, M. G. Larimore, and C. R. Johnson, Jr., "Stationary and Nonstationary Learning Characteristics of the LMS Adaptive Filter," *Proc. IEEE*, Vol. 64, pp. 1151–1162, Aug. 1976.
26. D. M. Etter and S. D. Stearns, "Adaptive Estimation of Time Delays in Sampled Data Systems," *IEEE Trans. Acoustics, Speech, and Signal Processing*, Vol. 29, pp. 582–587, June 1981.
27. S. A. White, "A Nonlinear Digital Adaptive Filter," *Conference Record, 14th Asilomar Conference on Circuits, Systems and Computers*, pp. 350–354, Nov. 1980.
28. A. V. Oppenheim and R. W. Schafer, *Digital Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1975, Chapter 8.
29. N. Wiener, *Extrapolation, Interpolation and Smoothing of Stationary Time Series with Engineering Applications*, Wiley, New York, 1949.
30. R. A. David, *IIR Adaptive Algorithms Based on Gradient Search Techniques*, Ph.D. Thesis, Stanford University, Stanford, CA, Aug. 1981.
31. P. L. Feintuch, "An Adaptive Recursive LMS Filter," *Proc. IEEE*, p. 1622, Nov. 1976.
32. M. G. Larimore, J. R. Treichler, and C. R. Johnson, Jr., "SHARF: An Algorithm for Adapting IIR Digital Filters," *IEEE Trans. Acoustics, Speech, and Signal Processing*, Vol. 28, pp. 428–440, Aug. 1980.
33. S. A. White, "An Adaptive Recursive Digital Filter," *Conference Record, 9th Asilomar Conference on Circuits, Systems, and Computers*, pp. 21–25, Nov. 1975.
34. N. Levinson, "The Wiener RMS Error Criterion in Filter Design and Prediction," *J. Math. and Physics*, Vol. 25, pp. 261–278, 1946.
35. S. Zohar, "Toeplitz Matrix Inversion: The Algorithm of W. F. Trench," *J. Association for Computing Machinery*, Vol. 16, pp. 592–601, Oct. 1969.
36. B. Widrow and J. M. McCool, "A Comparison of Adaptive Algorithms Based on the Methods of Steepest Descent and Random Search," *IEEE Trans. Antennas and Propagation*, Vol. 24, p. 615, Sept. 1976.
37. D. M. Etter and M. M. Masakawa, "A Comparison of Algorithms for Adaptive Estimation of the Time Delay Between Sampled Signals," *Proc. ICASSP-81*, p. 1253, March 1981.
38. G. B. Thomas, Jr., *Calculus and Analytic Geometry*, 4th ed., Addison-Wesley, Reading, MA, 1968, Section 10.3.
39. D. G. Luenberger, *Introduction to Linear and Nonlinear Programming*, Addison-Wesley, Reading, MA, 1973, Section 7.7.
40. N. Ahmed, D. R. Hummels, M. L. Uhl, and D. L. Soldan, "A Short-Term Sequential Regression Algorithm," *IEEE Trans. Acoustics, Speech, and Signal Processing*, Vol. 27, pp. 453–457, Oct. 1979.
41. N. J. Bershad and P. L. Feintuch, "Analysis of the Frequency Domain Adaptive Filter," *Proc. IEEE*, Vol. 67, pp. 1658–1659, Dec. 1979.

REFERENCES

1. B. Widrow and M. E. Hoff, Jr., "Adaptive Switching Circuits," *1960 IRE WESCON Convention Record*, Part 4, pp. 96–104, 1960.
2. B. Widrow et al., "Adaptive Noise Cancelling: Principles and Applications," *Proc. IEEE*, Vol. 63, pp. 1692–1716, Dec. 1975.
3. B. Widrow et al., "Adaptive Antenna Systems," *Proc. IEEE*, Vol. 55, pp. 2143–2159, Dec. 1967.
4. B. Widrow, "Adaptive Filters," in *Aspects of Network and System Theory*, R. E. Kalman and N. DeClaris, Eds., Holt, Rinehart, and Winston, New York, 1970, p. 563.
5. B. Widrow and S. D. Stearns, *Adaptive Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1985.
6. B. S. Atal and S. L. Hanauer, "Speech Analysis and Synthesis by Linear Prediction of the Speech Wave," *J. Acoustical Soc. Amer.*, Vol. 50, pp. 637–755, 1971.
7. J. Makhoul, "Spectral Analysis of Speech by Linear Prediction," *IEEE Trans. Audio and Electroacoustics*, Vol. 21, pp. 140–148, June 1973.
8. L. J. Griffiths, "Rapid Measurement of Digital Instantaneous Frequency," *IEEE Trans. Acoustics, Speech, and Signal Processing*, Vol. 23, pp. 207–222, April 1975.
9. G. A. Clark and P. W. Rodgers, "Adaptive Prediction Applied to Seismic Event Detection," *Proc. IEEE*, Vol. 69, pp. 1166–1168, Sept. 1981.
10. W. P. Dove and A. V. Oppenheim, "Event Location Using Recursive Least-Squares Signal Processing," *Proc. ICASSP-80*, pp. 848–850, April 1980.
11. S. D. Stearns and L. J. Vortman, "Seismic Event Detection Using Adaptive Predictors," *Proc. ICASSP-81*, pp. 1058–1061, March 1981.
12. J. R. Zeidler, E. H. Satorius, D. M. Chabries, and H. T. Wexler, "Adaptive Enhancement of Multiple Sinusoids in Uncorrelated Noise," *IEEE Trans. Acoustics, Speech, and Signal Processing*, Vol. 26, pp. 240–254, June 1978.
13. G. F. Franklin and J. D. Powell, *Digital Control of Dynamic Systems*, Addison-Wesley, Reading, MA, 1980.
14. A. I. Landau, *Adaptive Control: The Model Reference Approach*, Dekker, 1979.
15. B. Widrow, P. F. Titchener, and R. P. Gooch, "Adaptive Design of Digital Filters," *Proc. ICASSP-81*, pp. 243–246, March 1981.
16. D. H. Youn, N. Ahmed, and G. C. Carter, "Magnitude-Squared Coherence Function Estimation: An Adaptive Approach," *IEEE Trans. Acoustics, Speech, and Signal Processing*, Vol. 31, pp. 137–142, Feb. 1983.
17. B. Widrow, J. M. McCool, and B. Medoff, "Adaptive Control by Inverse Modeling," *Conference Record, 12th Asilomar Conference on Circuits, Systems, and Computers*, pp. 90–94, Nov. 1978.
18. B. Widrow, D. Shur, and S. Shaffer, "On Adaptive Inverse Control," *Conference Record, 15th Asilomar Conference on Circuits, Systems, and Computers*, pp. 185–190, Nov. 1981.
19. J. D. Markel, "Digital Inverse Filtering: A New Tool for Formant Trajectory Estimation," *IEEE Trans. Audio and Electroacoustics*, Vol. 20, pp. 129–137, June 1972.
20. A. Gersho, "Adaptive Equalization of Highly Dispersive Channels for Data Transmission," *Bell System Tech. J.*, pp. 55–70, Jan. 1969.

42. G. A. Clark, S. K. Mitra, and S. R. Parker, "Block Implementation of Adaptive Digital Filters," *IEEE Trans. Circuits and Systems*, Vol. 28, pp. 584–592, June 1981; *IEEE Trans. Acoustics, Speech, and Signal Processing*, Joint Special Issue on Adaptive Signal Processing, Vol. 29, pp. 744–752, June 1981.

43. G. A. Clark, S. R. Parker, and S. K. Mitra, "A Unified Approach to Time- and Frequency-Domain Realization of FIR Adaptive Digital Filters," *IEEE Trans. Acoustics, Speech, and Signal Processing*, Vol. 31, pp. 1073–1083, Oct. 1983.

44. M. J. Dentino, J. McCool, and B. Widrow, "Adaptive Filtering in the Frequency Domain," *Proc. IEEE*, Vol. 66, pp. 1658–1659, Dec. 1978.

45. E. R. Ferrara, "Fast Implementation of LMS Adaptive Filters," *IEEE Trans. Acoustics, Speech, and Signal Processsing*, Vol. 28, pp. 474–475, Aug. 1980.

46. F. A. Reed and P. L. Feintuch, "A Comparison of LMS Adaptive Cancellers Implemented in the Frequency Domain and the Time Domain," *IEEE Trans. Circuits and Systems*, Vol. 28, pp. 610–615, June 1981; *IEEE Trans. Acoustics, Speech, and Signal Processing*, Joint Special Issue on Adaptive Signal Processing, Vol. 29, pp. 770–775, June 1981.

47. F. Itakura and S. Saito, "Digital Filtering Techniques for Speech Analysis and Synthesis," *Proc. 7th International Congress on Acoustics*, pp. 261–264, 1971.

48. J. Makhoul, "A Class of All-Zero Lattice Digital Filters: Properties and Applications," *IEEE Trans. Acoustics, Speech, and Signal Processing*, Vol. 26, pp. 304–314, Aug. 1978.

49. J. L. Makhoul and L. K. Cosell, "Adaptive Lattice Analysis of Speech," *IEEE Trans. Acoustics, Speech, and Signal Processing*, Vol. 29, pp. 654–659, June 1981.

50. L. J. Griffiths, "A Continuously Adaptive Filter Implemented as a Lattice Structure," *Proc. ICASSP-77*, pp. 683–686, May 1977.

51. E. I. Jury, "A Note on the Reciprocal Zeros of a Real Polynomial with Respect to the Unit Circle," *IEEE Trans. Circuit Theory*, Vol. 11, p. 292, June 1964.

52. D. T. Lee, M. Morf, and B. Friedlander, "Recursive Least Squares Ladder Estimation Algorithms," *IEEE Trans. Circuits and Systems*, Vol. 28, pp. 467–481, June 1981.

53. J. Makhoul, "Stable and Efficient Lattice Methods for Linear Prediction," *IEEE Trans. Acoustics, Speech, and Signal Processing*, Vol. 25, pp. 423–428, Oct. 1977.

54. L. J. Griffiths, "An Adaptive Lattice Structure for Noise-Cancelling Applications," *Proc. ICASSP-78*, pp. 87–90, April 1978.

55. B. Widrow and E. Walach, "On the Statistical Efficiency of the LMS Algorithm with Nonstationary Inputs," *IEEE Trans. Information Theory*, Vol. 30, pp. 211–221, March 1984.

56. A. H. Gray, Jr., and J. D. Markel, "Digital Lattice and Ladder Filter Synthesis," *IEEE Trans. Audio and Electroacoustics*, Vol. 21, pp. 491–500, Dec. 1973.

57. T. Kailath, ed., "Special Issue on System Identification and Time Series Analysis," *IEEE Trans. Automatic Control*, Dec. 1974.

58. J. Makhoul and R. Viswanathan, "Adaptive Lattice Methods for Linear Prediction," *Proc. ICASSP-78*, pp. 83–86, May 1978.

59. R. S. Medaugh and L. J. Griffiths, "A Comparison of Two Fast Linear Predictors," *Proc. ICASSP-81*, pp. 293–296, April 1981.

60. S. K. Mitra and R. J. Sherwood, "Digital Ladder Networks," *IEEE Trans. Audio and Electroacoustics*, Vol. 21, pp. 30–36, Feb. 1973.

Special IEEE Issues on Adaptive Signal Processing

Proc. IEEE, Special Issue on Adaptive Systems, Vol. 64, Aug. 1976.

IEEE Trans. Acoustics, Speech, and Signal Processing, Joint Special Issue on Adaptive Signal Processing, Vol. 29, June 1981.

IEEE Trans. Antennas and Propagation, Special Issue on Active and Adaptive Antennas, Vol. 12, March 1964.

IEEE Trans. Antennas and Propagation, Special Issue on Adaptive Antennas, Vol. 24, Sept. 1976.

IEEE Trans. Circuits and Systems, Joint Special Issue on Adaptive Signal Processing, Vol. 28, June 1981.

IEEE Trans. Information Theory, Special Issue on Adaptive Filtering, Vol. 30, March 1984.

IEEE Trans. Circuits and Systems, Special Issue on Adaptive Systems and Applications, Vol. 34, in press.

ACKNOWLEDGMENTS

The author wishes to thank the following people who criticized and helped develop this chapter: Ruth A. David, Glenn R. Elliott, Delores M. Etter, Donald Hush, Kevin Keisner, John Shynk, Otis M. Solomon, and Bernard Widrow.