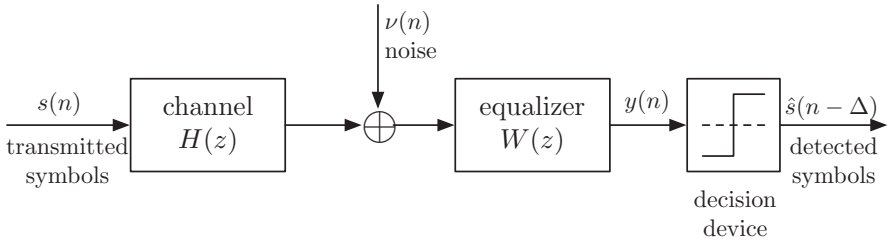

Figure 1.8 A baseband data transmission system with channel equalizer.

Channel Equalization

Figure 1.8 depicts the block diagram of a baseband transmission system equipped with a channel equalizer. Here, the channel represents the combined response of the transmitter filter, the actual channel, and the receiver front-end filter. The additive noise sequence, $\nu(n)$, arises from thermal noise in the electronic circuits and possible crosstalks from neighboring channels. The transmitted data symbols, $s(n)$, which appear in the form of amplitude/phase modulated pulses, are distorted by the channel. The most significant among the different distortions is the *pulse spreading effect*, which results because the channel impulse response is not equal to an ideal impulse function, and instead a response which is nonzero over many symbol periods. This distortion results in interference of neighboring data symbols with one another, thereby making the detection process through a simple threshold detector unreliable. The phenomenon of interference among neighboring data symbols is known as *intersymbol interference* (ISI). The presence of the additive noise samples, $\nu(n)$, further deteriorates the performance of data receivers. The role of the equalizer, as a filter, is to resolve the distortion introduced by the channel (i.e., rejection or minimization of ISI), while minimizing the effect of additive noise at the threshold detector input (equalizer output) as much as possible. If the additive noise could be ignored, the task of equalizer would be rather straightforward. For a channel $H(z)$, an equalizer with transfer function $W(z) = 1/H(z)$ could do the job perfectly as this results in an overall channel equalizer transfer function $H(z)W(z) = 1$, which implies that the transmitted data sequence, $s(n)$, will appear at the detector input without any distortion. Unfortunately, this is an ideal situation which cannot be used in most of the practical applications.

We note that the inverse of the channel transfer function, that is, $1/H(z)$, may be noncausal if $H(z)$ happens to have a zero outside the unit circle, thus making it unrealizable in practice. This problem is solved by selecting the equalizer so that $H(z)W(z) \approx z^{-\Delta}$, where Δ is an appropriate integer delay. This is equivalent to saying that a delayed replica of the transmitted symbols appears at the equalizer output. Example 3.4 of Chapter 3 clarifies the concept of noncausality of $1/H(z)$ and also the way the problem is (approximately) solved by introducing a delay, Δ. Greater details appear in Chapter 17.

We also note that the choice of $W(z) = 1/H(z)$ (or $W(z) \approx z^{-\Delta}/H(z)$) may lead to a significant enhancement of the additive noise, $\nu(n)$, in those frequency bands where the magnitude of $H(z)$ is small (i.e., $1/H(z)$ is large). Hence, in choosing an equalizer, $W(z)$, one should keep a balance between residual ISI and noise enhancement at the equalizer output. Wiener filter is a solution with such a balance (Chapter 3, Section 3.6.4).

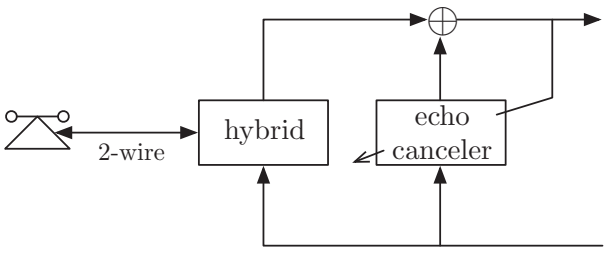

Figure 1.17 Adaptive echo canceler.

between the incoming and outgoing lines of the hybrid. By adapting the filter to realize an approximation of the echo path, a replica of the echo is obtained at its output. This is then subtracted from the outgoing signal to clear that from the undesirable echo.

Echo cancelers are usually implemented in transversal form. The time spread of echoes in a typical hybrid circuit is in the range of 20–30 ms. If we assume a sampling rate of 8 kHz for the operation of the echo canceler, an echo spread of 30 ms requires an adaptive filter with at least 240 taps (30 ms×8 kHz). This is a relatively long filter, requiring a high-speed digital signal processor for its realization. Frequency domain processing is often used to reduce the high computational complexity of long filters. The subject of frequency domain adaptive filters is covered in Chapter 8.

The echo cancelers described previously are applicable to both voice and data transmission. However, more stringent conditions need to be satisfied in the case of data transmission. To maximize the usage of the available bandwidth, full-duplex data transmission is often used. This requires the use of a hybrid circuit for connecting the data modem to the two-wire subscriber loop, as shown in Figure 1.18. The leakage of the transmitted data back to the receiver input is thus inevitable and an echo canceler has to be added, as indicated in Figure 1.18. However, we note that the data echo cancelers are different from the voice echo cancelers used in central switching offices in many ways. For instance, because the input to the data echo canceler are data symbols, it can operate at the data symbol rate that is in the range of 2.4–3 kHz (about three times smaller than the 8 kHz sampling frequency used in voice echo cancelers). For a given echo spread, a lower sampling frequency implies less number of taps for the echo canceler. Clearly, this simplifies the implementation of the echo canceler, greatly. On the other hand, the data echo cancelers require to achieve a much higher level of echo cancellation to ensure reliable transmission of data at higher bit rates. In addition, the echoes returned from the other side of the trunk lines should also be taken care of. Detailed discussions on these issues can be found in Lee and Messerschmitt (1994) and Gitlin, Hayes, and Weinstein (1992).

Acoustic Echo Cancellation

The problem of acoustic echo cancellation can be best explained by referring to Figure 1.19, which depicts the scenario that arises in teleconferencing applications. The speech signal from a far-end speaker, received through a communication channel, is broadcast by a loudspeaker in a room and its echo is picked up by a microphone.

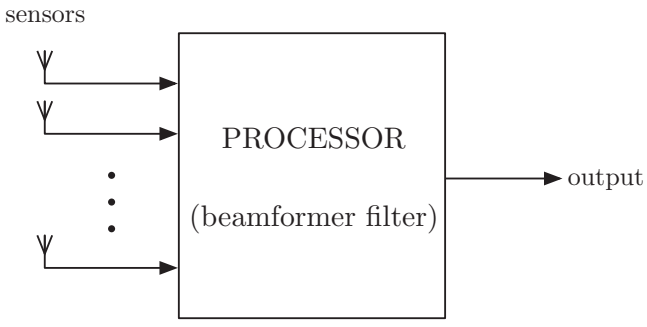

Figure 1.21 Spatial filtering (beamforming).

Beamforming

In the applications that have been discussed so far, the filters/predictors are used to combine together samples of the input signal(s) at different time instants to generate the output. Hence, these are classified as *temporal filtering*. Beamforming, however, is different from these in the sense that the inputs to a beamformer are samples of incoming signals at different positions in space. This is called *spatial filtering*. Beamforming finds applications in communication, radar and sonar (Johnson and Dudgeon, 1993), and also imaging in radar and medical engineering (Soumekh, 1994).

In spatial filtering, a number of independent sensors are placed at different points in space to pick up signals coming from various sources (Figure 1.21). In radar and communications, the signals are usually electromagnetic waves and the sensors are thus antenna elements. Accordingly, the term *antenna arrays* is often used to refer to these applications of beamformers. In sonar applications, the sensors are hydrophones designed to respond to acoustic waves.

In a beamformer, the samples of the signals picked up by the sensors at a particular instant of time constitutes a *snapshot*. The samples of snapshot (spatial samples) play the same role as the successive (temporal) samples of input in a transversal filter. The beamformer filter linearly combines the sensor signals so that signals arriving from some particular direction are amplified, while signals from other directions are attenuated. Thus, in analogy with the frequency response of temporal filters, spatial filters have responses that vary according to the direction-of-arrival of the incoming signal(s). This is given in the form of a polar plot (gain versus angle) and is referred to as *beam pattern*.

In many applications of beamformers, the signals picked up by sensors are narrow bands having the same carrier (center) frequency. These signals differ in their direction-of-arrival, which are related to the location of their sources. The operation of beamformers in such applications can be best explained by the following example.

Consider an antenna array consisting of two omnidirectional elements A and B, as presented in Figure 1.22. The tone (as approximation to narrow-band) signals $s(n) = \alpha \cos \omega_o n$ and $v(n) = \beta \cos \omega_o n$ arriving at angles 0 and θ_o (with respect to the line perpendicular to the line connecting A and B), respectively, are the inputs to the array (beamformer) filter, which consists of a phase-shifter and a subtracter. The signal $s(n)$ arrives at elements A and B at the same time, whereas the arrival times of signal $v(n)$ at

2

Discrete-Time Signals and Systems

Most of the adaptive algorithms have been developed for discrete-time (sampled) signals. Hence, discrete-time systems are used for implementation of adaptive filters. In this chapter, we present a short review of discrete-time signals and systems. Our assumption is that the reader is familiar with the basic concepts of discrete-time systems such as the Nyquist sampling theorem, z-transform and system functions, and also with the theory of random variables and stochastic processes. Our goal in this chapter, is to review these concepts and put them in a framework appropriate for the rest of the book.

2.1 Sequences and z-Transform

In discrete-time systems, we are concerned with processing signals that are represented by sequences. Such sequences may be samples of a continuous-time analog signal or may be discrete in nature. As an example, in the channel equalizer structure presented in Figure 1.9, the input sequence to the equalizer, $x(n)$, consists of the samples of the channel output which is an analog signal, but the original data sequence, $s(n)$, is discrete in nature.

A discrete-time sequence, $x(n)$, may be equivalently represented by its z-transform defined as

$$X(z) = \sum_{n=-\infty}^{\infty} x(n)z^{-n} \tag{2.1}$$

where z is a complex variable. The range of values of z for which the above summation converges is called the *region of convergence of* $X(z)$. The following two examples illustrate this.

Example 2.1

Consider the sequence

$$x_1(n) = \begin{cases} a^n, & n \geq 0 \\ 0, & n < 0 \end{cases} \tag{2.2}$$

Adaptive Filters: Theory and Applications, Second Edition. Behrouz Farhang-Boroujeny.

which is nothing but the Fourier transform of the autocorrelation function, $\phi_{xx}(k)$. We may, thus, write

$$\Phi_{xx}(e^{j\omega}) = \lim_{N\to\infty} \frac{1}{2N+1} E[|X_N(e^{j\omega})|^2] \tag{2.68}$$

The function $\Phi_{xx}(e^{j\omega})$ is called the power spectral density of the stochastic wide-sense stationary process $\{x(n)\}$. It is defined as in Eq. (2.68) or more conveniently as the Fourier transform of the autocorrelation function of $\{x(n)\}$,

$$\Phi_{xx}(e^{j\omega}) = \sum_{k=-\infty}^{\infty} \phi_{xx}(k) e^{-j\omega k} \tag{2.69}$$

The power spectral density possesses certain special properties. These are indicated below for our later reference.

Property 1: *When the limit in* Eq. (2.68) *exists*, $\Phi_{xx}(e^{j\omega})$ *has the following interpretation*:

$$\frac{1}{2\pi}\Phi_{xx}(e^{j\omega})d\omega = \text{average contribution of the frequency components of } \{x(n)\} \text{ located between } \omega \text{ and } \omega + d\omega \tag{2.70}$$

This interpretation matches Eq. (2.60) if both sides of Eq. (2.70) are integrated over ω from $-\pi$ to $+\pi$. We will elaborate more on this later, once we introduce response of linear systems to random signals; see Example 2.6.

Property 2: *The power spectral density* $\Phi_{xx}(e^{j\omega})$ *is always real and nonnegative.*

This property is obvious from the definition (Eq. 2.68), as $|X_N(e^{j\omega})|^2$ is always real and nonnegative.

Property 3: *The power spectral density of a real-valued stationary stochastic process is even, that is, symmetric with respect to the origin* $\omega = 0$. In other words,

$$\Phi_{xx}(e^{j\omega}) = \Phi_{xx}(e^{-j\omega}) \tag{2.71}$$

However, this may not be true when the process is complex-valued.

This follows from Eq. (2.69), by replacing k with $-k$ and noting that for a real-valued stationary process $\phi_{xx}(k) = \phi_{xx}(-k)$.

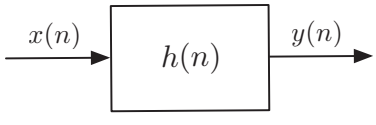

Figure 2.4 A linear time-invariant system.

P2.12 Show that for a linear time-invariant system with input $\{x(n)\}$, output $\{y(n)\}$, and system function $H(z)$

$$\Phi_{yx}(z) = H(z)\Phi_{xx}(z).$$

Also, if $\{d(n)\}$ is a third process

$$\Phi_{yd}(z) = H(z)\Phi_{xd}(z).$$

P2.13 Write the following z-transform relations in terms of the time series $h(n)$ and the correlation functions:

(i) $\Phi_{yx}(z) = H(z)\Phi_{xx}(z)$.
(ii) $\Phi_{xy}(z) = H^*(1/z^*)\Phi_{xx}(z)$.
(iii) $\Phi_{yd}(z) = H(z)\Phi_{xd}(z)$.
(iv) $\Phi_{dy}(z) = H^*(1/z^*)\Phi_{dx}(z)$.

P2.14 Consider the system shown in Figure P2.14. The input processes, $\{u(n)\}$ and $\{v(n)\}$ are zero-mean and uncorrelated with each other. Derive the relations which relate $\Phi_{uu}(z)$, $\Phi_{vv}(z)$, $H(z)$, and $G(z)$ with the following functions:

(i) $\Phi_{uy}(z)$.
(ii) $\Phi_{vy}(z)$.
(iii) $\Phi_{yy}(z)$.

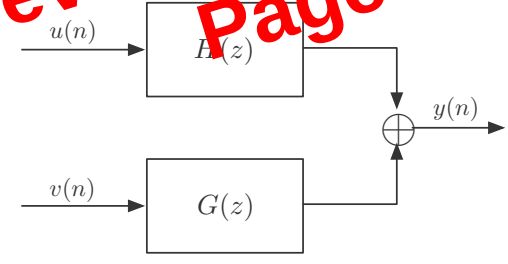

Figure P2.14

P2.15 Consider the system shown in Figure P2.15. The input, $\{v(n)\}$, is a stationary zero-mean unit-variance white noise process. Show that

(i) $\Phi_{xx}(z) = \frac{1}{(1-0.5z^{-1})(1-0.5z)}$.
(ii) $\Phi_{yy}(z) = 4$.
(iii) $\Phi_{vy}(z) = \frac{1-2z}{1-0.5z}$.

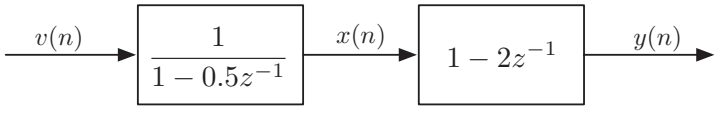

Figure P2.15

3

Wiener Filters

In this chapter, we study a class of optimum linear filters known as *Wiener filters*. As we will see in later chapters, the concept of Wiener filters is essential as well as helpful to understand and appreciate adaptive filters. Furthermore, Wiener filtering is general and applicable to any application that involves linear estimation of a desired signal sequence from another related sequence. Applications such as prediction, smoothing, joint process estimation, and channel equalization (deconvolution) are all covered by Wiener filters.

We study Wiener filters by looking at them from different angles. We first develop the theory of causal transversal Wiener filters for the case of discrete-time real-valued signals. This will then be extended to the case of complex valued signals. Our discussion follows with a study of *unconstrained Wiener filters*. The term *unconstrained* signifies that the filter impulse response is allowed to be noncausal and infinite in duration. The study of unconstrained Wiener filters is very insightful as it reveals many important aspects of Wiener filters, which otherwise would be difficult to see.

In the theory of Wiener filters, the underlying signals are assumed to be random processes, and the filter design is done using the statistics obtained by ensemble averaging. We follow this approach while doing the theoretical development and analysis of Wiener filters. However, from the implementation point of view and, in particular, while developing adaptive algorithms in later chapters, we have to consider the use of time averages instead of ensemble averages. Adoption of this approach in the development of Wiener filters is also possible, once we assume all the underlying processes are ergodic; that is, their time and ensemble averages are the same (Section 2.4.5).

3.1 Mean-Squared Error Criterion

Figure 3.1 shows the block schematic of a linear discrete-time filter $W(z)$ in the context of estimating a desired signal $d(n)$ based on an excitation $x(n)$. Here, we assume that both $x(n)$ and $d(n)$ are samples of infinite length random processes. The filter output is $y(n)$, and $e(n)$ is the estimation error. Clearly, the smaller the estimation error, the better the filter performance. As the error approaches zero, the output of the filter approaches the desired signal, $d(n)$. Hence, the question that arises is the following: what is the most appropriate choice for the parameters of the filter, which would result in the smallest possible estimation error? To a certain extent, the statement of this question itself gives

Adaptive Filters: Theory and Applications, Second Edition. Behrouz Farhang-Boroujeny.

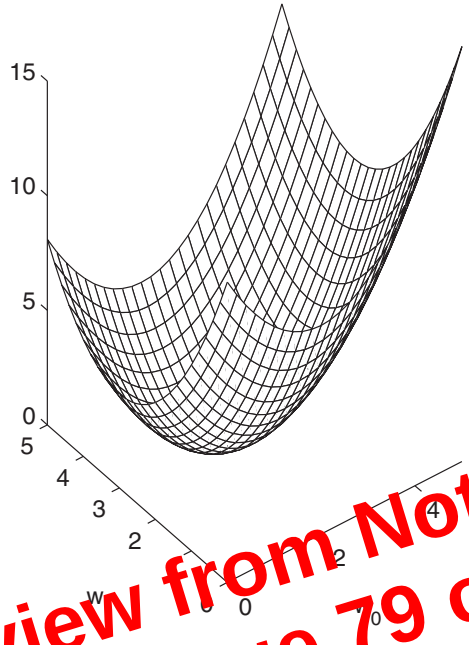

Figure 3.4 The performance surface of the modeling problem of Figure 3.3.

3.3 Principle of Orthogonality

In this section, we present an alternative approach for the design of Wiener filters. This presentation is a complement to the derivations in the last section in the sense that the approach presented below can be considered as a simplified/shortened version of the approach in the last section. More importantly, it leads to more insight into the concept of Wiener filtering problem.

We start with the cost function equation (3.1), which in the case of real-valued data may be written as

$$\xi = E[e^2(n)] \tag{3.36}$$

Taking partial derivatives of ξ with respect to the filter tap weights, $\{w_i; i = 0, 1, \ldots, N-1\}$, and interchanging the derivative and expectation operators (since these are linear operators), we obtain

$$\frac{\partial \xi}{\partial w_i} = E\left[2e(n)\frac{\partial e(n)}{\partial w_i}\right], \quad \text{for } i = 0, 1, \ldots, N-1 \tag{3.37}$$

where $e(n) = d(n) - y(n)$. As $d(n)$ is independent of the filter tap weights, we get

$$\frac{\partial e(n)}{\partial w_i} = -\frac{\partial y(n)}{\partial w_i} = -x(n-i) \tag{3.38}$$

Squaring both sides of Eq. (3.48) and taking expectation, we get

$$E[d^2(n)] = E[e_o^2(n)] + E[y_o^2(n)] + 2E[e_o(n)y_o(n)] \tag{3.49}$$

We may note that $E[e_o^2(n)] = \xi_{min}$, and the last term in Eq. (3.49) is zero because of Eq. (3.42). Thus, we obtain

$$\xi_{min} = E[d^2(n)] - E[y_o^2(n)] \tag{3.50}$$

which suggests that *the minimum mean-squared error at the Wiener filter output is the difference between the mean-squared of the desired output and the mean-squared of the best estimate of that at the filter output*.

It is appropriate if we define the ratio

$$\zeta = \frac{\xi}{E[d^2(n)]} \tag{3.51}$$

as the normalized performance function. We may note that $\zeta = 1$ when $y(n)$ is forced to zero; that is, when no estimation of $d(n)$ has been made. It reaches its minimum value, ζ_{min}, when the filter tap weights are chosen to achieve the minimum mean-squared error. This is given by

$$\zeta_{min} = 1 - \frac{E[y_o^2(n)]}{E[d^2(n)]} \tag{3.52}$$

Noting that ζ_{min} cannot be negative, we find that its value remains between 0 and 1. *The value of ζ_{min} is an indication of the ability of the filter in estimating the desired output. A value of ζ_{min} close to zero is an indication of good performance of the filter, and a value of ζ_{min} close to 1 indicates poor performance of the filter.*

3.5 Extension to Complex-Valued Case

There are some practical applications in which the underlying random processes are complex-valued. For instance, in data transmission (Chapter 17), the most frequently used signaling techniques are phase shift keying (PSK) and quadrature-amplitude modulation (QAM) in which the baseband signal consists of two separate components which are the real and imaginary parts of a complex-valued signal. Moreover, in the case of frequency domain implementation of adaptive filters (Chapter 8) and subband-adaptive filters (Chapter 9), we will be dealing with complex-valued signals, even though the original signals may be real-valued.

In this section, we extend the results of the last two sections to the case of complex-valued signals. We assume a transversal filter as shown in Figure 3.2. The input, $x(n)$, the desired output, $d(n)$, and the filter tap weights are all assumed to be complex-valued. Then, the estimation error, $e(n)$, is also complex-valued and we may write

$$\xi = E[|e(n)|^2] = E[e(n)e^*(n)] \tag{3.53}$$

where the asterisk denotes complex conjugation.

As in the real-valued case, the performance function, ξ, in the complex-valued case is also a quadratic function of the filter tap weights. Similarly, to find the optimum set of the

Noting that $s(n)$ and $v(n)$ are uncorrelated with each other and recalling the results of Section 2.4.4, we obtain, from Eq. (3.131)

$$\Phi_{xx}(z) = \Phi_{vv}(z) + \Phi_{ss}(z)|H(z)|^2 \tag{3.133}$$

To find $\Phi_{dx}(z)$, we note that $d(n)$ and $x(n)$ are related with each other through the signal sequences $s(n)$ and $v(n)$ and the filters $H(z)$ and $G(z)$. As $s(n)$ and $v(n)$ are uncorrelated with each other, their contribution in $\Phi_{dx}(z)$ may be considered separately. In particular, we may write

$$\Phi_{dx}(z) = \Phi^s_{dx}(z) + \Phi^v_{dx}(z) \tag{3.134}$$

where $\Phi^s_{dx}(z)$ is $\Phi_{dx}(z)$ when $v(n) = 0$, for all values of n, and $\Phi^v_{dx}(z)$ is $\Phi_{dx}(z)$ when $s(n) = 0$, for all values of n. Thus, we obtain

$$\Phi^s_{dx}(z) = H^*(z)\Phi_{ss}(z) \tag{3.135}$$

and

$$\Phi^v_{dx}(z) = G(z)\Phi_{vv}(z) \tag{3.136}$$

Substituting Eqs. (3.135) and (3.136) in Eq. (3.134), we get

$$\Phi_{dx}(z) = H^*(z)\Phi_{ss}(z) + G(z)\Phi_{vv}(z) \tag{3.137}$$

Using Eqs. (3.133) and (3.137) in Eq. (3.132), we obtain

$$W_o(z) = \frac{H^*(z)\Phi_{ss}(z) + G(z)\Phi_{vv}(z)}{\Phi_{vv}(z) + \Phi_{ss}(z)|H(z)|^2} \tag{3.138}$$

A comparison of Eq. (3.138) with Eqs. (3.106) and (3.125) reveals that Eq. (3.138) may be thought as a generalization of the results we obtained in the last two sections for the modeling and inverse modeling scenarios. In fact, if we refer to Figure 3.12, we can easily find that the modeling and inverse modeling scenarios are embedded in the noise canceler setup. While trying to minimize the mean-squared value of the output error, one must strike a balance between noise cancellation and signal cancellation at the output of the noise canceler. Cancellation of the noise $v(n)$ occurs when the Wiener filter $W(z)$ is chosen to be close to $G(z)$, and cancellation of the signal $s(n)$ occurs when $W(z)$ is close to the inverse of $H(z)$. In this sense, we may note that the noise canceler treats $s(n)$ and $v(n)$ without making any distinction between them and tries to cancel both of them as much as possible so as to achieve the minimum mean-squared error in $e(n)$. This seems contrary to the main goal of the noise canceler, which is meant to cancel only the noise. The following discussion aims at revealing some of the peculiar characteristics of the noise canceler setup and show under which condition an acceptable cancellation occurs.

To proceed with our discussion, we define $\rho_{\text{pri}}(e^{j\omega})$, $\rho_{\text{ref}}(e^{j\omega})$, and $\rho_{\text{out}}(e^{j\omega})$ as the signal-to-noise power spectral density ratios at the primary input, reference input, and output, respectively. By direct inspection of Figure 3.12 and application of Eq. (2.85), we obtain

$$\rho_{\text{pri}}(e^{j\omega}) = \frac{\Phi_{ss}(e^{j\omega})}{|G(e^{j\omega})|^2\Phi_{vv}(e^{j\omega})} \tag{3.139}$$

(ii) For your choice in (i), find the optimum values of the filter tap weights.
(iii) Find an expression for the signal and jammer components reaching the canceler output, and confirm the power inversion formula.

P3.28 Consider an array of three omnidirectional antennas as shown in Figure P3.28. The signal, $s(n)$, and jammer, $v(n)$, are narrow-band processes, as in Example 3.6.

(i) Find the optimum values of the filter tap weights, which minimize the mean-squared of the output error, $e(n)$.
(ii) Find an expression for the canceler output, and confirm the power inversion formula.

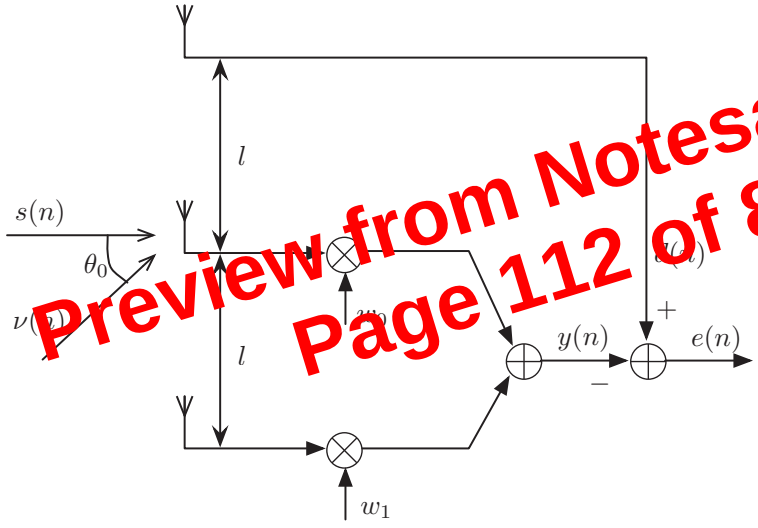

Figure P3.28

P3.29 Repeat P3.28 for the array shown in Figure P3.29, and compare the results obtained with those of P3.28.

P3.30 To prevent time averages and derive the results presented in Example 3.6 through ensemble averages, the desired signal and jammer may be redefined as $s(n) = \alpha(n)\cos(n\omega_o + \varphi_1)$ and $v(n) = \beta(n)\cos(n\omega_o + \varphi_2)$, respectively, where φ_1 and φ_2 are random initial phases of the carrier, and assumed to be uniformly distributed in the interval $-\pi$ to $+\pi$. The amplitudes $\alpha(n)$ and $\beta(n)$, as in Example 3.6, are uncorrelated narrow-band baseband signals. Furthermore, the random phases φ_1 and φ_2 are assumed to be independent among themselves as well as with respect to $\alpha(n)$ and $\beta(n)$.

(i) Using the new definitions of $s(n)$ and $v(n)$, show that the same result as in Eq. (3.160) is also obtained through ensemble averages.

4

Eigenanalysis and Performance Surface

The transversal Wiener filter was introduced in the last chapter as a powerful signal processing structure with a unique performance function, which has many desirable features for adaptive filtering applications. In particular, it was noted that the performance function of the transversal Wiener filter has a unique global minimum point, which can be easily obtained using the second-order moments of the underlying processes. This is a consequence of the fact that the performance function of the transversal Wiener filter is a convex quadratic function of its tap weights.

Our goal in this chapter is to analyze the quadratic performance function of the transversal Wiener filter in detail. We get a clear picture of the shape of the performance function when it is visualized as a surface in the $(N+1)$-dimensional space of variables consisting of the filter tap weights, as the first N axes, and the performance function, as the $(N+1)$th axis. This is called *performance surface*.

The shape of the performance surface of a transversal Wiener filter is closely related to the eigenvalues of the correlation matrix $\mathbf{R}$ of the filter tap inputs. Hence, we start with a thorough discussion on the eigenvalues and eigenvectors of the correlation matrix $\mathbf{R}$.

4.1 Eigenvalues and Eigenvectors

Let

$$\mathbf{R} = E[\mathbf{x}(n)\mathbf{x}^{\mathrm{H}}(n)] \tag{4.1}$$

be the N-by-N correlation matrix of a complex-valued wide-sense stationary stochastic process represented by the N-by-1 observation vector $\mathbf{x}(n) = [x(n)\ x(n-1)\cdots x(n-N+1)]^{\mathrm{T}}$, where the superscripts H and T denote Hermitian and transpose, respectively.

A nonzero N-by-1 vector $\mathbf{q}$ is said to be an *eigenvector* of $\mathbf{R}$, if it satisfies the equation

$$\mathbf{R}\mathbf{q} = \lambda\mathbf{q} \tag{4.2}$$

for some scalar constant λ. The scalar λ is called the *eigenvalue* of $\mathbf{R}$ associated with the eigenvector $\mathbf{q}$. We note that if $\mathbf{q}$ is an eigenvector of $\mathbf{R}$, then for any nonzero scalar a,

Adaptive Filters: Theory and Applications, Second Edition. Behrouz Farhang-Boroujeny.

and the order of the eigenvalues $\lambda_0, \lambda_1, \ldots, \lambda_{N-1}$ *matches that of the corresponding eigenvectors in the columns of* $\mathbf{Q}$.

To prove this property, we note that the set of equations

$$\mathbf{R}\mathbf{q}_i = \lambda_i \mathbf{q}_i, \quad \text{for } i = 0, 1, \ldots, N-1 \tag{4.21}$$

may be packed together as a single matrix equation

$$\mathbf{R}\mathbf{Q} = \mathbf{Q}\mathbf{\Lambda}. \tag{4.22}$$

Then, postmultiplying Eq. (4.22) by $\mathbf{Q}^{\mathrm{H}}$ and noting that $\mathbf{Q}\mathbf{Q}^{\mathrm{H}} = \mathbf{I}$, we can get Eq. (4.19). The right-hand side of Eq. (4.19) may be expanded as

$$\mathbf{R} = \sum_{i=0}^{N-1} \lambda_i \mathbf{q}_i \mathbf{q}_i^{\mathrm{H}} \tag{4.23}$$

Property 6: *Let* $\lambda_0, \lambda_1, \ldots, \lambda_{N-1}$ *be the eigenvalues of the correlation matrix* $\mathbf{R}$. *Then,*

$$\mathrm{tr}[\mathbf{R}] = \sum_{i=0}^{N-1} \lambda_i \tag{4.24}$$

where $\mathrm{tr}[\mathbf{R}]$ *denotes trace of* $\mathbf{R}$ *and is defined as the sum of the diagonal elements of* $\mathbf{R}$.

Taking the trace on both sides of Eq. (4.19), we get

$$\mathrm{tr}[\mathbf{R}] = \mathrm{tr}[\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{\mathrm{H}}] \tag{4.25}$$

To proceed, we may use the following result of matrix algebra. If $\mathbf{A}$ and $\mathbf{B}$ are N-by-M and M-by-N matrices, respectively, then,

$$\mathrm{tr}[\mathbf{A}\mathbf{B}] = \mathrm{tr}[\mathbf{B}\mathbf{A}] \tag{4.26}$$

Using this result, we may swap $\mathbf{Q}\mathbf{\Lambda}$ and $\mathbf{Q}^{\mathrm{H}}$ on the right-hand side of Eq. (4.25). Then, noting that $\mathbf{Q}^{\mathrm{H}}\mathbf{Q} = \mathbf{I}$, Eq. (4.25) is simplified as

$$\mathrm{tr}[\mathbf{R}] = \mathrm{tr}[\,\mathbf{\Lambda}\,] \tag{4.27}$$

Using the definition (4.20) in Eq. (4.27) completes the proof.
An alternative way of proving the above result is by direct expansion of Eq. (4.4); see Problem P4.8. This proof shows that the identity (4.24) is not limited to the Hermitian matrices. It applies to any square matrix.

Property 7: Minimax Theorem.[1] *The distinct eigenvalues* $\lambda_0 > \lambda_1 > \cdots > \lambda_{N-1}$ *of the correlation matrix* $\mathbf{R}$ *of an observation vector* $\mathbf{x}(n)$, *and their corresponding eigenvectors,* $\mathbf{q}_0, \mathbf{q}_1, \ldots \mathbf{q}_{N-1}$, *may be obtained through the following optimization procedure*:

$$\lambda_{\max} = \lambda_0 = \max_{\|\mathbf{q}_0\|=1} E[|\mathbf{q}_0^{\mathrm{H}}\mathbf{x}(n)|^2] \tag{4.28}$$

[1] In matrix algebra literature, the minimax theorem is usually stated using the Hermitian form $\mathbf{q}_i^{\mathrm{H}}\mathbf{R}\mathbf{q}_i$ instead of $E[|\mathbf{q}^{\mathrm{H}}\mathbf{x}(n)|^2]$ (Haykin, 1991). The method that we have adopted here is to simplify some of our discussions in the following chapters. This method has been adopted from Farhang-Boroujeny and Gazor (1992).

Also, using Eq. (2.80) of Chapter 2 and noting that α is real-valued, we obtain

$$\Phi_{xx}(z) = H(z)H(z^{-1})\Phi_{vv}(z) = \frac{1-\alpha^2}{(1-\alpha z^{-1})(1-\alpha z)}. \tag{4.72}$$

Taking inverse z-transform, we get

$$\phi_{xx}(k) = \alpha^{|k|}, \quad \text{for } k = \ldots -2, -1, 0, 1, 2, \ldots \tag{4.73}$$

Using this result, we find that the correlation matrix of an N-tap transversal filter with input $\{x(n)\}$ is

$$\mathbf{R} = \begin{bmatrix} 1 & \alpha & \alpha^2 & \cdots & \alpha^{N-1} \\ \alpha & 1 & \alpha & \cdots & \alpha^{N-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \alpha^{N-1} & \alpha^{N-2} & \alpha^{N-3} & \cdots & 1 \end{bmatrix}. \tag{4.74}$$

Next, we present some numerical results, which demonstrate the relationships between the power spectral density of the process $\{x(n)\}$, $\Phi_{xx}(e^{j\omega})$, and its corresponding correlation matrix.

Figure 4.1 shows a set of the plots of $\Phi_{xx}(e^{j\omega})$ for values of $\alpha = 0$, 0.5, and 0.75. We note that $\alpha = 0$ corresponds to the case where $\{x(n)\}$ is white and, therefore, its power spectral density is flat. As α increases from 0 to 1, $\{x(n)\}$ becomes more colored and for values of α close to 1, most of its energy is concentrated around $\omega = 0$.

From Property 8, we recall that the eigenvalues of the correlation matrix $\mathbf{R}$ are bounded by the minimum and maximum values of $\Phi_{xx}(e^{j\omega})$. To illustrate this, in Figure 4.2a, b, and c, we have plotted the minimum and maximum eigenvalues of $\mathbf{R}$ for values of $\alpha = 0.5$, 0.75, and 0.9, as N varies from 2 to 20. It may be noted that the limits predicted by the minimum and maximum values of $\Phi_{xx}(e^{j\omega})$ are achieved asymptotically as N increases.

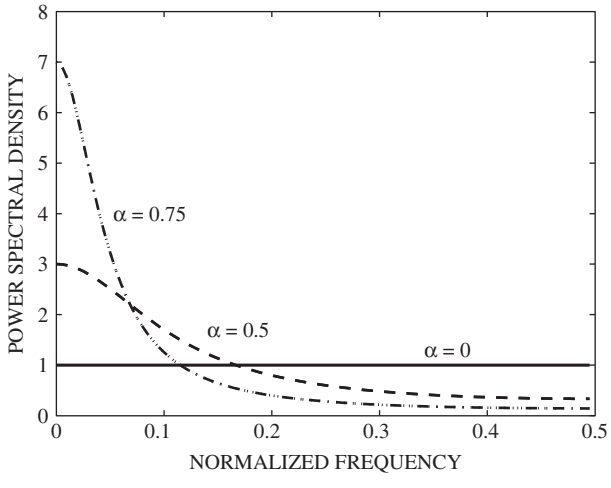

Figure 4.1 Power spectral density of $\{x(n)\}$ for different values of the parameter α.

4.3 Performance Surface

With the background developed so far, we are now ready to proceed with exploring the performance surface of transversal Wiener filters. We start with the case where the filter coefficients, input, and desired output are real-valued. The results will then be extended to the complex-valued case.

We recall from Chapter 3 that the performance function of a transversal Wiener filter with a real-valued input sequence $x(n)$ and a desired output sequence $d(n)$ is

$$\xi = \mathbf{w}^{\mathrm{T}}\mathbf{R}\mathbf{w} - 2\mathbf{p}^{\mathrm{T}}\mathbf{w} + E[d^2(n)] \tag{4.81}$$

where the superscript T denotes vector or matrix transpose, $\mathbf{w} = [w_0 \;\; w_1 \cdots \; w_{N-1}]^{\mathrm{T}}$ is the filter tap-weight vector, $\mathbf{R} = E[\mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n)]$ is the correlation matrix of the filter tap-input vector $\mathbf{x}(n) = [x(n) \;\; x(n-1) \cdots x(n-N+1)]^{\mathrm{T}}$, and $\mathbf{p} = E[d(n)\mathbf{x}(n)]$ is the cross-correlation vector between $d(n)$ and $\mathbf{x}(n)$. We want to study the shape of the performance function ξ when it is viewed as a surface in the $(N+1)$-dimensional Euclidean space constituted by the filter tap weights w_i, $i = 0, 1, \ldots, N-1$, and the performance function, ξ.

Also, we recall that the optimum value of the Wiener filter tap-weight vector is obtained from the Wiener-Hopf equation

$$\mathbf{R}\mathbf{w}_{\mathrm{o}} = \mathbf{p} \tag{4.82}$$

The performance function ξ may be rearranged as follows:

$$\xi = \mathbf{w}^{\mathrm{T}}\mathbf{R}\mathbf{w} - \mathbf{w}^{\mathrm{T}}\mathbf{p} - \mathbf{p}^{\mathrm{T}}\mathbf{w} + E[d^2(n)] \tag{4.83}$$

where we have noted that $\mathbf{w}^{\mathrm{T}}\mathbf{p} = \mathbf{p}^{\mathrm{T}}\mathbf{w}$. Next, we substitute for $\mathbf{p}$ in Eq. (4.83) from Eq. (4.82) and add and subtract the term $\mathbf{w}_{\mathrm{o}}^{\mathrm{T}}\mathbf{R}\mathbf{w}_{\mathrm{o}}$ to obtain

$$\xi = \mathbf{w}^{\mathrm{T}}\mathbf{R}\mathbf{w} - \mathbf{w}^{\mathrm{T}}\mathbf{R}\mathbf{w}_{\mathrm{o}} - \mathbf{w}_{\mathrm{o}}^{\mathrm{T}}\mathbf{R}^{\mathrm{T}}\mathbf{w} + \mathbf{w}_{\mathrm{o}}^{\mathrm{T}}\mathbf{R}\mathbf{w}_{\mathrm{o}} + E[d^2(n)] - \mathbf{w}_{\mathrm{o}}^{\mathrm{T}}\mathbf{R}\mathbf{w}_{\mathrm{o}} \tag{4.84}$$

As $\mathbf{R}^{\mathrm{T}} = \mathbf{R}$, the first four terms on the right-hand side of Eq. (4.84) can be combined to obtain

$$\xi = (\mathbf{w} - \mathbf{w}_{\mathrm{o}})^{\mathrm{T}}\mathbf{R}(\mathbf{w} - \mathbf{w}_{\mathrm{o}}) + E[d^2(n)] - \mathbf{w}_{\mathrm{o}}^{\mathrm{T}}\mathbf{R}\mathbf{w}_{\mathrm{o}} \tag{4.85}$$

We may also recall from Chapter 3 that

$$\xi_{\min} = E[d^2(n)] - \mathbf{w}_{\mathrm{o}}^{\mathrm{T}}\mathbf{R}\mathbf{w}_{\mathrm{o}} \tag{4.86}$$

where $\xi_{\min}$ is the minimum value of ξ, which is obtained when $\mathbf{w} = \mathbf{w}_{\mathrm{o}}$. Substituting Eq. (4.86) in Eq. (4.85), we get

$$\xi = \xi_{\min} + (\mathbf{w} - \mathbf{w}_{\mathrm{o}})^{\mathrm{T}}\mathbf{R}(\mathbf{w} - \mathbf{w}_{\mathrm{o}}) \tag{4.87}$$

This result has the following interpretation. The nonnegative definiteness of the correlation matrix $\mathbf{R}$ implies that the second term on the right-hand side of Eq. (4.87) is nonnegative. When $\mathbf{R}$ is positive definite (a case very likely to happen in practice), the second term on the right-hand side of Eq. (4.87) is zero only when $\mathbf{w} = \mathbf{w}_{\mathrm{o}}$, and in that case ξ coincides with its minimum value. This is depicted in Figure 4.4 where a typical performance

Hint: Note that $\mathbf{R}_I^{\mathrm{T}} = -\mathbf{R}_I$ and this implies that for any arbitrary vector $\mathbf{v}$, $\mathbf{v}^{\mathrm{T}}\mathbf{R}_I\mathbf{v} = 0$.

(ii) Show that equation

$$\mathbf{R}\mathbf{q}_i = \lambda_i \mathbf{q}_i \tag{P4.14.1}$$

implies

$$\begin{bmatrix} \mathbf{R}_R & -\mathbf{R}_I \\ \mathbf{R}_I & \mathbf{R}_R \end{bmatrix} \begin{bmatrix} \mathbf{q}_{i,R} \\ \mathbf{q}_{i,I} \end{bmatrix} = \lambda_i \begin{bmatrix} \mathbf{q}_{i,R} \\ \mathbf{q}_{i,I} \end{bmatrix}.$$

Also, multiplying Eq. (P4.14.1) through by $j = \sqrt{-1}$, we get $\mathbf{R}(j\mathbf{q}_i) = \lambda_i(j\mathbf{q}_i)$. Show that this implies

$$\begin{bmatrix} \mathbf{R}_R & -\mathbf{R}_I \\ \mathbf{R}_I & \mathbf{R}_R \end{bmatrix} \begin{bmatrix} -\mathbf{q}_{i,I} \\ \mathbf{q}_{i,R} \end{bmatrix} = \lambda_i \begin{bmatrix} -\mathbf{q}_{i,I} \\ \mathbf{q}_{i,R} \end{bmatrix}.$$

Relate these with Eq. (4.113).

Computer-Oriented Problems

The following problems involve numerical evaluation/analysis of large matrices. MATLAB will work best for completing the solutions to these problems.

P4.15 Consider a random process $x(n) = \nu(n) + \cos(0.3\pi n + \theta)$, where $\nu(n)$ is a white process with power spectral density $\Phi_{\nu\nu}(e^{j\omega}) = \sigma_\nu^2$ and θ is a random variable uniformly distributed in the interval 0 to 2π.

(i) Find an expression for the autocorrelation coefficients $\phi_{xx}(k)$.

(ii) Using the result of (i), find an expression for the power spectral density $\Phi_{xx}(e^{j\omega})$.

(iii) Present the autocorrelation matrix $\mathbf{R}$ of $\mathbf{x}(n) = [x(n)\ \ x(n-1)\cdots\ x(n-N+1)]^{\mathrm{T}}$.

(iv) For $N = 10$ and $\sigma_\nu^2 = 0.01$, find the numerical results for the eigenvectors and eigenvalues of $\mathbf{R}$.

(v) Repeat (iv), for $N = 10$ and the choices of $\sigma_\nu^2 = 0$ and 0.1 and compare your results of the three choices of σ_ν^2. Explain any relationship that you may find and explain your findings with the theoretical results in this chapter.

(vi) Present a plot of $\Phi_{xx}(e^{j\omega})$, in a form similar to Figure 4.3, within the normalized frequency range 0 to 1.

(vii) For the numerical results evaluated in (iv) and (v), add the plots of the magnitude responses of the associated eigenfilters to the results of (vi). Make the observation that all eigenfilters, except one, have zeros at the normalized frequencies 0.15 and 0.85. Following the argument made in Example 4.2, explain this observation.

P4.16 Repeat Problem P4.15 for the case where $x(n) = \nu(n) + e^{j0.3\pi n+\theta}$ and explain differences that you observe in the results compared to those in Problem P4.15.

P4.17 Consider a random process $x(n) = \nu(n) + \cos(0.3\pi n + \theta_1) + \cos(0.5\pi n + \theta_2)$, where $\nu(n)$ is a white process with power spectral density $\Phi_{\nu\nu}(e^{j\omega}) = \sigma_\nu^2$, and θ_1 and θ_2 are two independent random variables both uniformly distributed in the interval 0 to 2π.

(i) Find an expression for the autocorrelation coefficients $\phi_{xx}(k)$.
(ii) Using the result of (i), find an expression for the power spectral density $\Phi_{xx}(e^{j\omega})$.
(iii) Present the autocorrelation matrix $\mathbf{R}$ of $\mathbf{x}(n) = [x(n)\ x(n-1) \cdots x(n-N+1)]^T$.
(iv) For $N = 10$ and $\sigma_v^2 = 0.01$, find the numerical results for the eigenvectors and eigenvalues of $\mathbf{R}$.
(v) Repeat (iv), for $N = 10$ and the choices of $\sigma_v^2 = 0$ and 0.1 and compare your results of the three choices of σ_v^2. Explain any relationship that you may find and explain your findings with the theoretical results in this chapter.
(vi) Present a plot of $\Phi_{xx}(e^{j\omega})$, in a form similar to Figure 4.3, within the normalized frequency range 0 to 1.
(vii) For the numerical results evaluated in (iv) and (v), add the plots of the magnitude responses of the associated eigenfilters to the results of (vi). Make the observation that all eigenfilters, except two, have zeros at the normalized frequencies 0.15, 0.25, 0.75, and 0.85. Expand the argument made in Example 4.2 to explain this observation.

P4.18 Consider the case where a random process $\{x(n)\}$ is generated as in Figure P4.18. Let $\mathbf{x}(n) = [x(n)\ x(n-1)\ \cdots\ x(n-N+1)]^T$.

(i) Find and present the correlation matrix $\mathbf{R} = E[\mathbf{x}(n)\mathbf{x}^H(n)]$.
(ii) Find and present the eigenvalues, λ_i, and eigenvectors, $\mathbf{q}_i$, of $\mathbf{R}$.
(iii) Present a plot of the power spectral density $\Phi_{xx}(e^{j\omega})$.
(iv) Add the plots of the magnitude responses of the eigenfilters of $\mathbf{R}$ to the power spectral density plot in Part (c).
(v) Make an attempt to relate the plots in Part (d) to the eigenvalues in Part (b).

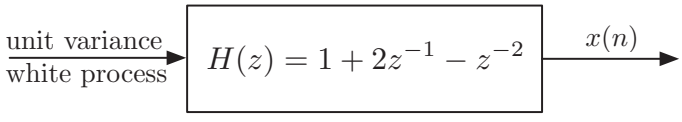

Figure P4.18

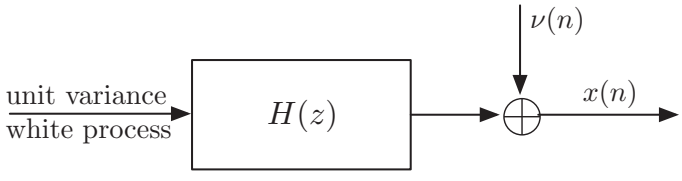

Figure P4.20

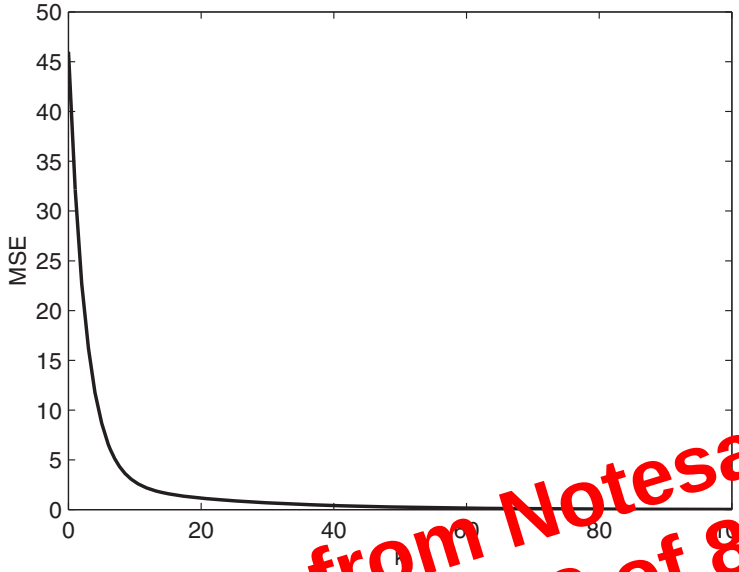

Figure 5.5 A learning curve of the modeling problem. The ξ (MSE) axis is scaled linearly.

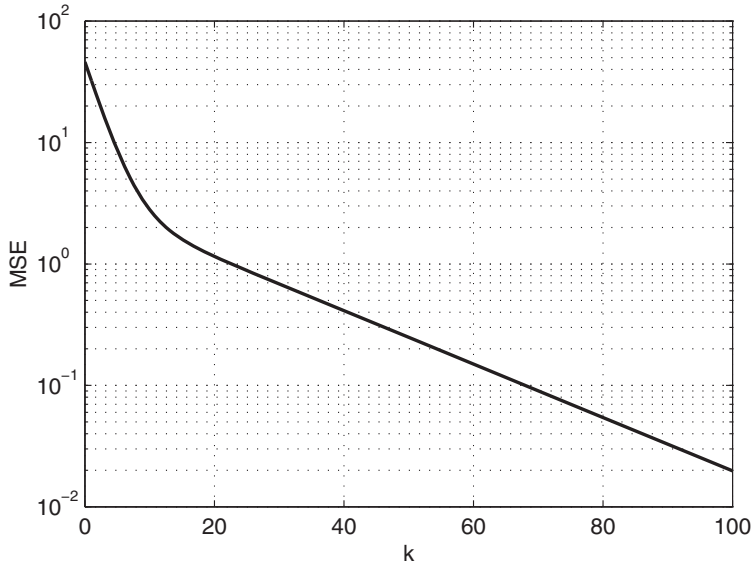

Figure 5.6 A learning curve of the modeling problem. The ξ (MSE) axis is scaled logarithmically.

method does. To derive Newton's method for the quadratic case, we start from the steepest-descent algorithm given in Eq. (5.10). Using $\mathbf{p} = \mathbf{R}\mathbf{w}_o$, Eq. (5.10) becomes

$$\mathbf{w}(k+1) = \mathbf{w}(k) - 2\mu\mathbf{R}(\mathbf{w}(k) - \mathbf{w}_o) \tag{5.43}$$

We may note that it is the presence of $\mathbf{R}$ in Eq. (5.43), which causes the eigenvalue spread problem in the steepest-descent algorithm. Newton's method overcomes this problem by replacing the scalar step-size parameter μ with a matrix step-size given by $\mu\mathbf{R}^{-1}$. The resulting algorithm is

$$\mathbf{w}(k+1) = \mathbf{w}(k) - \mu\mathbf{R}^{-1}\nabla_k\xi \tag{5.44}$$

Figure 5.8 demonstrates the effect of the addition of $\mathbf{R}^{-1}$ in front of the gradient vector in Newton's update Eq. (5.44). This has the effect of rotating the gradient vector to the direction pointing toward the minimum point of the performance surface.

Substituting Eq. (5.7) in Eq. (5.44), we obtain

$$\begin{aligned}\mathbf{w}(k+1) &= \mathbf{w}(k) - 2\mu\mathbf{R}^{-1}(\mathbf{R}\mathbf{w}(k) - \mathbf{p}) \\ &= (1-2\mu)\mathbf{w}(k) + 2\mu\mathbf{R}^{-1}\mathbf{p}\end{aligned} \tag{5.45}$$

We also note that $\mathbf{R}^{-1}\mathbf{p}$ is equal to the optimum tap-weight vector $\mathbf{w}_o$. Using this in Eq. (5.45), we obtain

$$\mathbf{w}(k+1) = (1-2\mu)\mathbf{w}(k) + 2\mu\mathbf{w}_o \tag{5.46}$$

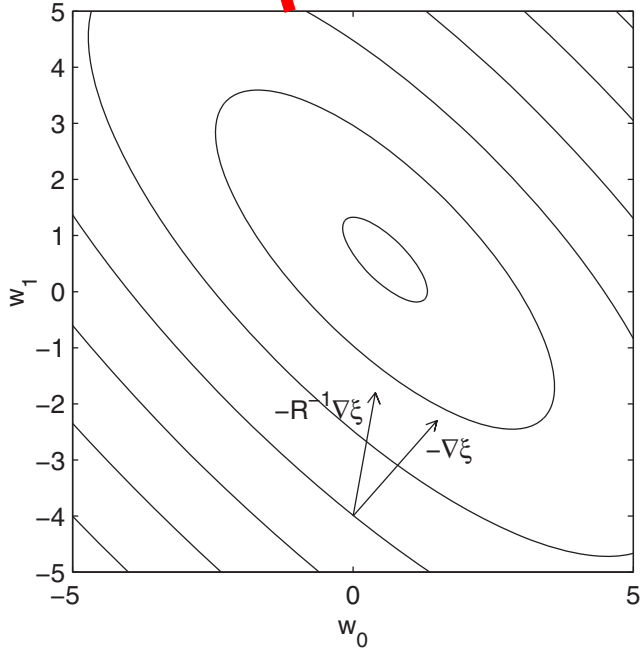

Figure 5.8 The negative gradient vector and its correction by Newton's method.

(ii)
$$\lim_{n\to\infty}(\mathbf{I}+(\mathbf{I}-\mathbf{R})+(\mathbf{I}-\mathbf{R})^2+\cdots+(\mathbf{I}-\mathbf{R})^n)=\mathbf{R}^{-1}$$

P5.9 Consider the modeling problem depicted in Figure P5.6. Note that the input to the model is a noisy version of the plant input. The additive noise at the model input, $v_i(n)$, is white and its variance is σ_i^2. The sequence $v_o(n)$ is the plant noise. It is uncorrelated with $u(n)$ and $v_i(n)$. The correlation matrix of the plant input, $u(n)$, is denoted by $\mathbf{R}$. The model has to be selected so that the MSE at the model output is minimized.

(i) Find the correlation matrix of the model input and show that it shares the same set of eigenvectors with $\mathbf{R}$.
(ii) Derive the corresponding Wiener–Hopf equation.
(iii) Show that the difference between the plant tap-weight vector, $\mathbf{w}_o$, and its estimate, $\hat{\mathbf{w}}_o$, which is obtained through the Wiener–Hopf equation derived in (ii), is
$$\mathbf{w}_o-\hat{\mathbf{w}}_o=\sigma_i^2\sum_{l=0}^{N-1}\frac{\mathbf{q}_l^T\mathbf{p}}{\lambda_l(\lambda_l+\sigma_i^2)}\mathbf{q}_l$$
where $\mathbf{q}_l$s are the eigenvectors of $\mathbf{R}$ and $\mathbf{p}$ is the cross-correlation between the model input and the desired output.
(iv) Show that
$$\text{MMSE}=\sigma_o^2+\sigma_i^2\sum_{l=0}^{N-1}\frac{(\mathbf{q}_l^T\mathbf{p})^2}{\lambda_l(\lambda_l+\sigma_i^2)^2}$$
(v) If the steepest-descent algorithm is used to find $\hat{\mathbf{w}}_o$, find the time constants of the resulting learning curve. How do these time constants vary with σ_i^2? Discuss on the eigenvalue spread problem as σ_i^2 varies.

P5.10 Consider a transversal filter with the input and tap-weight vectors $\mathbf{x}(n)$ and $\mathbf{w}$, respectively, and output
$$y(n)=\mathbf{w}^T\mathbf{x}(n)$$
Define the vector
$$\breve{\mathbf{x}}(n)=\mathbf{R}^{-1/2}\mathbf{x}(n)$$
where $\mathbf{R}=E[\mathbf{x}(n)\mathbf{x}^T(n)]$. Let $\breve{\mathbf{x}}(n)$ be the input to a filter whose output is obtained through the equation
$$\breve{y}(n)=\breve{\mathbf{w}}^T\breve{\mathbf{x}}(n)$$
where $\breve{\mathbf{w}}$ is the filter tap-weight vector.

(i) Derive an equation for $\breve{\mathbf{w}}$ so that the two outputs $y(n)$ and $\breve{y}(n)$ be the same.
(ii) Derive a steepest-descent update equation for the tap-weight vector $\breve{\mathbf{w}}$.
(iii) Derive an equation that demonstrates the variation of the tap weights of the filter as the steepest-descent algorithm derived in Part (ii) is running.
(iv) Find the time constants of the learning curve of the algorithm.
(v) Show that the update equation derived in (ii) is equivalent to Newton's algorithm.

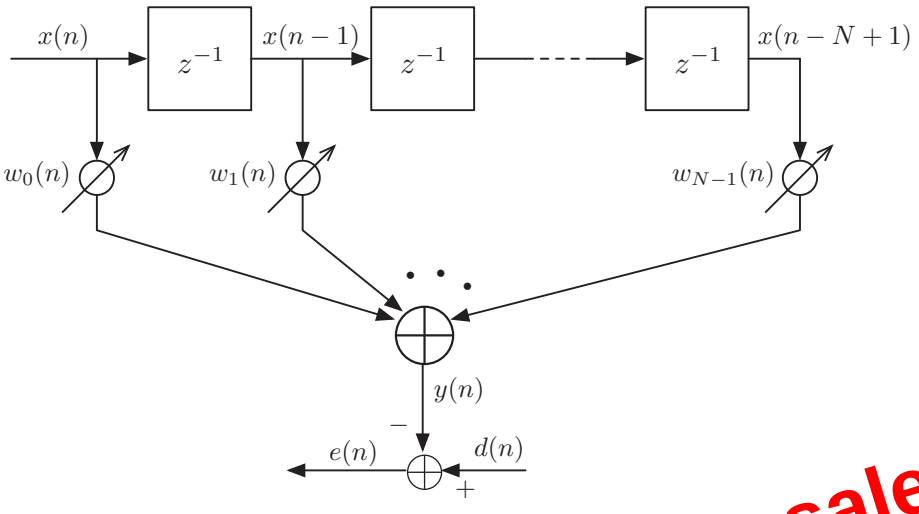

Figure 6.1 An N-tap transversal adaptive filter

estimate $\hat{\xi}(n) = e^2(n)$. Substituting $\hat{\xi}(n) = e^2(n)$ for ξ in the steepest-descent recursion (5.9), of Chapter 5, and replacing the iteration index k by the time index n, we obtain

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \mu\nabla e^2(n) \tag{6.3}$$

where $\mathbf{w}(n) = [w_0(n)\, w_1(n) \cdots w_{N-1}(n)]^{\mathrm{T}}$, μ is the algorithm step-size parameter and ∇ is the gradient operator defined as the column vector

$$\nabla = \left[\frac{\partial}{\partial w_0}\ \frac{\partial}{\partial w_1}\ \cdots\ \frac{\partial}{\partial w_{N-1}}\right]^{\mathrm{T}} \tag{6.4}$$

We note that the ith element of the gradient vector $\nabla e^2(n)$ is

$$\frac{\partial e^2(n)}{\partial w_i} = 2e(n)\frac{\partial e(n)}{\partial w_i} \tag{6.5}$$

Substituting Eq. (6.2) in the last factor on the right-hand side of Eq. (6.5) and noting that $d(n)$ is independent of w_i, we obtain

$$\frac{\partial e^2(n)}{\partial w_i} = -2e(n)\frac{\partial y(n)}{\partial w_i} \tag{6.6}$$

Substituting for $y(n)$ from Eq. (6.1), we get

$$\frac{\partial e^2(n)}{\partial w_i} = -2e(n)x(n-i) \tag{6.7}$$

Using Eqs. (6.4) and (6.7), we obtain

$$\nabla e^2(n) = -2e(n)\mathbf{x}(n) \tag{6.8}$$

Table 6.1 Summary of the LMS algorithm.

Input:	Tap-weight vector, $\mathbf{w}(n)$, Input vector, $\mathbf{x}(n)$, and desired output, $d(n)$
Output:	Filter output, $y(n)$, Tap-weight vector update, $\mathbf{w}(n+1)$

1. Filtering:
$$y(n) = \mathbf{w}^{\mathrm{T}}(n)\mathbf{x}(n)$$

2. Error estimation:
$$e(n) = d(n) - y(n)$$

3. Tap-weight vector adaptation:
$$\mathbf{w}(n+1) = \mathbf{w}(n) + 2\mu e(n)\mathbf{x}(n)$$

where $\mathbf{x}(n) = [x(n)\, x(n-1) \cdots x(n-N+1)]^{\mathrm{T}}$. Substituting this result in Eq. (6.3), we get

$$\mathbf{w}(n+1) = \mathbf{w}(n) + 2\mu e(n)\mathbf{x}(n) \tag{6.9}$$

This is referred to as the *LMS recursion*. It suggests a simple procedure for recursive adaptation of the filter coefficients after arrival of every new input sample, $x(n)$, and its corresponding desired output sample, $d(n)$. Equations (6.1), (6.2), and (6.9), in this order, specify the three steps required to complete each iteration of the LMS algorithm. Equation (6.1) is referred to as *filtering*. It is performed to obtain the filter output. Equation (6.2) is used to calculate the estimation error. Equation (6.9) is tap-weight adaptation recursion. Table 6.1 gives a summary of the LMS algorithm.

The eminent feature of the LMS algorithm, which has made it the most popular adaptive filtering scheme, is its simplicity. Its implementation requires, $2N + 1$ multiplications (N multiplications for calculating the output $y(n)$, one to obtain $(2\mu) \times e(n)$ and N for scalar by vector multiplication $(2\mu e(n)) \times \mathbf{x}(n))$ and $2N$ additions. Another important feature of the LMS algorithm, which is equally important from implementation point of view, is its stable and robust performance against different signal conditions. This aspect of the LMS algorithm will be studied in the later chapters when it is compared with other alternative adaptive filtering algorithms. The major problem of the LMS recursion (6.9) is its slow convergence when the underlying input process is highly colored. This aspect of the LMS algorithm is discussed in the next section and solutions to that will be given in the later chapters.

6.2 Average Tap-Weight Behavior of the LMS Algorithm

Consider the case where the filter input, $x(n)$, and its desired output, $d(n)$, are stationary. In that case, the optimum tap-weight vector, $\mathbf{w}_{\mathrm{o}}$, of the transversal Wiener filter is fixed and can be obtained according to the Wiener–Hopf equation (3.24). Subtracting $\mathbf{w}_{\mathrm{o}}$ from

This may be explained as follows. The tap-weight vector $\mathbf{w}(n)$ at any given time has been affected by the whole past history of the observation data samples $(\mathbf{x}(n-1), d(n-1))$, $(\mathbf{x}(n-2), d(n-2)), \ldots$. When the step-size parameter μ is small, the share of the last N observations in the present value of $\mathbf{w}(n)$ is small, and thus we may say $\mathbf{x}(n)$ and $\mathbf{w}(n)$ are weakly dependent. This clearly leads to Eq. (6.15), with some degree of approximation, if we can assume that the observation samples, which are apart from each other at a distance of N or greater, are weakly dependent. This reasoning seems to be more appealing than the independence assumption. In any case, we use Eq. (6.15) and other similar equations (approximations), which will be introduced later to proceed with our analysis in this book.

Substituting Eq. (6.15) in Eq. (6.14), we obtain

$$E[\mathbf{v}(n+1)] = (\mathbf{I} - 2\mu\mathbf{R})E[\mathbf{v}(n)] \tag{6.16}$$

where $\mathbf{R} = E[\mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n)]$ is the correlation matrix of the input vector $\mathbf{x}(n)$.

Comparing the recursions (6.16) and (5.14), we find that they are of exactly the same mathematical form. The deterministic weight-error vector $\mathbf{v}(k)$ in Eq. (5.14) of the steepest-descent algorithm is replaced by the averaged weight-error vector $E[\mathbf{v}(n)]$ of the LMS algorithm. This suggests that, on average, the LMS algorithm behaves just like the steepest-descent algorithm. In particular, similar to the steepest-descent algorithm, the LMS algorithm is controlled by N modes of convergence, which are characterized by the eigenvalues of the correlation matrix $\mathbf{R}$. Consequently, the convergence behavior of the LMS algorithm is directly linked to the eigenvalue spread of the correlation matrix $\mathbf{R}$. Furthermore, recalling the relationship between the eigenvalue spread of $\mathbf{R}$ and the power spectrum of $x(n)$, we can say that the convergence of the LMS algorithm is directly related to the flatness of the spectral content of the underlying input process.

Following a similar procedure as in Chapter 5, by manipulating Eq. (6.16), one can show that $E[\mathbf{v}(n)]$ converges to zero when μ remains within the range

$$0 < \mu < \frac{1}{\lambda_{\max}} \tag{6.17}$$

where $\lambda_{\max}$ is the maximum eigenvalue of $\mathbf{R}$. However, we should point out here that the above range does not necessarily guarantee the stability of the LMS algorithm. *The convergence of the LMS algorithm requires convergence of the mean of* $\mathbf{w}(n)$ *toward* $\mathbf{w}_{\mathrm{o}}$ *and also convergence of the variance of elements of* $\mathbf{w}(n)$ *to some limited values*. As we shall show later, to guarantee the stability of the LMS algorithm, the latter requirement imposes a much stringent condition on the size of μ. Furthermore, we may note that the independence assumption used to obtain Eq. (6.16) was based on the assumption that μ was very small. The upper limit of μ in Eq. (6.17) may badly violate this assumption. Thus, the validity of Eq. (6.17), even for the convergence of $E[\mathbf{w}(n)]$, is questionable.

Example 6.1

Consider the modeling problem of Example 5.1 which is repeated in Figure 6.2, for convenience. As in Example 5.1, the input signal, $x(n)$, is generated by passing a white noise signal, $v(n)$, through a coloring filter with the system function

$$H(z) = \frac{\sqrt{1-\alpha^2}}{1-\alpha z^{-1}} \tag{6.18}$$

trajectory as the steepest-descent algorithm. The noisy variations of the filter tap weights in the case of LMS algorithm introduce some additional error and push up its learning curve compared to that of the steepest-descent algorithm. However, when the step-size parameter, μ, is small (which is usually the case in practice), one finds that the difference between the two curves is noticeable only when they have converged and approached their steady state. The following example shows this.

Example 6.2

Figure 6.4 shows the learning curves of the LMS algorithm and the steepest-descent algorithm for the modeling problem discussed in Examples 5.1 and 6.1, when $\alpha = 0.75$ and $\mu = 0.01$. For both cases, the filter tap weights have been initialized with $w_0(0) = w_1(0) = 0$. The learning curve of the steepest-descent algorithm has been obtained by inserting the numerical values of the parameters in Eq. (5.31). The learning curve of the LMS algorithm is obtained by an ensemble average of the sequence $e^2(n)$ over 1000 independent runs. We note that the two curves match closely. The learning curve of the LMS algorithm remains slightly above the learning curve of the steepest-descent algorithm. This is because of the use of noisy estimates of the gradient vector in the LMS algorithm.

We shall emphasize that, despite the noisy variation of the filter tap weights, the learning curve of the LMS algorithm matches closely with the theoretical result of the steepest-descent algorithm. In particular, Eq. (5.31) is applicable and the time constant equation

$$\tau = \frac{1}{4\mu\lambda} \tag{6.33}$$

can be used for predicting the transient behavior of the LMS algorithm.

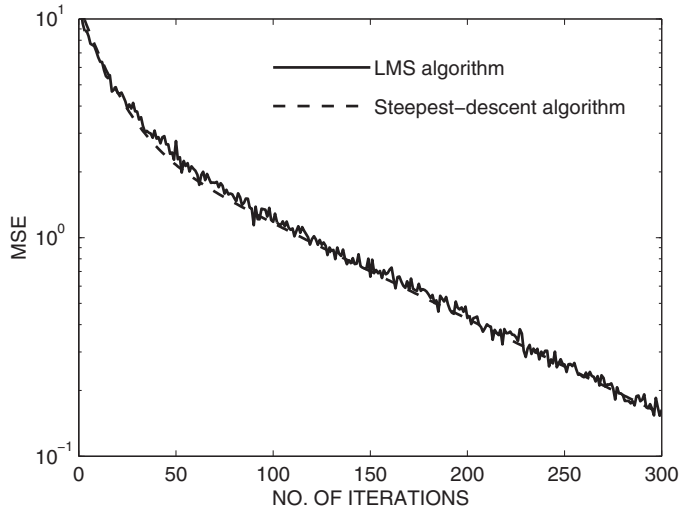

Figure 6.4 Learning curves of the steepest-descent algorithm and LMS algorithm for the modeling problem of Figure 6.2 and the parameter values of $\alpha = 0.75$ and $\mu = 0.01$.

It is useful to simplify this result, by making some appropriate approximations, so that it can conveniently be used for the selection of the step-size parameter, μ. In practice, one usually selects μ, so that a misadjustment of 10% ($\mathcal{M} = 0.1$) or less is achieved. In that case, we may find that

$$\sum_{i=0}^{N-1} \frac{\mu\lambda_i}{1 - 2\mu\lambda_i} \approx \mu \sum_{i=0}^{N-1} \lambda_i = \mu \text{tr}[\mathbf{R}] \tag{6.62}$$

where the last equality is obtained from Eq. (4.24). This approximation is understood if we note that when $\mathcal{M}$ is small, the summation on the left-hand side of Eq. (6.62) is also small. Moreover, when the latter summation is small, $\mu\lambda_i \ll 1$, for $i = 0, 1, \ldots, N-1$, and, thus, these may be deleted from the denominators of the terms under the summation on the right-hand side of Eq. (6.62). Thus, we obtain

$$\mathcal{M} = \frac{\mu \text{tr}[\mathbf{R}]}{1 - \mu \text{tr}[\mathbf{R}]} \tag{6.63}$$

Furthermore, we note that when $\mathcal{M}$ is small, say $\mathcal{M} \leq 0.1$, μtr[**R**] is also small and, thus, it may be ignored in the denominator of Eq. (6.63), to obtain

$$\mathcal{M} \approx \mu \text{tr}[\mathbf{R}] \tag{6.64}$$

This is a very convenient equation, as tr[**R**] is equal to the sum of the powers of the signal samples at the filter tap inputs. This can be easily measured and used for the selection of the step-size parameter, μ, for achieving a certain level of misadjustment. Furthermore, when the input process to the filter is nonstationary, tr[**R**] may be updated recursively and the step-size parameter, μ, chosen accordingly to keep a certain level of misadjustment.

6.3.4 *Stability*

In Chapter 5, we noted that the steepest-descent algorithm remains stable only when its corresponding step-size parameter, μ, takes a value between zero and an upper bound value, which was found to be dependent on the statistics of the filter input. The same is true for the LMS algorithm. However, the use of stochastic gradient in the LMS algorithm makes it more sensitive to the value of its step-size parameter, μ, and, as a result, the upper bound of μ, which can ensure a stable behavior of the LMS algorithm, is much lower than the corresponding bound in the case of the steepest-descent algorithm. To find the upper bound of μ which guarantees the stability of the LMS algorithm, we elaborate on the misadjustment equation (6.61).

We define

$$\mathcal{J} = \sum_{i=0}^{N-1} \frac{\mu\lambda_i}{1 - 2\mu\lambda_i} \tag{6.65}$$

and note that

$$\mathcal{M} = \frac{\mathcal{J}}{1 - \mathcal{J}} \tag{6.66}$$

6.4 Computer Simulations

In the study of adaptive filters, computer simulation plays a major role. In the analysis that was presented in the previous section, we had to consider a number of assumptions to make the problem mathematically tractable. The validity of these assumptions and the matching between mathematical results and the actual performance of adaptive filters are usually verified through computer simulations.

In this section, we present a few examples of computer simulations. We present examples of four different applications of adaptive filters:

- System modeling
- Channel equalization
- Adaptive line enhancement (this is an example of prediction)
- Beamforming.

Our objectives in this presentation are:

1. To help the novice readers to have a fast start in doing computer simulations.
2. To check the accuracy of the developed theoretical results.
3. To enhance the understanding of the theoretical results by careful observation and interpretation of simulation results.

All the results which are given in the following have been generated by using the MATLAB numerical package. The MATLAB programs used to generate the results presented in this section and other parts of this book are available on an accompanying website. A list of these programs (m-files as they are called in MATLAB) is given at the end of the book and also in the `read.me` file on the accompanying website. We encourage all the novice readers to try to run these programs, as this, we believe, is essential for a better understanding of the adaptive filtering concepts.

6.4.1 System Modeling

Consider a system modeling problem, as depicted in Figure 6.5. The filter input is obtained by passing a unit variance white Gaussian sequence, $v(n)$, through a filter with the system function $H(z)$. The plant, $W_o(z)$, is assumed to be a FIR system with the impulse response duration of N samples. The plant output is contaminated with an additive white Gaussian noise sequence, $e_o(n)$, with variance σ_o^2. An N-tap adaptive filter, $W(z)$, is used to estimate the plant parameters.

For simulations, in this section, we select $N = 15$, $\sigma_o^2 = 0.001$ and

$$W_o(z) = \sum_{i=0}^{7} z^{-i} - \sum_{i=8}^{14} z^{-i} \tag{6.79}$$

We present results of simulations for two choices of input, which are characterized by

$$H(z) = H_1(z) = 0.35 + z^{-1} - 0.35z^{-2} \tag{6.80}$$

and $\Phi_{\nu\nu}(e^{j\omega}) = 1$, since $\nu(n)$ is a unit variance white noise process. The fact that $H_2(z)$ generates a process that is highly colored, while the process generated by $H_1(z)$ is relatively flat, is clearly seen.

Figure 6.7a and b shows the learning curves of the LMS algorithm for the two choices of $H(z)$. The step-size parameter, μ, is selected according to the simplified misadjustment equation (6.64) for the misadjustment values 10%, 20%, and 30%. The filter tap weights are initialized to zero. Each plot is obtained by an ensemble average of 100 independent simulation runs. We note that $\xi_{\min} = \sigma_o^2 = 0.001$, and this is achieved when the model and plant coefficients match. Careful examination of the results presented in Figure 6.7a and b reveals that the predictions made by Eq. (6.64) are accurate for the cases where μ is set for a misadjustment of 10% (or less). For larger values of μ, one finds that a more accurate theoretical estimate of the misadjustment is obtained using Eq. (6.61). Such estimate, of course, requires calculation of the eigenvalues of the correlation matrix $\mathbf{R}$. The MATLAB program "`modeling.m`" on the accompanying website contains instructions, which generate matrix $\mathbf{R}$ and the other parameters required for these calculations. The reader is encouraged to use this program and experiment with that to examine the effect of various parameters, such as the step-size, μ, the plant model order, and the input sequence to the adaptive filter. Such experiments will greatly enhance the reader's understanding of the concepts of convergence and misadjustment.

Experiments with the LMS algorithm show that the accuracy of the misadjustment equations developed above varies with the statistics of the filter input and the step-size parameter. For example, one finds that all of the three plots in Figure 6.7a and two of the plots in Figure 6.7b match the theoretical predictions made by Eq. (6.61), but the third plot in Figure 6.7b (i.e., the case $\mathcal{M} = 30\%$) does not match Eq. (6.61). In the latter case, the LMS algorithm experiences some instability problem. The mismatch between the theory and experiments here is attributed to the fact that the independence assumption made in the development of the theoretical results is badly violated for larger values of μ.

6.4.2 *Channel Equalization*

Figure 6.8 depicts a channel equalization problem. The input sequence to the channel is assumed to be binary (taking values of $+1$ and -1) and white. The channel system function is denoted by $H(z)$. The channel noise, $\nu_c(n)$, is modeled as an additive white Gaussian process with variance $\sigma_{\nu_c}^2$. The equalizer is implemented as an N-tap transversal filter. The desired output of the equalizer is assumed to be $s(n - \Delta)$, that is, a delayed replica of the transmitted data symbols. For the training of the equalizer, it is assumed that the transmitted data symbols are available at the receiver. This is called *training mode*. Once the equalizer is trained and switched to the data mode, its output, after passing through a slicer, gives the transmitted symbols. A discussion on the training and data modes of equalizers can be found in Chapter 1. More detailed explanations and adaptation algorithms are presented in Chapter 17.

Two choices of the channel response, $H(z)$, are considered for our study, here. These are purposefully selected to be the same as the two choices of $H(z)$ in the modeling problem, above, where $H(z)$ was used to shape the power spectral density of the input process to the plant and model. This facilitates a comparison of the results in the two

is minimized in magnitude. Substituting Eq. (6.103) in Eq. (6.104) and rearranging, we obtain

$$e^+(n) = (1 - 2\mu(n)\mathbf{x}^{\mathrm{T}}(n)\mathbf{x}(n))e(n) \tag{6.105}$$

Minimizing $(e^+(n))^2$ with respect to $\mu(n)$ results in the following:

$$\mu(n) = \frac{1}{2\mathbf{x}^{\mathrm{T}}(n)\mathbf{x}(n)} \tag{6.106}$$

which forces $e^+(n)$ to zero. Substituting Eq. (6.106) in Eq. (6.103), we obtain

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \frac{1}{\mathbf{x}^{\mathrm{T}}(n)\mathbf{x}(n)} e(n)\mathbf{x}(n) \tag{6.107}$$

This is the NLMS recursion. When this is combined with the filtering equation (6.1) and the error estimation equation (6.2), we obtain the NLMS algorithm.

There have been a variety of interpretations to the NLMS algorithm. We review some of these in the following as it can help in enhancing our understanding of this algorithm.

1. The use of $\mu(n)$ as in Eq. (6.106) is appealing as it selects a step-size parameter proportional to the inverse of the instantaneous signal energy at the adaptive filter input. This matches the misadjustment equation (6.64), which suggests that the step-size parameter of the LMS algorithm should be selected proportional to the inverse of the average total energy at the filter tap inputs. Note that

$$\mathrm{tr}[\mathbf{R}] = \sum_{i=0}^{N-1} E[x^2(n-i)] = E\left[\sum_{i=0}^{N-1} x^2(n-i)\right]$$

   and $\sum_{i=0}^{N-1} x^2(n-i)$ is the total instantaneous signal energy at the filter tap inputs.
2. The NLMS recursion (6.107) is equivalent to running the LMS recursion for every new sample of input many iterations until it converges (Nitzberg, 1985); see Problem P6.17.
3. The NLMS recursion may also be derived by solving the following constrained optimization problem (Goodwin and Sin, 1984):

   Given the tap-input vector $\mathbf{x}(n)$ and the desired output sample $d(n)$, choose the updated tap-weight vector $\mathbf{w}(n+1)$ so as to minimize the squared Euclidean norm of the difference

$$\boldsymbol{\eta}(n) = \mathbf{w}(n+1) - \mathbf{w}(n) \tag{6.108}$$

   subject to the constraint

$$\mathbf{w}^{\mathrm{T}}(n+1)\mathbf{x}(n) = d(n) \tag{6.109}$$

   Observe that the solution given by Eq. (6.107) satisfies the constraint (6.109). Hence, define $\boldsymbol{\eta}_{\mathrm{NLMS}}(n)$ as

$$\boldsymbol{\eta}_{\mathrm{NLMS}}(n) = \mathbf{w}(n+1) - \mathbf{w}(n) = \frac{1}{\mathbf{x}^{\mathrm{T}}(n)\mathbf{x}(n)} e(n)\mathbf{x}(n) \tag{6.110}$$

   We will now show that $\boldsymbol{\eta}_{\mathrm{NLMS}}(n)$ is indeed the solution to the problem posed above.

For $M = 1$, the APLMS algorithm reduces to the NLMS algorithm. However, the APLMS algorithm offers a significant convergence improvement over the NLMS algorithm as M increases. Clearly, this improvement is at a cost of additional computational complexity. A comparison of the number of operations in Tables 6.2 and 6.3 reveal that the APLMS algorithm is at least M times more complex than the NLMS algorithm. This does not include the computation and inversion of the $M \times M$ matrix $\mathbf{X}^{\mathrm{T}}(n)\mathbf{X}(n) + \psi\mathbf{I}$. Nevertheless, further studies reveal that the tap-weight vector adaptation (6.133) may be executed once after every M sample, without any significant loss in the convergence behavior, hence, brings down its complexity to a level comparable to that of the NLMS algorithm. A summary of various versions of the APLMS algorithm can be found in Morgan and Kratzer (1996).

A convergence analysis of the APLMS algorithm is presented in Shin and Sayed (2004). The following observations are made from the results presented in this work. (i) The convergence rate of APLMS algorithm improves as M increases. (ii) The misadjustment of APLMS algorithm, on the other hand, increases (i.e., degrades) as M increases. Here, without getting involved into the details of mathematical derivations, we make an attempt to explain these observations through some intuitions.

Using the matrix inversion lemma formula (6.60), one can show that

$$\mathbf{X}(n)(\mathbf{X}^{\mathrm{T}}(n)\mathbf{X}(n) + \psi\mathbf{I})^{-1} = (\mathbf{X}(n)\mathbf{X}^{\mathrm{T}}(n) + \psi\mathbf{I})^{-1}\mathbf{X}(n) \tag{6.134}$$

Substituting Eq. (6.134) in Eq. (6.133), we obtain

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu(\mathbf{X}(n)\mathbf{X}^{\mathrm{T}}(n) + \psi\mathbf{I})^{-1}\mathbf{X}(n)\mathbf{e}(n) \tag{6.135}$$

Examining Eq. (6.135), one finds that $\mathbf{X}(n)\mathbf{e}(n) = \sum_{i=0}^{M-1}\mathbf{x}(n-i)e(n-i)$ and this in turn implies that $\mathbf{X}(n)\mathbf{e}(n)$ is a random vector whose mean is equal to $-\frac{M}{2}\nabla_{\mathbf{w}}\xi$. Also, as M increases, the mean of the $N \times N$ matrix $\mathbf{X}(n)\mathbf{X}^{\mathrm{T}}(n)$ approaches $M\mathbf{R}$. Hence, one may think of $(\mathbf{X}(n)\mathbf{X}^{\mathrm{T}}(n) + \psi\mathbf{I})^{-1}\mathbf{X}(n)\mathbf{e}(n)$ as a noisy (and regularized) sample of $-\frac{1}{2}\mathbf{R}^{-1}\nabla_{\mathbf{w}}\xi$, and, thus, argue that the update equation of APLMS algorithm attempts to implement a stochastic version of the Newton's method. In other words, one may argue that the premultiplication of the stochastic gradient vector $\mathbf{X}(n)\mathbf{e}(n)$ by $(\mathbf{X}(n)\mathbf{X}^{\mathrm{T}}(n) + \psi\mathbf{I})^{-1}$ results in a vector that, on average, points toward the minimum of the performance surface of the adaptive filter, hence, avoiding the slow modes of convergence of the LMS algorithm.

To explain why the misadjustment of APLMS algorithm increases with M, we resort to a generalization of the geometrical interpretation of the NLMS algorithm that was presented earlier in Figure 6.15. Figure 6.16 presents a diagram that expands Figure 6.15 to the case where $M = 2$. Here, to satisfy the pair of constraints $\mathbf{w}^{\mathrm{T}}\mathbf{x}(n) = d(n)$ and $\mathbf{w}^{\mathrm{T}}\mathbf{x}(n-1) = d(n-1)$, while minimizing $\|\boldsymbol{\eta}_{\mathrm{APLMS}}(n)\|$, the error vector $\boldsymbol{\eta}_{\mathrm{APLMS}}(n)$ must be orthogonal to the intersection of the subspaces of the two constraints. Hence, one may note that $\boldsymbol{\eta}_{\mathrm{APLMS}}(n)$ is not necessarily orthogonal to the subspace of the constraint $\mathbf{w}^{\mathrm{T}}\mathbf{x}(n) = d(n)$ and thus $\|\boldsymbol{\eta}_{\mathrm{APLMS}}(n)\| \geq \|\boldsymbol{\eta}_{\mathrm{NLMS}}(n)\|$. Obviously, $\|\boldsymbol{\eta}_{\mathrm{APLMS}}(n)\|$ increases further as M is given larger values. On the other hand, we recall that $\boldsymbol{\eta}_{\mathrm{NLMS}}(n)$ and, similarly, $\boldsymbol{\eta}_{\mathrm{APLMS}}(n)$ may be thought as a perturbation that may be imposed on the filter tap weights as the NLMS and APLMS algorithms proceed. In the steady state, a larger perturbation of the tap weights, clearly, results in a larger misadjustment. A few problems

Table 6.4 Summary of an implementation of variable step-size LMS algorithm.

Input:	Tap-weight vector, $\mathbf{w}(n)$, input vector, $\mathbf{x}(n)$, Gradient terms $g_0(n-1), g_1(n-1), \ldots, g_{N-1}(n-1)$, Step-size parameters, $\mu_0(n-1), \mu_1(n-1), \ldots, \mu_{N-1}(n-1)$, and desired output, $d(n)$
Output:	Filter output, $y(n)$, tap-weight vector update, $\mathbf{w}(n+1)$, gradient terms $g_0(n), g_1(n), \ldots, g_{N-1}(n)$, and updated step-size parameters $\mu_0(n), \mu_1(n), \ldots, \mu_{N-1}(n)$

1. Filtering:
$$y(n) = \mathbf{w}^{\mathrm{T}}(n)\mathbf{x}(n)$$
2. Error estimation: $e(n) = d(n) - y(n)$
3. Tap weights and step-size parameters adaptation:

For $i = 0, 1, \ldots, N-1$

$g_i(n) = e(n)x(n-i)$

$\mu_i(n) = \mu_i(n-1) + \rho \text{sign}[g_i(n)]\text{sign}[g_i(n-1)]$

if $\mu_i(n) > \mu_{\max}$, $\mu_i(n) = \mu_{\max}$

if $\mu_i(n) < \mu_{\min}$, $\mu_i(n) = \mu_{\min}$

$w_i(n+1) = w_i(n) + 2\mu_i(n)g_i(n)$

end

This leads to the inequality[6]

$$\text{tr}[\boldsymbol{\mu}(n)\mathbf{R}] < \frac{1}{3} \tag{6.141}$$

as a sufficient condition, which ensures the stability of the VSLMS algorithm. Although the inequality (6.141) may be used to impose some bounds on the step-size parameters $\mu_i(n)$'s dynamically as the adaptation of the filter proceeds, this leads to a rather complicated process. Instead, in practice, one usually prefers to use Eq. (6.74) to limit all $\mu_i(n)$'s to the same maximum value, say $\mu_{\max}$.

The minimum bound, which may be imposed on the variable step-size parameters, $\mu_i(n)$'s, can be as low as zero. However, in actual practice, a positive bound is usually used, so that the adaptation process will be on all the time and possible variations in the adaptive filter optimum tap weights can always be tracked. Here, we use the notation $\mu_{\min}$ to refer to this lower bound. Table 6.4 gives the summary of an implementation of the VSLMS algorithm.

6.9 LMS Algorithm for Complex-Valued Signals

In applications such as data transmission with quadrature-amplitude modulation (QAM) signaling and beamforming with baseband processing of signals, the underlying data

[6] See Chapter 14 for a formal derivation of Eq. (6.141).

Note that for all cases, the eigenvalues are normalized such that $\sum_i \lambda_i = \text{tr}[\mathbf{R}] = 10$. This implies that the tight stability bound Eq. (6.74) for all cases is the same. The goal of this problem is to show the impact of distribution of eigenvalues of the correlation matrix of an adaptive on the variation of the true stability bound of the LMS algorithm, that is, the range of μ that guarantees a stable LMS algorithm.

(i) Make a plot of $\mathcal{J}$ (as defined in Eq. (6.65)) for each case when μ varies from 0 to 0.1.
(ii) Find the range of μ in each case, which results in a stable LMS algorithm.
(iii) Discuss the various ranges that you have obtained in (ii) and compare them with the tight-bound $1/(3\text{tr}[\mathbf{R}])$ and a softer bound that may be defined as $1/\text{tr}[\mathbf{R}]$. Discuss in which cases one is closer to the former bound or to the latter bound.

P6.6 Equations (6.84) and (6.87) provide approximate expressions for expected learning curves of the LMS algorithm in the two cases of system modeling and channel equalization. For the five cases noted in Problem P6.5, plot the expected learning curves of the LMS algorithm for system modeling and channel equalization and discuss your observation.

P6.7 The input process to a system modeling problem, using a 10-tap FIR adaptive filter, has the power spectral density shown in Figure P6.7. Assume that the MSE at zeroth iteration is equal to 1 and $\xi_{\min} = 0.0001$, and the step-size parameter μ has been chosen for a 10% misadjustment. Present a typical learning curve of an LMS algorithm in this setup. Indicate the time constants of the various modes of convergence on the presented curve and the mean squared error (MSE) that the LMS algorithm converges to.

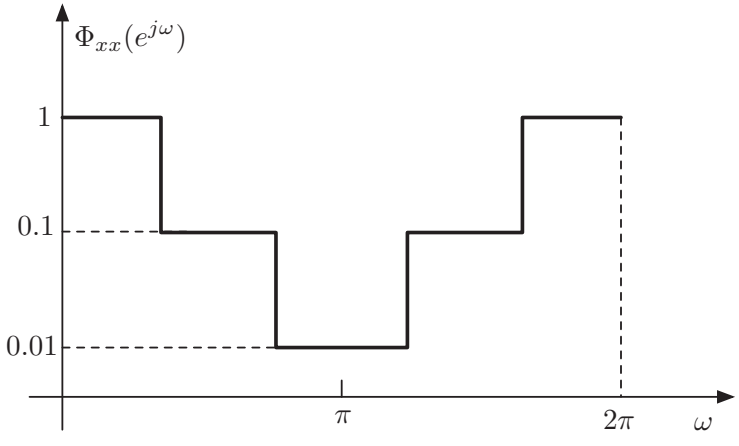

Figure P6.7

P6.8 Consider a channel equalization problem similar to the one depicted in Figure 6.8. The magnitude response of the channel, $|H(z)|$, is as shown in Figure P6.8.

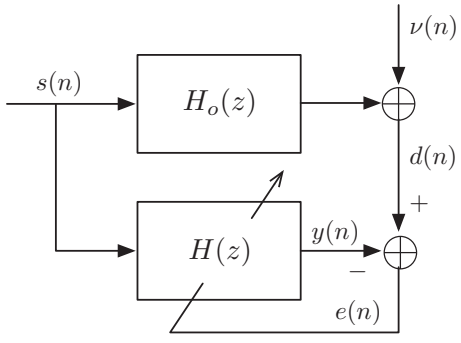

Figure P6.12

(iii) Use the result of Part (ii) to find the range of μ, which guarantees the convergence of $\|\mathbf{g}(n)\|^2$. Does this also guarantees the convergence of the LMS algorithm?

(iv) Compare the range obtained in Part (iii) with the range of μ given in Eq. (6.74).

P6.13 In this problem, we discuss the effect of the power level of the input process to an adaptive filter and its variation on the convergence of the LMS algorithm.

(i) Consider the LMS recursion (6.9) and assume that the time constants of its different modes of convergence are $\tau_0, \tau_1, \ldots, \tau_{N-1}$. Keep μ fixed, replace $\mathbf{x}(n)$ by $\mathbf{x}'(n) = \alpha\mathbf{x}(n)$, where α is a constant, and obtain the corresponding time constants of the resulting recursion, in terms of τ_i's, under the condition that the step-size parameter μ is small enough to guarantee the convergence of the algorithm.

(ii) Under the condition that the power levels of the elements of $\mathbf{x}(n)$ are time varying and fluctuate slowly between high and low levels, what is the shortcoming of the LMS algorithm (discuss)? Can you suggest any solution to this?

P6.14 This problem attempts to show the validity of the approximation (6.86) in a nonrigorous manner.

Consider a random process $x(n)$ and its associated $(2M+1)$-by-$(2M+1)$ correlation matrix $\mathbf{R}$. Let

$$\mathbf{q}_i = [q_{i,-M} \cdots q_0 \cdots q_{i,M}]^{\mathrm{T}}, \qquad \text{with} \quad \mathbf{q}_i^{\mathrm{H}}\mathbf{q}_i = 1$$

be the ith eigenvector of $\mathbf{R}$ and λ_i be its corresponding eigenvalue.

(i) Show that the expansion of the relationship $\mathbf{R}\mathbf{q}_i = \lambda_i\mathbf{q}_i$ leads to

$$\sum_{k=-M}^{M} \phi_{xx}(k-l)q_{i,k} = \lambda_i q_{i,l}, \qquad \text{for } -M \le l \le M \qquad \text{(P6.11.1)}$$

where $\phi_{xx}(k-l)$ is the autocorrelation function of $x(n)$ for lag $k-l$.

(i) Define $\bar{\mathbf{w}}(n) = E[\mathbf{w}(n)]$, where $E[\cdot]$ denotes statistical expectation, and use the *independence assumption* of Section 6.3 to show that the following recursive equation holds:

$$\bar{\mathbf{w}}(n+1) = (\mathbf{I} - 2\mu\mathbf{R}')\bar{\mathbf{w}}(n) + 2\mu\mathbf{p}$$

Specify $\mathbf{R}'$ and $\mathbf{p}$ and obtain the time constants of the learning curve of the leaky LMS algorithm in terms of the eigenvalues of the correlation matrix $\mathbf{R} = E\{\mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n)\}$ and the parameters β and μ.

(ii) Assuming that the step-size parameter μ is small enough to guarantee the convergence of the leaky LMS algorithm, derive an equation for $\bar{\mathbf{w}}(\infty)$ in terms of $\mathbf{R}'$ and $\mathbf{p}$.

(iii) Show that the difference between $\bar{\mathbf{w}}(\infty)$ and the optimum tap-weight vector of the adaptive filter is given by the following equation.

$$\bar{\mathbf{w}}(\infty) - \mathbf{w}_{\mathrm{o}} = -\gamma \sum_{i=0}^{N-1} \frac{\mathbf{q}_i^{\mathrm{T}}\mathbf{p}}{\lambda_i(\lambda_i + \gamma)}\mathbf{q}_i$$

where $\gamma = \frac{1-\beta}{2\mu}$, and λ_i's and $\mathbf{q}_i$'s are the eigenvalues and eigenvectors of $\mathbf{R}$, respectively.

P6.26 Define the scalar quantity $\|\mathbf{v}(n)\|^2 = E[\mathbf{v}^{\mathrm{T}}(n)\mathbf{v}(n)]$ as the misalignment of an adaptive filter tap weights.

(i) Show that

$$\|\mathbf{v}(n)\|^2 = \mathrm{tr}[\mathbf{K}'(n)]$$

where the correlation matrix $\mathbf{K}'(n)$ is defined as in Eq. (6.31).

(ii) Use Eq. (6.52) to show that

$$\|\mathbf{v}(\infty)\|^2 = \frac{\sum_{i=0}^{N-1} \mu/(1-2\mu\lambda_i)}{1 - \sum_{i=0}^{N-1} \mu\lambda_i/(1-2\mu\lambda_i)}$$

(iii) Show that when μ is small, the above result reduces to

$$\|\mathbf{v}(\infty)\|^2 = \mu N$$

P6.27 A complex-valued random process $x(n) = u(n) + v(n)$ is available. The process $u(n) = a\mathrm{e}^{j(\omega n+\phi)}$, where a and ϕ are random, but fixed for every realization of $x(n)$. The process $v(n)$ is a complex-valued noise which may not be white. Assuming that the frequency, ω, of $u(n)$ is known, propose an adaptive filter and its associated adaptation algorithm to filter out $v(n)$ from $x(n)$ and enhance $u(n)$ in the minimum MSE sense, preserving its phase, ϕ, and its amplitude, a.

P6.28 Repeat Problem P6.27 when $u(n) = a\cos(\omega n + \phi)$ and $v(n)$ is a real-valued noise sequence.

where α is a constant in the range of -1 to $+1$. For $\alpha = 0.9$, we obtain

$$\mathbf{R} = \begin{bmatrix} 1.0000 & 0.9000 & 0.8100 & 0.7290 \\ 0.9000 & 1.0000 & 0.9000 & 0.8100 \\ 0.8100 & 0.9000 & 1.0000 & 0.9000 \\ 0.7290 & 0.8100 & 0.9000 & 1.0000 \end{bmatrix} \tag{7.18}$$

For a derivation of $\mathbf{R}$, see Example 4.1. Using the DCT as the transformation, we get

$$\mathbf{R}_{\mathcal{T}} = \begin{bmatrix} 3.5245 & 0.0000 & -0.0855 & 0.0000 \\ 0.0000 & 0.3096 & 0.0000 & -0.0032 \\ -0.0855 & 0.0000 & 0.1045 & 0.0000 \\ 0.0000 & -0.0032 & 0.0000 & 0.0614 \end{bmatrix} \tag{7.19}$$

This clearly is much closer to diagonal (i.e., its off-diagonal elements are relatively closer to zero) when compared to $\mathbf{R}$.

The normalization performed in the implementation of the LMS algorithm in transform domain (as we shall see later), in effect, is equivalent to normalization of the elements of $\mathbf{x}_{\mathcal{T}}(n)$ to the power of unity. This is done by premultiplying $\mathbf{x}_{\mathcal{T}}(n)$ with a diagonal matrix, $\mathbf{D}^{-1/2}$, before the filtering and adaptation process, where $\mathbf{D}^{-1/2}$ is the inverse of the square root of the diagonal matrix

$$\mathbf{D} = \begin{bmatrix} E[x^2_{\mathcal{T},0}(n)] & 0 & \cdots & 0 \\ 0 & E[x^2_{\mathcal{T},1}(n)] & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & E[x^2_{\mathcal{T},N-1}(n)] \end{bmatrix} \tag{7.20}$$

Thus, we get

$$\mathbf{x}^{\mathrm{n}}_{\mathcal{T}}(n) = \mathbf{D}^{-1/2}\mathbf{x}_{\mathcal{T}}(n) \tag{7.21}$$

where $\mathbf{x}^{\mathrm{n}}_{\mathcal{T}}(n)$ is the normalized tap-input vector. The correlation matrix associated with $\mathbf{x}^{\mathrm{n}}_{\mathcal{T}}(n)$ is

$$\mathbf{R}^{\mathrm{n}}_{\mathcal{T}} = \mathbf{D}^{-1/2}\mathbf{R}_{\mathcal{T}}\mathbf{D}^{-1/2} \tag{7.22}$$

Furthermore, we note that

$$\mathbf{D} = \mathrm{diag}[\mathbf{R}_{\mathcal{T}}] \tag{7.23}$$

where $\mathrm{diag}[\mathbf{R}_{\mathcal{T}}]$ denotes the diagonal matrix consisting of the diagonal elements of $\mathbf{R}_{\mathcal{T}}$. The reader may easily verify that the mean-squared values of the elements of $\mathbf{x}^{\mathrm{n}}_{\mathcal{T}}(n)$ as well as the diagonal elements of $\mathbf{R}^{\mathrm{n}}_{\mathcal{T}}$ are all equal to unity as a result of this normalization.

For the above example, we get

$$\mathbf{D}^{-1/2} = \begin{bmatrix} 0.5327 & 0 & 0 & 0 \\ 0 & 1.7972 & 0 & 0 \\ 0 & 0 & 3.0934 & 0 \\ 0 & 0 & 0 & 4.0357 \end{bmatrix} \tag{7.24}$$

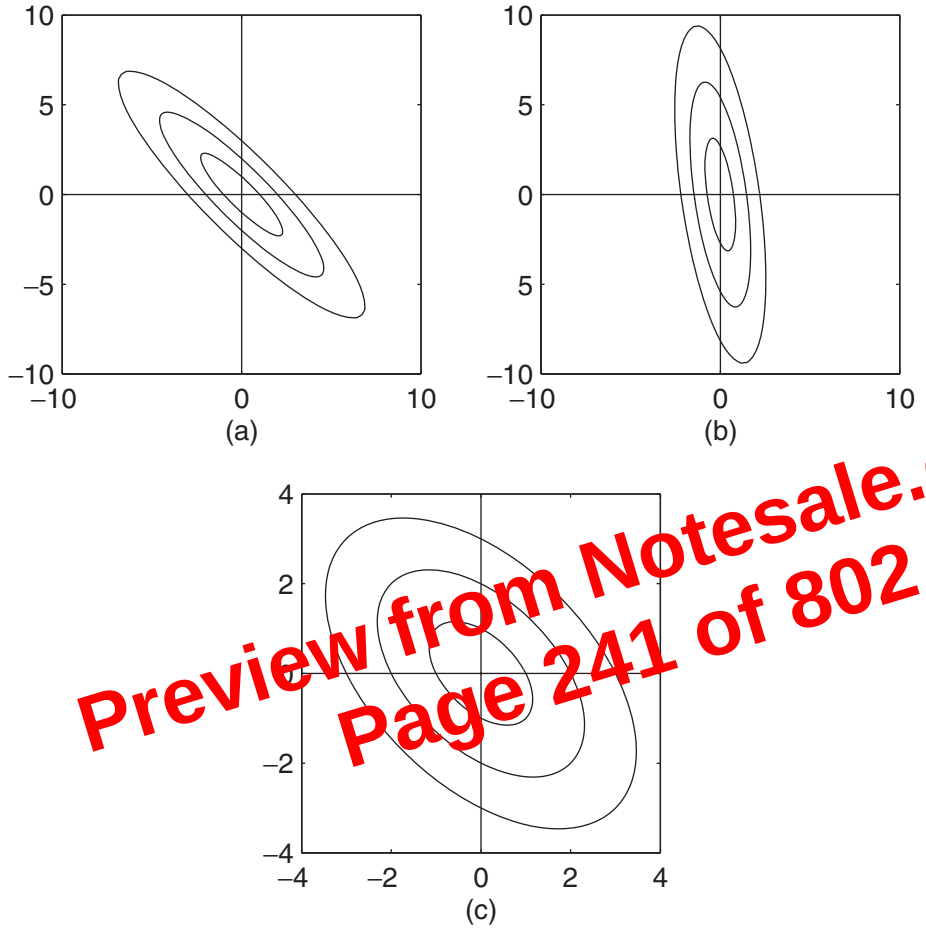

Figure 7.3 A geometrical interpretation of the TDLMS algorithm. (a) Performance surface before transformation; (b) performance surface after transformation, but without normalization; and (c) performance surface after transformation and normalization (adopted from Marshall *et al.* (1989)).

Figure 7.3a shows the contour plot associated with this performance surface. As we may recall from our discussions in the previous chapters, the eccentricity of the contour ellipses in Figure 7.3a, is related to the eigenvalue spread of the correlation matrix $\mathbf{R}$. A large eccentricity is due to a large eigenvalue spread and that, in turn, results in certain slow mode(s) of convergence when the conventional LMS algorithm is used to adjust the filter tap weights.

Application of an orthogonal transform, $\mathcal{T}$, converts the tap-input vector $\mathbf{x}(n)$ to $\mathbf{x}_{\mathcal{T}}(n) = \mathcal{T}\mathbf{x}(n)$, whose associated correlation matrix, $\mathbf{R}_{\mathcal{T}}$, is related to $\mathbf{R}$ according to Eq. (7.9). As a numerical example, let us choose

$$\mathcal{T} = \begin{bmatrix} 0.8 & 0.6 \\ -0.6 & 0.8 \end{bmatrix} \tag{7.38}$$

This, with $\mathbf{R}$ as given in Eq. (7.36), results in

$$\mathbf{R}_{\mathcal{T}} = \begin{bmatrix} 1.864 & 0.252 \\ 0.252 & 0.136 \end{bmatrix} \tag{7.39}$$

If no normalization is applied to the transformed samples, the performance surface associated with the TDAF will be

$$\xi_{\mathcal{T}}(\mathbf{v}_{\mathcal{T}}) = \xi_{\min} + \mathbf{v}_{\mathcal{T}}^{T}\mathbf{R}_{\mathcal{T}}\mathbf{v}_{\mathcal{T}} \tag{7.40}$$

which for the present numerical example can be expanded as

$$\xi_{\mathcal{T}}(v_{\mathcal{T},0}, v_{\mathcal{T},1}) = \xi_{\min} + 1.864v_{\mathcal{T},0}^2 + 0.136v_{\mathcal{T},1}^2 + 0.504v_{\mathcal{T},0}v_{\mathcal{T},1} \tag{7.41}$$

Figure 7.3b shows the contours associated with the performance surface defined by Eq. (7.41). Note that the effect of the transformation is only to rotate the performance surface with respect to the coordinate axes. The shape of the performance surface, that is, the eccentricity of the contour ellipses, has not changed. This can be mathematically explained by noting that, as $\boldsymbol{\mathcal{T}}\boldsymbol{\mathcal{T}}^{\mathrm{T}} = \boldsymbol{\mathcal{T}}^{\mathrm{T}}\boldsymbol{\mathcal{T}} = \mathbf{I}$,

$$\begin{aligned} \xi(\mathbf{v}) &= \xi_{\min} + \mathbf{v}^{\mathrm{T}}\mathbf{R}\mathbf{v} \\ &= \xi_{\min} + \mathbf{v}^{\mathrm{T}}\boldsymbol{\mathcal{T}}^{\mathrm{T}}\boldsymbol{\mathcal{T}}\mathbf{R}\boldsymbol{\mathcal{T}}^{\mathrm{T}}\boldsymbol{\mathcal{T}}\mathbf{v} \\ &= \xi_{\min} + \mathbf{v}_{\mathcal{T}}^{\mathrm{T}}\mathbf{R}_{\mathcal{T}}\mathbf{v}_{\mathcal{T}} = \xi_{\mathcal{T}}(\mathbf{v}_{\mathcal{T}}) \end{aligned} \tag{7.42}$$

where $\mathbf{v}$ and $\mathbf{v}_{\mathcal{T}}$ are related according to the equation $\mathbf{v}_{\mathcal{T}} = \boldsymbol{\mathcal{T}}\mathbf{v}$. This result, which can also be written as $\xi_{\mathcal{T}}(\mathbf{v}_{\mathcal{T}}) = \xi(\boldsymbol{\mathcal{T}}^{\mathrm{T}}\mathbf{v}_{\mathcal{T}})$, shows that the performance surface defined by Eq. (7.40) is obtained from the one defined by Eq. (7.35) by a rotation of the coordinate axes according to the relationship $\mathbf{v}_{\mathcal{T}} = \boldsymbol{\mathcal{T}}\mathbf{v}$, or, equivalently, by keeping the coordinate axes fixed and rotating the performance surface in the opposite direction. This observation shows that *transformation without normalization has no effect on the convergence behavior of the steepest-descent method and, thus, the LMS algorithm*. Thus, we emphasize that normalization has to be considered as an integrated part of any transform domain adaptive algorithm (as introduced in the case of TDLMS algorithm, in Section 7.4), otherwise transformation adds up to the filter complexity without any gain in convergence.

When the elements of $\mathbf{x}_{\mathcal{T}}(n)$ are normalized to the power of unity[4], the corresponding correlation matrix is given by Eq. (7.22) and its associated performance surface is defined as

$$\xi_{\mathcal{T}}^{\mathrm{n}}(\mathbf{v}_{\mathcal{T}}^{\mathrm{n}}) = \xi_{\min} + \mathbf{v}_{\mathcal{T}}^{\mathrm{n}T}\mathbf{R}_{\mathcal{T}}^{\mathrm{n}}\mathbf{v}_{\mathcal{T}}^{\mathrm{n}} \tag{7.43}$$

For the present example, we obtain

$$R_{\mathcal{T}}^{\mathrm{n}} = \begin{bmatrix} 1.0000 & 0.5005 \\ 0.5005 & 1.0000 \end{bmatrix} \tag{7.44}$$

and

$$\xi_{\mathcal{T}}^{\mathrm{n}}(v_{\mathcal{T},0}^{\mathrm{n}}, v_{\mathcal{T},1}^{\mathrm{n}}) = \xi_{\min} + (v_{\mathcal{T},0}^{\mathrm{n}})^2 + (v_{\mathcal{T},1}^{\mathrm{n}})^2 + 1.001v_{\mathcal{T},0}^{\mathrm{n}}v_{\mathcal{T},1}^{\mathrm{n}} \tag{7.45}$$

[4] We recall that the step-normalization, as applied in Eq. (7.26), and normalization of the elements of $\mathbf{x}_{\mathcal{T}}(n)$ to the power of unity are equivalent.

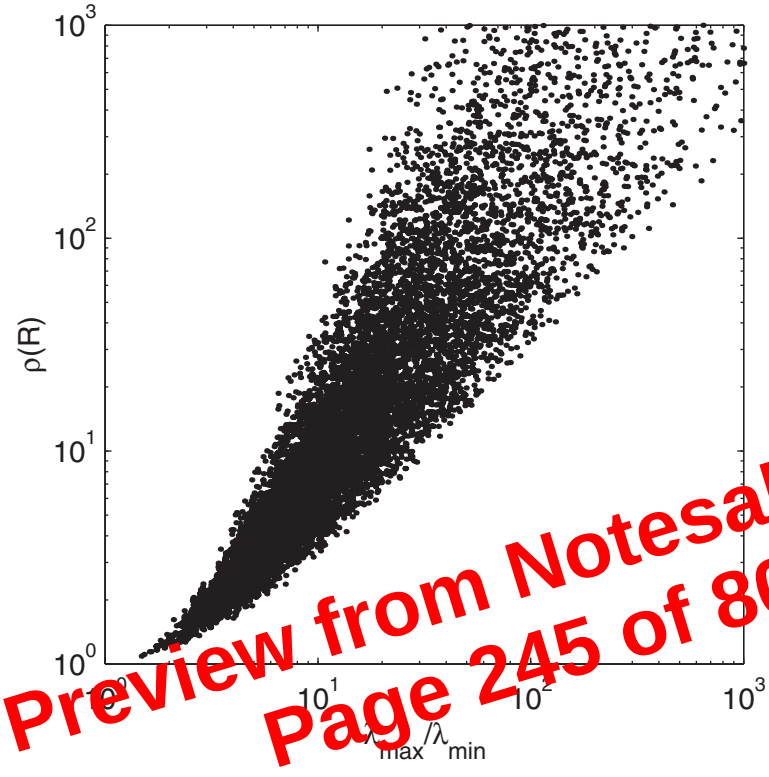

Figure 7.4 Variation of $\rho(\mathbf{R})$ versus eigenvalue spread of $\mathbf{R}$.

where det[$\mathbf{R}$] is the determinant of $\mathbf{R}$, we obtain

$$\rho(\mathbf{R}) = \frac{(\text{tr}[\mathbf{R}]/N)^N}{\det[\mathbf{R}]} \tag{7.50}$$

Now, one may appreciate the index $\rho(\mathbf{R})$ because of its closed-form nature in terms of elements of $\mathbf{R}$.

Before we proceed with the application of the performance index $\rho(\mathbf{R})$ to further study the TDLMS algorithm, we may remark that the relationship between $\rho(\mathbf{R})$ and the eigenvalue spread of $\mathbf{R}$, that is, $\lambda_{max}/\lambda_{min}$, is rather complicated. The index $\rho(\mathbf{R})$ depends on not only $\lambda_{max}/\lambda_{min}$, but also the distribution of the rest of eigenvalues of $\mathbf{R}$ in the range λ_{min} to λ_{max}. However, the general trend is that a large eigenvalue spread of $\mathbf{R}$ implies a large $\rho(\mathbf{R})$ and vice versa. Similarly, a $\rho(\mathbf{R})$ close to 1 implies that the eigenvalue spread of $\mathbf{R}$ is small. Figure 7.4 shows how $\rho(\mathbf{R})$ varies as a function of $\lambda_{max}/\lambda_{min}$ when $N = 10$ and the eigenvalues of $\mathbf{R}$ are assumed to be a set of random numbers distributed in the range 0 to 1.

7.6.3 Improvement Factor and Comparisons

To compare a pair of LMS-based algorithms, say LMS_1 and LMS_2, we define an improvement factor, I_ρ, as the natural logarithm of the ratio of the performance index $\rho(\cdot)$ in the

$$= \frac{\det[\text{diag}[\mathbf{R}_{\mathcal{T}}]]}{(\text{tr}[\text{diag}[\mathbf{R}_{\mathcal{T}}]]/N)^N} \cdot \frac{(\text{tr}[\mathbf{R}_{\mathcal{T}}]/N)^N}{\det[\mathbf{R}_{\mathcal{T}}]}$$

$$= \frac{\rho(\mathbf{R}_{\mathcal{T}})}{\rho(\text{diag}[\mathbf{R}_{\mathcal{T}}])} \tag{7.62}$$

Substituting Eq. (7.62) in Eq. (7.59) and noting that $\rho(\mathbf{R}) = \rho(\mathbf{R}_{\mathcal{T}})$, completes the proof.

The corollary is as follows:

Corollary 7.1 *As* $\ln \rho(\text{diag}[\mathbf{R}_{\mathcal{T}}])$ *is always nonnegative, the performance of a TDLMS algorithm can never be worse than its conventional LMS counterpart.*

The following remark may also be made. When comparing a TDLMS algorithm with its conventional LMS counterpart, the degree of improvement achieved depends on the distribution of the signal power at various outputs of the transformation, that is, the tap inputs $x_{\mathcal{T},i}(n)$. A wide spread of signal power at the taps indicates significant improvement. Similarly, a small spread in signal powers indicates that the improvement achievable is very less.

7.6.4 *Filtering View*

The quantitative result of the above theorem suggests that for a given input process, a transformation matrix will effectively decorrelate the samples of input if it implements a set of parallel FIR filters whose output powers are close to maximally spread. The maximally spread signal powers, here, is quantified by the *minimax theorem*, which was introduced in Chapter 4. When the correlation matrix of the underlying input process is known, the minimax theorem suggests a procedure for the optimal selection of a set of filters, which achieve maximum power spreading. It starts with the design of a set of filters (with orthogonal coefficient vectors) whose output powers are maximized. Instead, it may also start with the design of another set of filters whose output powers are minimized. We also note that these two optimization procedures are implemented independent of each other, but both result in the same set of eigenvectors. This gives an intuitive feeling of how the minimax theorem (procedure) finds a transformation with a maximum spread of signal powers at its outputs.

We note that while the minimax theorem suggests a procedure for the design of the optimal transform for a given input process, the above theorem gives a measure of effectiveness of a transformation matrix in decorrelating the samples of an underlying input process. We note that for a given input process with correlation matrix $\mathbf{R}$, the maximum attainable improvement factor is $I_{\rho,\max} = \ln \rho(\mathbf{R})$, and this is achieved when $\mathcal{T}$ is the KLT of the underlying input process. On the other hand, for a given transformation, $\mathcal{T}$, $I^{\text{n}}_{\rho,\mathcal{T}} = \ln \rho(\text{diag}[\mathbf{R}_{\mathcal{T}}])$. Thus, the difference $I_{\rho,\max} - I^{\text{n}}_{\rho,\mathcal{T}}$ gives a measure of the success of $\mathcal{T}$ in decorrelating the input samples. A small value of $I_{\rho,\max} - I^{\text{n}}_{\rho,\mathcal{T}}$ indicates that the transformation used is close to optimal and vice versa. Furthermore, as explained in Section 7.6.3, $I_{\rho,\max} - I^{\text{n}}_{\rho,\mathcal{T}} = \ln \rho(\mathbf{R}^{\text{n}}_{\mathcal{T}})$ is also the distance of the TDLMS from the ideal LMS–Newton algorithm.

It is instructive to elaborate more on the power spreading effect of a transformation $\mathcal{T}$ and relate that to the above findings. We recall that the output power of a filter with the transfer function $F(e^{j\omega})$, input $x(n)$, and output $y(n)$ is given by (Chapter 2)

$$E[y^2(n)] = \frac{1}{2\pi}\int_0^{2\pi} \Phi_{xx}(e^{j\omega})|F(e^{j\omega})|^2 d\omega \tag{7.63}$$

where $\Phi_{xx}(e^{j\omega})$ is the power spectral density of $x(n)$. Now, if $F(e^{j\omega})$ is the transfer function of a filter whose coefficients constitute the elements of a row of a transformation matrix $\mathcal{T}$, with $\mathcal{T}\mathcal{T}^{\mathrm{T}} = \mathbf{I}$, then $F(e^{j\omega})$ is constrained to satisfy the following identity

$$\frac{1}{2\pi}\int_0^{2\pi} |F(e^{j\omega})|^2 d\omega = 1 \tag{7.64}$$

This follows from the Parseval's relation (Chapter 2, Section 2.2). Noting this, we may say that the diagonal elements of $\mathbf{R}_{\mathcal{T}}$ (i.e., the signal powers at the outputs of the FIR filters defined by the rows of $\mathcal{T}$) are a set of averaged values of the power spectral density function, $\Phi_{xx}(e^{j\omega})$, of the underlying input process. The weighting functions used to obtain these averages are the squared magnitude responses of the FIR filters associated with the various rows of $\mathcal{T}$.

The numerical example that was given in Section 7.3 shows that the DCT is very effective in decorrelating the samples of the input process, $x(n)$, which was considered there. A closer look at this particular example is very instructive. Figure 7.5 shows the power spectral density, $\Phi_{xx}(e^{j\omega})$, of the underlying input process, $x(n)$. The main characteristic of this process to be noted here is that it is of lowpass nature, that is, most of

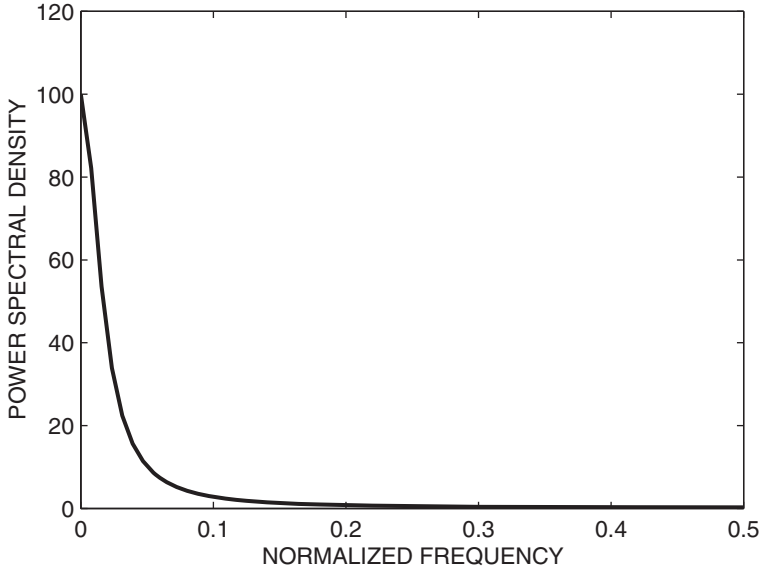

Figure 7.5 Power spectral density of the process $x(n)$ that is generated by the coloring filter Eq.(7.17).

Table 7.3 Transfer functions associated with the various transforms (frequency sampling filters).

$$H^k_{\mathrm{DFT}}(z) = \frac{1-z^{-N}}{1-\mathrm{e}^{-j2\pi k/N}z^{-1}}$$

$$H^k_{\mathrm{RDFT}}(z) = \begin{cases} \frac{1-z^{-N}}{1-z^{-1}}, & \text{for } k = 0 \\ \frac{\left(1-\cos\frac{2\pi k}{N}z^{-1}\right)(1-z^{-N})}{1-2\cos\frac{2\pi k}{N}z^{-1}+z^{-2}}, & \text{for } 1 \le k \le \frac{1}{2}N-1 \\ \frac{1-z^{-N}}{1+z^{-1}}, & \text{for } k = \frac{1}{2}N \\ \frac{z^{-1}(1-z^{-N})}{1-2\cos\frac{2\pi k}{N}z^{-1}+z^{-2}}, & \text{for } \frac{1}{2}N+1 \le k \le N-1 \end{cases}$$

$$H^k_{\mathrm{DHT}}(z) = \frac{\left(1-\left(\cos\frac{2\pi k}{N}-\sin\frac{2\pi k}{N}\right)z^{-1}\right)(1-z^{-N})}{1-2\cos\frac{2\pi k}{N}z^{-1}+z^{-2}}$$

$$H^k_{\mathrm{DCT}}(z) = \frac{(1-z^{-1})(1-(-1)^k z^{-N})}{1-2\cos\frac{\pi k}{N}z^{-1}+z^{-2}}$$

$$H^k_{\mathrm{DST}}(z) = \frac{1+(-1)^k z^{-(N+1)}}{1-2\cos\frac{\pi(k+1)}{N+1}z^{-1}+z^{-2}}$$

here a recursive realization of the DCT filters. Recursive realization of the other transforms which follow the same concept is then straightforward.

From Table 7.3, we have

$$H^k_{\mathrm{DCT}}(z) = \frac{(1-z^{-1})(1-(-1)^k z^{-N})}{1-2\cos\frac{\pi k}{N}z^{-1}+z^{-2}} \tag{7.73}$$

This is the transfer function of the kth DCT filter. Figure 7.9 depicts a detailed realization of Eq. (7.73). In this realization, we have purposefully divided the transfer function of $H^k_{\mathrm{DCT}}(z)$ into three separate parts. Namely, the forward parts, $1-z^{-1}$ and $1-(-1)^k z^{-N}$, and the feedback part, $\frac{1}{1-2\cos\pi k/N z^{-1}+z^{-2}}$. This separation facilitates the integration of the DCT filters (for $k = 0, 1, \ldots, N-1$) in a parallel structure.

Figure 7.10 depicts a block diagram of the DCT frequency sampling filters when they are put together in a parallel structure. Points to be noted here are:

1. For $k = 0$,

$$\frac{1}{1-2\cos\frac{\pi k}{N}z^{-1}+z^{-2}} = \frac{1}{(1-z^{-1})^2}$$

Substituting this result in Eq. (7.73), we obtain

$$H^0_{\mathrm{DCT}}(z) = \frac{1-z^{-N}}{1-z^{-1}} \tag{7.74}$$

This has been considered in the block diagram of Figure 7.10 and, thus, the case $k = 0$ has been treated separately.

Bruun's Algorithm as Sliding DFT

The transfer functions of the DFT frequency sampling filters are (from Table 7.3):

$$H_{\text{DFT}}^{k}(z) = \frac{1 - z^{-N}}{1 - \mathrm{e}^{-j2\pi k/N} z^{-1}}, \quad \text{for } k = 0, 1, \ldots, N-1 \tag{7.75}$$

We note that the zeros of these filters are all taken from the set of Nth roots of unity, that is, $\mathrm{e}^{-j2\pi k/N}$, for $k = 0, 1, \ldots, N-1$. We also note that each DFT filter has one pole that belongs to the same set. As a result, we find that a pole-zero cancellation occurs and, thus, each DFT filter has effectively $N-1$ zeros out of the set of Nth roots of unity and no pole.

Bruun used this simple concept and suggested an elegant factorization of $1 - z^{-N}$ and used these results to form a tree structure, as shown in Figure 7.11 (for $N = 16$), to realize the various FIR frequency sampling filters of DFT. The following identities are used for the factorization of $1 - z^{-N}$:

$$1 - z^{-2M} = (1 - z^{-M})(1 + z^{M}) \tag{7.76}$$

and

$$1 + az^{-2M} + z^{-4M} = (1 + \sqrt{2 - a}\, z^{-M} + z^{-2M})(1 - \sqrt{2 - a}\, z^{-M} + z^{-2M}) \tag{7.77}$$

These factorizations, which are used until the last stage of the tree structure, have the following two features:

1. Each factor consists of either two or three sparse taps.
2. There is at most one nontrivial real-valued coefficient in each factor.

To see how the above identities could be used to develop the tree structure of Figure 7.11, we note that the factors which appear in the first stage are those of $1 - z^{-16} = (1 - z^{-8})(1 + z^{-8})$. The branches that follow after the factor $1 + z^{-8}$ are made of the factors of the other branch of the first stage, that is, $1 - z^{-8} = (1 - z^{-4})(1 + z^{-4})$. Similarly, the branches that follow the factor $1 - z^{-8}$ are made of the factors of $1 + z^{-8} = (1 + \sqrt{2}z^{-2} + z^{-4})(1 - \sqrt{2}z^{-2} + z^{-4})$. The same procedure is used to determine the other branches of the structure. At the end of the third stage (in our particular example), each path of the tree covers 14 out of the 16 zeros of $1 - z^{-16}$. The remaining two zeros that have not been covered by each path are complex conjugates, except for the top path whose corresponding missing zeros are $z = \pm 1$. One out of the two missing zeros is, then, added at the last stage. The same procedure can be used to develop the same structure for any value of N (the transform length), which is a power of 2.

Bruun (1978) elaborated on the tree structure of Figure 7.11 and proposed his FFT structure. In the context of the TDLMS algorithm, we are interested in an efficient implementation of the DFT frequency sampling filters and updating their outputs after the arrival of every new data sample. The tree structure of Figure 7.11 is exactly what we are looking for. Thus, we hold on to this structure as an efficient way of implementing the nonrecursive sliding DFT filters.

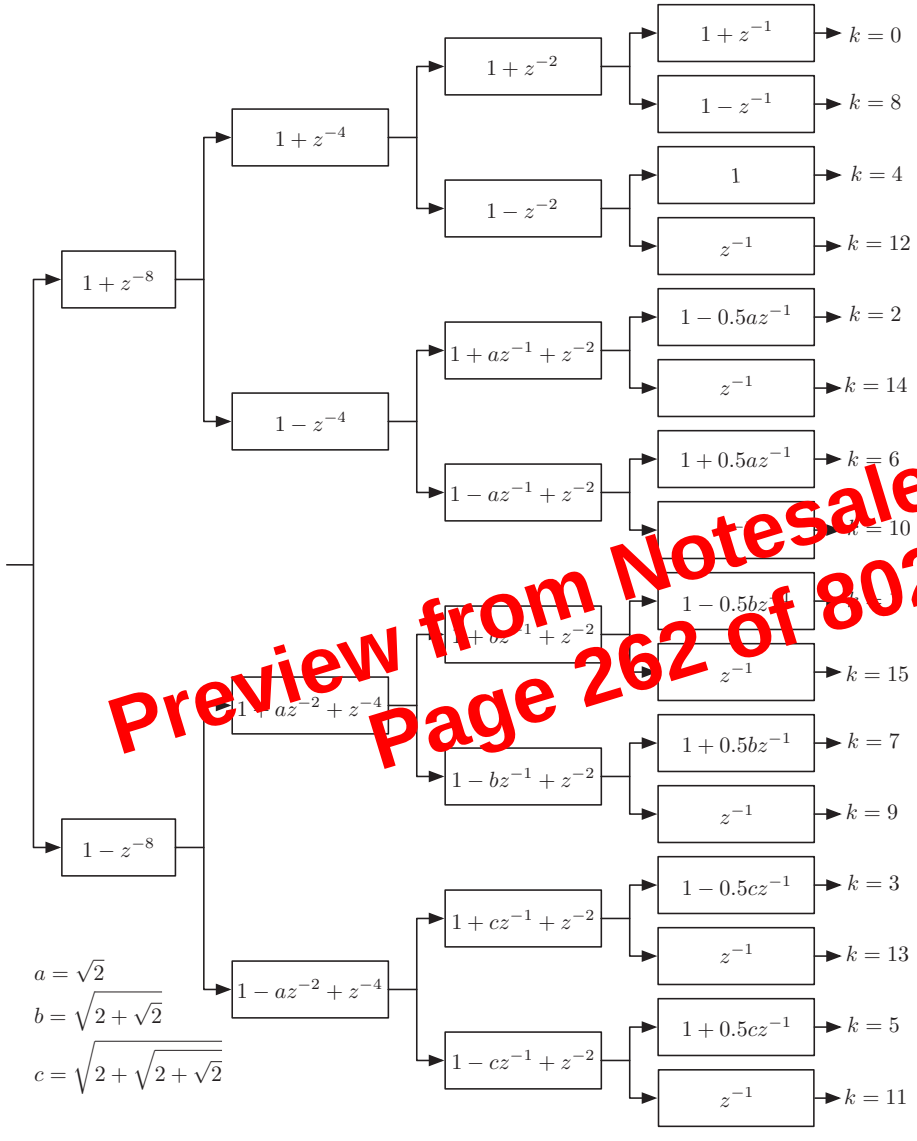

Figure 7.14 Nonrecursive sliding RDFT: $N = 16$ (Farhang-Boroujeny *et al.* 1996).

7.8.4 Comparison of Recursive and Nonrecursive Sliding Transforms

In terms of robustness to numerical round-off errors, the nonrecursive sliding transforms are superior to their recursive counterparts. A simple inspection of the nonrecursive sliding structures shows that each output in these structures is calculated based on a very limited number of multiplications and additions. Furthermore, there is no feedback of numerical errors, thereby avoiding error accumulation. This property, which is inherent to all

Table 8.2 Eigenvalues of $\mathcal{R}_{xx}^{b_i,\mathrm{n}}$ for different number of partitions, P.

P	2	3	4	5	6	7	8	9	10
λ_0	1.500	1.707	1.809	1.866	1.901	1.924	1.940	1.951	1.959
λ_1	0.500	1.000	1.309	1.500	1.623	1.707	1.766	1.809	1.841
λ_2		0.293	0.691	1.000	1.222	1.383	1.500	1.588	1.655
λ_3			0.191	0.500	0.778	1.000	1.174	1.309	1.415
λ_4				0.134	0.376	0.617	0.826	1.000	1.142
λ_5					0.099	0.293	0.500	0.691	0.858
λ_6						0.076	0.234	0.412	0.585
λ_7							0.060	0.191	0.345
λ_8								0.049	0.159
λ_9									0.041
$\lambda_{\max}/\lambda_{\min}$	3	5.828	9.472	13.93	19.20	25.27	32.16	39.86	48.37

that these are widely spread and their dispersion increases significantly as P grows. This means that for large values of P, the PFBLMS algorithm may suffer from slow convergence and/or numerical instability (as in Eq. (8.88), $\mathcal{R}_{xx}^{b_i,\mathrm{n}}$ becomes badly ill-conditioned, for large P.

Observe from the PFBLMS structure shown in Figure 8.4 that the successive partitions of the input samples are 50% overlapped. The value of $|\alpha_i| = 0.5$ in Eq. (8.88), which in turn results in the large eigenvalue spread in $\mathcal{R}_{xx}^{b_i,\mathrm{n}}$ is a direct consequence of this 50% overlapping. Numerical studies show that this eigenvalue spread reduces as $|\alpha_i|$ decreases. Furthermore, $|\alpha_i|$ can be reduced by reducing the amount of overlap of the successive partitions of the input samples. This is easily achieved by choosing a block length, L, smaller than the partition length, M, as explained in the next section.

Before proceeding with this modification of the PFBLMS algorithm, we shall make some comments on the convergence behavior of the constrained PFBLMS algorithm. As was noted before, the correlation matrix $\mathcal{R}_{xx}^{b_i,\mathrm{n}}$ was the outcome of an analysis of the unconstrained PFBLMS algorithm. A detailed examination of the constrained PFBLMS algorithm is rather involved and beyond the scope of this book. As we will demonstrate through computer simulations later, the effect of overlapping of successive blocks is resolved when the tap weights of the filter are constrained. As a result, we find that the constrained PFBLMS algorithm does not have any convergence problem. It converges almost as fast as its nonpartitioned counterpart. These observations are in line with the theoretical findings that have presented in Chan (2000) and also in Chan and Farhang-Boroujeny (2001).

8.4.2 PFBLMS Algorithm with $M > L$

Assuming a block length L and a partition length M, define the vector

$$\tilde{\mathbf{x}}_0(k) = [x(kL - M)\, x(kL - M + 1) \cdots x(kL + L - 1)]^{\mathrm{T}} \tag{8.89}$$

Let us choose $M = pL$, where p is an integer. As we show later, this choice of L and M leads to an efficient implementation of the PFBLMS algorithm. We note that

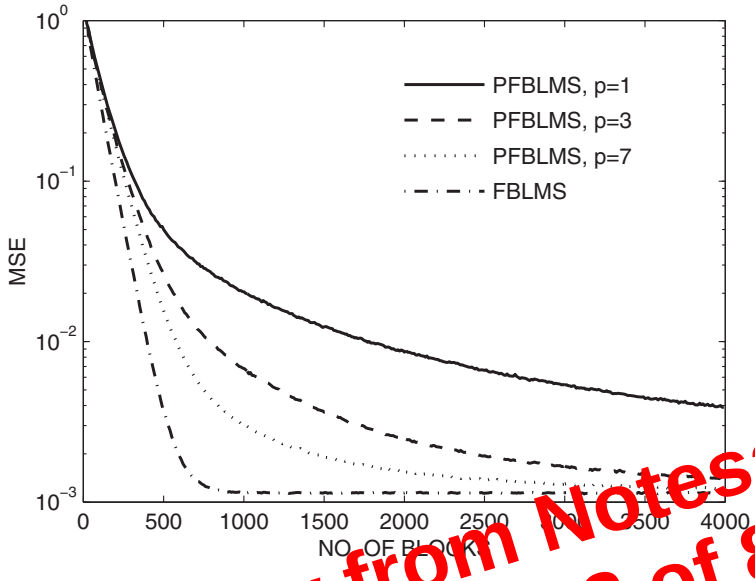

Figure 8.7 Learning curves of the FBLMS and PFBLMS algorithms with white input.

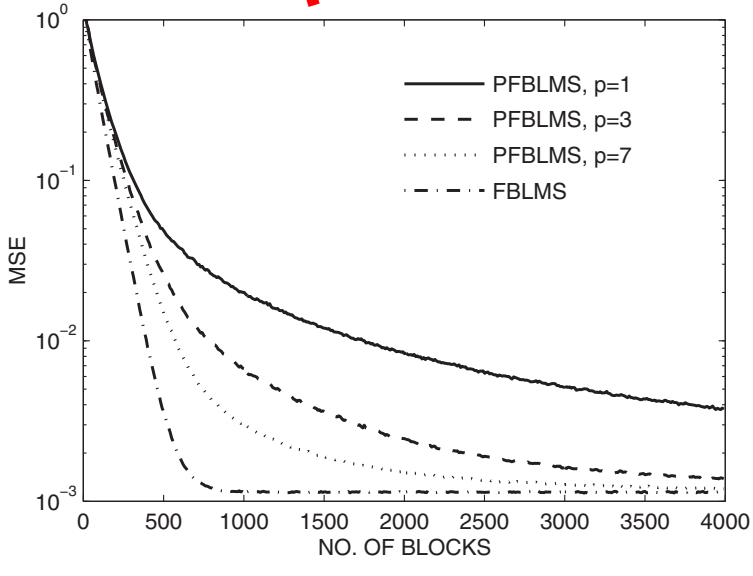

Figure 8.8 Learning curves of the FBLMS and PFBLMS algorithms for a colored input.

is the diagonal matrix consisting of the elements 1, $e^{j2\pi/N}$, $e^{j4\pi/N}$, ..., $e^{j2(N-1)\pi/N}$.

(iii)

$$\begin{aligned} y(k) &= \text{the last term of } \quad \boldsymbol{\mathcal{F}}^{-1}(\mathbf{w}_{\mathcal{F}}(k) \odot \mathbf{x}_{\mathcal{F}}(k)) \\ &= \mathbf{w}_{\mathcal{F}}^{\mathrm{T}}(k)\mathbf{x}_{\mathcal{F}}(k) \end{aligned}$$

where $\odot$ denotes the element-wise multiplication of vectors.

(iv) Now, consider the TDLMS algorithm with $\boldsymbol{\mathcal{T}} = \boldsymbol{\mathcal{F}}$. Write down the equations corresponding to this case and compare them with the above results. Verify that the FBLMS algorithm with block length $L = 1$ is equivalent to the TDLMS algorithm with $\boldsymbol{\mathcal{T}} = \boldsymbol{\mathcal{F}}$.

P8.11 An alternative procedure for derivation of Eq. (8.61) is proposed in this problem.

(i) Show that

$$\mathbf{P}_{0,L} = \boldsymbol{\mathcal{F}}\boldsymbol{\mathcal{P}}_{0,L}^{*}\boldsymbol{\mathcal{F}}^{-1}$$

and conclude that $\boldsymbol{\mathcal{P}}_{0,L}$ is a circular matrix.

(ii) Show that

$$\text{First column of } \boldsymbol{\mathcal{P}}_{0,L} = \frac{1}{N'}\boldsymbol{\mathcal{F}}\mathbf{p}_{0,L}$$

where $\mathbf{p}_{0,L}$ is the column vector consisting of the diagonal elements of $\mathbf{P}_{0,L}$.

(iii) Considering the fact that $\boldsymbol{\mathcal{X}}_{\mathcal{F}}(k)$ is a diagonal matrix, show that

$$\boldsymbol{\mathcal{X}}_{\mathcal{F}}^{*}(k)\boldsymbol{\mathcal{P}}_{0,L}\boldsymbol{\mathcal{X}}_{\mathcal{F}} = \boldsymbol{\mathcal{P}}_{0,L} \odot ([\mathbf{x}_{\mathcal{F}}(k)\mathbf{x}_{\mathcal{F}}^{\mathrm{H}}(k)]^{*})$$

where $\mathbf{x}_{\mathcal{F}}(k)$ is the column vector consisting of the diagonal elements of $\boldsymbol{\mathcal{X}}_{\mathcal{F}}(k)$, and $\odot$ denotes element-wise multiplication of the matrices. Thus, show that

$$\boldsymbol{\mathcal{R}}_{xx}^{u} = \boldsymbol{\mathcal{P}}_{0,L} \odot \boldsymbol{\mathcal{R}}_{xx}^{*}$$

where $\boldsymbol{\mathcal{R}}_{xx} = E[\mathbf{x}_{\mathcal{F}}(k)\mathbf{x}_{\mathcal{F}}^{\mathrm{H}}(k)]$.

(iv) Assuming that the cross-correlation between different elements of the vector $\mathbf{x}_{\mathcal{F}}(k)$ are negligible, show that

$$\boldsymbol{\mathcal{R}}_{xx} \approx N' \times \operatorname{diag}\left(\Phi_{xx}\left(e^{j\frac{2\pi\times 0}{N'}}\right), \Phi_{xx}\left(e^{j\frac{2\pi}{N'}}\right), \ldots, \Phi_{xx}\left(e^{j\frac{2\pi(N'-1)}{N'}}\right)\right).$$

(v) Using the results of (iii) and (iv), derive Eq. (8.61).

(vi) Do a thorough study of the elements of the matrix $\boldsymbol{\mathcal{P}}_{0,L}$. In particular, verify that the largest (in magnitude) elements of $\boldsymbol{\mathcal{P}}_{0,L}$ are its diagonal elements. Use your findings to conclude that $\boldsymbol{\mathcal{R}}_{xx}^{u}$ is closer to diagonal than $\boldsymbol{\mathcal{R}}_{xx}$, in the sense that the nondiagonal elements of the normalized matrix $(\operatorname{diag}[\boldsymbol{\mathcal{R}}_{xx}^{u}])^{-1}\boldsymbol{\mathcal{R}}_{xx}^{u}$ are smaller than the corresponding elements of $(\operatorname{diag}[\boldsymbol{\mathcal{R}}_{xx}])^{-1}\boldsymbol{\mathcal{R}}_{xx}$.

P8.12 Verify the results presented in Eq. (8.87).

P8.13 Verify the results presented in Eq. (8.93).

of $\boldsymbol{\mathcal{X}}_{\mathcal{F}}(k)$, and do some manipulations and approximation similar to what was done above, we obtain

$$\begin{aligned}\mathbf{K}_{\mathcal{F}}(k+1) \approx \mathbf{K}_{\mathcal{F}}(k) - 2\mu_{\mathrm{o}}\boldsymbol{\Lambda}^{-1}\boldsymbol{\mathcal{R}}^{u}_{xx}\mathbf{K}_{\mathcal{F}}(k) - 2\mu_{\mathrm{o}}\mathbf{K}_{\mathcal{F}}(k)\boldsymbol{\mathcal{R}}^{u}_{xx}\boldsymbol{\Lambda}^{-1} \\ + 4\mu_{\mathrm{o}}^{2}\boldsymbol{\Lambda}^{-1}E[\boldsymbol{\mathcal{X}}^{*}_{\boldsymbol{\mathcal{F}}}(k)(\mathcal{P}_{0,L}\mathbf{e}_{\mathrm{o},\mathcal{F}}(k))(\mathcal{P}_{0,L}\mathbf{e}_{\mathrm{o},\mathcal{F}}(k))^{\mathrm{H}}\boldsymbol{\mathcal{X}}_{\mathcal{F}}(k)]\boldsymbol{\Lambda}^{-1} \qquad (8\mathrm{B}.20)\end{aligned}$$

where $\mathbf{K}_{\mathcal{F}}(k) = E[\mathbf{v}_{\mathcal{F}}(k)\mathbf{v}^{\mathrm{H}}_{\boldsymbol{\mathcal{F}}}(k)]$. Since $\mathbf{e}_{o,\mathcal{F}}(k)$ and $\boldsymbol{\mathcal{X}}_{\mathcal{F}}(k)$ are independent, we get using Eq. (8B.14)

$$\begin{aligned}&E[\boldsymbol{\mathcal{X}}^{*}_{\boldsymbol{\mathcal{F}}}(k)(\mathcal{P}_{0,L}\mathbf{e}_{\mathrm{o},\mathcal{F}}(k))(\mathcal{P}_{0,L}\mathbf{e}_{\mathrm{o},\mathcal{F}}(k))^{\mathrm{H}}\boldsymbol{\mathcal{X}}_{\mathcal{F}}(k)] \\ &\quad = E[\boldsymbol{\mathcal{X}}^{*}_{\boldsymbol{\mathcal{F}}}(k)E[(\mathcal{P}_{0,L}\mathbf{e}_{\mathrm{o},\mathcal{F}}(k))(\mathcal{P}_{0,L}\mathbf{e}_{\mathrm{o},\mathcal{F}}(k))^{\mathrm{H}}]\boldsymbol{\mathcal{X}}_{\mathcal{F}}(k)] \\ &\quad = N'\xi_{\min}E[\boldsymbol{\mathcal{X}}^{*}_{\boldsymbol{\mathcal{F}}}(k)\mathcal{P}_{0,L}\boldsymbol{\mathcal{X}}_{\mathcal{F}}(k)] \\ &\quad = N'\xi_{\min}\boldsymbol{\mathcal{R}}^{u}_{xx} \qquad (8\mathrm{B}.21)\end{aligned}$$

We note that $\boldsymbol{\Lambda}$ is a diagonal matrix consisting of the estimates of the powers of the input signal samples in the frequency domain. Considering the spectral separation property of the DFT (see e.g., Oppenheim and Schafer (1975)), we obtain

$$\boldsymbol{\Lambda} \approx N' \times \mathrm{diag}\left(\Phi_{xx}\left(e^{j\frac{2\pi\times 0}{N'}}\right), \Phi_{xx}\left(e^{j\frac{2\pi}{N'}}\right), \ldots, \Phi_{xx}\left(e^{j\frac{2\pi(N'-1)}{N'}}\right)\right) \qquad (8\mathrm{B}.22)$$

where $\Phi_{xx}(e^{j\omega})$ is the power spectral density of the underlying input process, $x(n)$. The factor N' in Eq. (8B.22) is the length of the DFT in the present case. Comparing Eq. (8B.22) with Eq. (8.61), we find that

$$\boldsymbol{\Lambda} \approx \frac{N'}{L}\boldsymbol{\mathcal{R}}^{u}_{xx} \qquad (8\mathrm{B}.23)$$

Substituting Eqs. (8B.21) and (8B.23) in Eq. (8B.20), we get

$$\mathbf{K}_{\mathcal{F}}(k+1) \approx \mathbf{K}_{\mathcal{F}}(k) - 4\mu_{o}\frac{L}{N'}\mathbf{K}_{\mathcal{F}}(k) + 4\mu_{o}^{2}\xi_{\min}\frac{L^{2}}{N'}\boldsymbol{\mathcal{R}}^{u-1}_{xx} \qquad (8\mathrm{B}.24)$$

In the steady state, when $\mathbf{K}_{\mathcal{F}}(k+1) = \mathbf{K}_{\mathcal{F}}(k)$, we obtain

$$\mathbf{K}_{\mathcal{F}}(k) \approx \mu_{o}\xi_{\min}L\boldsymbol{\mathcal{R}}^{u-1}_{xx} \qquad (8\mathrm{B}.25)$$

Substituting Eq. (8B.25) in Eq. (8B.7), we get

$$\xi^{\mathrm{un}}_{\mathrm{excess}} \approx \mu_{o}\xi_{\min} \qquad (8\mathrm{B}.26)$$

Substituting this result in Eq. (8B.1), we obtain the misadjustment for the unconstrained FBLMS algorithm with step-normalization as

$$\mathcal{M}^{\mathrm{un}}_{\mathrm{FBLMS}} \approx \mu_{\mathrm{o}} \qquad (8\mathrm{B}.27)$$

Constrained FBLMS algorithm without step-normalization

In this case, the FBLMS algorithm is an exact and fast implementation of the BLMS algorithm, that is, with a reduced computational complexity. Hence, the corresponding excess MSE is given by Eq. (8B.16). Noting that $\mathbf{R}$ is N-by-N and its diagonal elements are all equal to $\phi_{xx}(0)$, Eq. (8A.16) may also be written as

$$\xi^{c}_{\text{excess}} = \mu N \xi_{\min} \phi_{xx}(0) \tag{8B.28}$$

to be in line with the rest of the results in this appendix. Substituting this result in Eq. (8B.1), we obtain the corresponding misadjustment as

$$\mathcal{M}^{\text{cn}}_{\text{FBLMS}} \approx \mu N \phi_{xx}(0) \tag{8B.29}$$

Constrained FBLMS algorithm with step-normalization

Premultiplication of the gradient vector $\mathcal{X}^{*}_{\mathcal{F}}(k)\mathbf{e}_{\mathcal{F}}(k)$ by the matrix $2\mu_o \mathbf{\Lambda}^{-1}$ implements the constraining step (8.47). Combining this step with step-normalization, we get the recursion

$$\begin{aligned}\mathbf{v}_{\mathcal{F}}(k+1) = {} & (\mathbf{I} - 2\mu_o \mathbf{\Lambda}^{-1}\mathcal{P}_{N,0}\mathcal{X}^{*}_{\mathcal{F}}(k)\mathcal{P}_{0,L}\mathcal{X}_{\mathcal{F}}(k))\mathbf{v}_{\mathcal{F}}(k) \\ & + 2\mu_o \mathbf{\Lambda}^{-1}\mathcal{P}_{N,0}\mathcal{X}^{*}_{\mathcal{F}}(k)\mathcal{P}_{0,L}\mathbf{e}_{o,\mathcal{F}}(k)\end{aligned} \tag{8B.30}$$

analogous to Eq. (8B.19). Following the same line of derivations as in the case of Eq. (8B.19), we obtain

$$\begin{aligned}& - 2\mu_o \mathbf{\Lambda}^{-1}\mathcal{P}_{N,0}\mathcal{R}^{u}_{xx}\mathbf{K}_{\mathcal{F}}(k) - 2\mu_o \mathbf{K}_{\mathcal{F}}(k)\mathcal{R}^{u}_{xx}\mathcal{P}_{N,0}\mathbf{\Lambda}^{-1} \\ & \qquad + 4\mu_o^2 N' \xi_{\min} \mathbf{\Lambda}^{-1}\mathcal{P}_{N,0}\mathcal{R}^{u}_{xx}\mathcal{P}_{N,0}\mathbf{\Lambda}^{-1} = 0\end{aligned} \tag{8B.31}$$

We shall now solve this equation to find $\mathbf{K}_{\mathcal{F}}(k)$.

To proceed, let us define

$$\mathbf{G} = \mathbf{\Lambda}^{-1}\mathcal{P}_{N,0}\mathcal{R}^{u}_{xx} \tag{8B.32}$$

and note that $\mathbf{G}^{\text{H}} = \mathcal{R}^{u}_{xx}\mathcal{P}_{N,0}\mathbf{\Lambda}^{-1}$ as $\mathcal{P}_{N,0}$ is Hermitian and $\mathcal{R}^{u}_{xx}$ and $\mathbf{\Lambda}^{-1}$ are diagonal matrices. Using these, Eq. (8B.31) may be rearranged as

$$\mathbf{G}\mathbf{K}_{\mathcal{F}}(k) - \mu_o N' \xi_{\min}\mathbf{G}\mathcal{P}_{N,0}\mathbf{\Lambda}^{-1} + \mathbf{K}_{\mathcal{F}}(k)\mathbf{G}^{\text{H}} - \mu_o N' \xi_{\min}\mathbf{\Lambda}^{-1}\mathcal{P}_{N,0}\mathbf{G}^{\text{H}} = 0 \tag{8B.33}$$

or

$$\mathbf{G}(\mathbf{K}_{\mathcal{F}}(k) - \mu_o N' \xi_{\min}\mathcal{P}_{N,0}\mathbf{\Lambda}^{-1}) + (\mathbf{K}_{\mathcal{F}}(k) - \mu_o N' \xi_{\min}\mathbf{\Lambda}^{-1}\mathcal{P}_{N,0})\mathbf{G}^{\text{H}} = 0 \tag{8B.34}$$

General solution of Eq. (8B.37) turns out to be difficult. However, a trivial solution of that, which closely matches the simulation results, can be easily identified as

$$\mathbf{K}_{\mathcal{F}}(k) = \mu_o N' \xi_{\min}\mathcal{P}_{N,0}\mathbf{\Lambda}^{-1} \tag{8B.35}$$

9

Subband Adaptive Filters

In the last two chapters, we have discussed two classes of LMS adaptive filtering algorithms that have improved convergence behavior compared to the conventional LMS algorithm. Convergence improvement in both classes was found to be a direct consequence of using orthogonal transforms for decomposing the filter input into a number of partially mutually exclusive bands. This was referred to as *band-partitioning*. Moreover, our study of transform domain adaptive filters in Chapter 7 clearly showed that the imperfect separation of the input signal into mutually exclusive bands is the main reason for the suboptimal convergence behavior of such filters.

In this chapter, we present another class of adaptive filters that also uses the concept of band-partitioning to improve the convergence behavior of LMS algorithm. This structure, which is called *subband adaptive filter*, is different from the transform domain adaptive filters in many ways. Firstly, the filters used for band-partitioning of the input signal are well-designed filters with *high* stop-band rejection, that is, very low side lobes. As a result, we find that the subband adaptive filters achieve higher degree of improvement in convergence compared to the transform domain adaptive filters of Chapter 7. Secondly, because of the high stop-band rejection, the subband signals can be *decimated* (down-sampled to a lower rate) before doing any filtering in subbands. Thirdly, implementation of subband filters at a decimated rate results in significant reduction in the computational complexity of the overall filter. However, this reduction is not as significant as what is usually achieved by the fast block LMS (FBLMS) algorithm of Chapter 8. We will make some comments on comparison of the subband adaptive structure and FBLMS algorithm in Section 9.11.

The subject of subband filtering is closely related to *multirate signal processing*. In a subband adaptive filter, the filter input is first partitioned into a set of subband signals through an analysis filter bank. These subband signals are then decimated to a lower rate and passed through a set of independent or partially independent adaptive filters that operate at the decimated rate. The outputs from these filters are subsequently combined together using a synthesis filter bank to reconstruct the fullband output of the overall filter. The DFT filter banks are commonly used for efficient realization of the analysis and synthesis filter banks. We thus start this chapter with a short review of the DFT filter banks and introduce the method of *weighted overlap–add* for efficient realization of these filter banks. We also discuss the conditions that should be imposed on the analysis and synthesis filters so that the reconstructed fullband signals have negligible distortion.

Adaptive Filters: Theory and Applications, Second Edition. Behrouz Farhang-Boroujeny.

Then, using Eq. (9.12), we can write Eq. (9.11) as

$$y(n) = \sum_{k=-\infty}^{\infty} \hat{y}_k(n - kL) \tag{9.13}$$

That is, the output sequence $y(n)$ is obtained by overlapping and adding the sequences $\hat{y}_k(n)$'s, thus the name overlap–add.

Equation (9.12) may also be written as

$$\hat{y}_k(n) = g_n \tilde{y}_k(n) \tag{9.14}$$

where

$$\tilde{y}_k(n) = \frac{1}{M} \sum_{i=0}^{M-1} [\bar{y}_i(k) W_M^{ikL}] W_M^{in} \tag{9.15}$$

Note that $\tilde{y}_k(n)$ is a periodic function of n with period M as W_M^{in} is periodic in i with period M, and the rest of the terms on the right-hand side of Eq. (9.15) are independent of n. Furthermore, it is straightforward to see that the values of $\tilde{y}_k(n)$ for $n = 0, 1, \ldots, M - 1$ (i.e., the first period of $\tilde{y}_k(n)$) are samples of the inverse DFT of the sequence $\bar{y}_i(k) W_M^{ikL}$, for $i = 0, 1, \ldots, M - 1$.

From the above observations, we may adopt the following procedure to generate the samples of the synthesized output sequence, $y(n)$:

1. Upon the receipt of the latest samples of the subband signals, say $\bar{y}_i(k)$, for $i = 0, 1, \ldots, M - 1$, we construct the vector

$$\bar{\mathbf{y}}(k) = [\bar{y}_0(k) \;\; \bar{y}_1(k) W_M^{kL} \;\; \bar{y}_2(k) W_M^{2kL} \cdots \bar{y}_{M-1}(k) W_M^{k(M-1)L}]$$

   and compute the inverse DFT of $\bar{\mathbf{y}}(k)$.
2. The result of this inverse DFT is repeated to generate a periodic sequence. This makes the sequence $\tilde{y}_k(n)$ of Eq. (9.15).
3. The sequence $\hat{y}_k(n)$ is obtained by multiplying the sequences $\tilde{y}_k(n)$ and g_n on an element-by-element basis, as in Eq. (9.14). Assuming that g_n is causal, $\hat{y}_k(n)$ will also be causal.
4. Finally, to generate the samples of $y(n)$, the sequence $\hat{y}_k(n)$ is added to a buffer holding the accumulated results of the previous iterations, that is, $\sum_{l=-\infty}^{k-1} \hat{y}_l(n - lL)$. The first L elements of the updated buffer are the samples $y(kL)$, $y(kL + 1), \ldots, y(kL + L - 1)$ of the synthesized output. While these samples are being sent to the output, the content of the buffer is shifted and filled with zeros from its other end and becomes ready for stacking the next set of samples, that is, $\hat{y}_{k+1}(n)$, in the next iteration.

9.2 Complementary Filter Banks

In multirate signal processing, in general, analysis and synthesis filters need to satisfy certain conditions in order that the reconstructed fullband signals have no or, at least, insignificant distortion. For this to be true in subband adaptive filters, we find

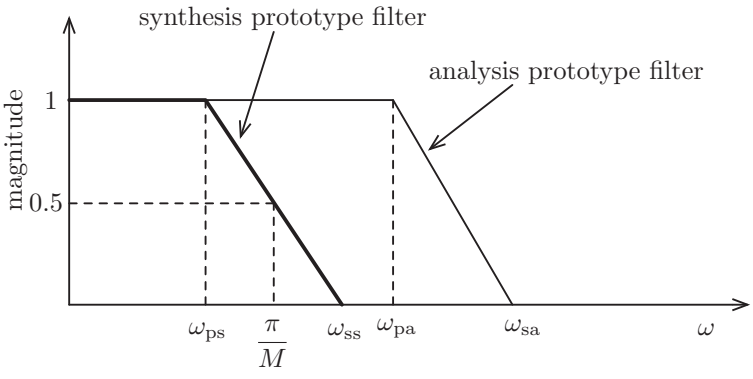

Figure 9.9 A possible choice of analysis and synthesis prototype filters that resolves the problem of slow convergence of subband adaptive filters.

Figure 9.9 presents a diagram showing a good choice of analysis and synthesis prototype filters that resolves the problem of slow convergence of subband adaptive filters. The analysis prototype filter is chosen such that it has a flat magnitude response and linear phase response (constant group-delay[3] between zero and a frequency larger than or equal to ω_{ss}, where ω_{ss} is the beginning of the stop-band (i.e., the end of the transition band) of the synthesis prototype filter. Moreover, the synthesis filters are chosen to be complementary. The cascade of the analysis and synthesis filters will then be a complementary filter bank because in this case the multiplication (cascade) of the analysis and synthesis prototype filters is just the same as the synthesis prototype filter. The analysis filters introduce only a fixed delay in the overall response of the subband structure.

Next, we explain why the choice of the analysis and synthesis prototype filters, as shown in Figure 9.9, resolves the problem of poor convergence of subband adaptive filters. Assuming that the power spectral density of the fullband input, $x(n)$, does not vary significantly over each subband, using an analysis prototype filter similar to the one shown in Figure 9.9, would result in all decimated subband sequences to have approximately flat spectra over the range of frequencies $|\omega| \leq \omega_{pa}$. On the other hand, the band of interest over which matching between the frequency response of each subband adaptive filter and its associated desired response from the respective band of the plant should be achieved is $|\omega| < \omega_{ss}$, as frequencies beyond this are cut off by the synthesis filters (Figure 9.9). We may thus say that in a subband adaptive filter structure whose analysis and synthesis prototype filters are selected as shown in Figure 9.9, all the subband filters will be well excited over their respective bands of interest, and hence there will not be any slow mode, which may affect the convergence behavior of the overall filter.

Another consideration that should be noted in the implementation of subband adaptive filters, and hence in the design of analysis and synthesis filters, is the problem of delay (or latency) in the filter output, $y(n)$. This delay is caused by the analysis and synthesis filters. Minimization of this delay is exceedingly important as the maximum delay permitted in many applications is often very much limited. For instance, in the application of acoustic

[3] Group-delay of a system is defined as the derivative of its phase response with respect to the angular frequency, ω.

although may seem quite reasonable at the first glance, has some drawbacks when it comes to adaptation of the subband filters. It results in significant augmentation of the misadjustment, as explained next.

The Fourier transform of the desired signal, $\bar{d}_i(n)$, in the ith subband is given by

$$\bar{D}_i(\mathrm{e}^{j\omega}) = \sum_{m=i-1}^{i+1} X(\mathrm{e}^{j\frac{\omega-2\pi m}{L}}) W_{\mathrm{o}}(\mathrm{e}^{j\frac{\omega-2\pi m}{L}}) H(\mathrm{e}^{j\frac{\omega-2\pi m}{L}}) \tag{9.24}$$

where $X(\mathrm{e}^{j\omega})$ is the Fourier transform of the input, $x(n)$, and $W_{\mathrm{o}}(\mathrm{e}^{j\omega})$ and $H(\mathrm{e}^{j\omega})$ are the frequency responses of the plant and the analysis prototype filter, respectively. The three terms contributing to the spectrum of $\bar{d}_i(n)$ are (i) the ith band spectrum, $m = i$, (ii) the aliased spectrum from the immediately following band, $m = i + 1$, and (iii) the aliased spectrum from the immediately preceding band, $m = i - 1$. The division of the frequency, ω, by L is due to the spectral expansion because of the L-fold decimation. Similarly, the Fourier transform of the output, $\bar{y}_i(k)$, of the ith subband filter is obtained as

$$\bar{Y}_i(\mathrm{e}^{j\omega}) = \bar{W}_i(\mathrm{e}^{j\omega}) \sum_{m=i-1}^{i+1} X(\mathrm{e}^{j\frac{\omega-2\pi m}{L}}) H(\mathrm{e}^{j\frac{\omega-2\pi m}{L}}) \tag{9.25}$$

Using Eqs. (9.24) and (9.25), the Fourier transform of the subband error sequence $\bar{e}_i(k) = \bar{d}_i(k) - \bar{y}_i(k)$ is obtained as

$$\begin{aligned}
\bar{E}_i(\mathrm{e}^{j\omega}) &= \bar{D}_i(\mathrm{e}^{j\omega}) - \bar{Y}_i(\mathrm{e}^{j\omega}) \\
&= \left[W_{\mathrm{o}}(\mathrm{e}^{j\frac{\omega-2\pi i}{L}}) - \bar{W}_i(\mathrm{e}^{j\omega}) \right] H(\mathrm{e}^{j(\omega-2\pi i)/L}) X(\mathrm{e}^{j\frac{\omega-2\pi i}{L}}) \\
&\quad + \left[W_{\mathrm{o}}(\mathrm{e}^{j\frac{\omega-2\pi(i+1)}{L}}) - \bar{W}_i(\mathrm{e}^{j\omega}) \right] H(\mathrm{e}^{j\frac{\omega-2\pi(i+1)}{L}}) X(\mathrm{e}^{j\frac{\omega-2\pi(i+1)}{L}}) \\
&\quad + \left[W_{\mathrm{o}}(\mathrm{e}^{j\frac{\omega-2\pi(i-1)}{L}}) - \bar{W}_i(\mathrm{e}^{j\omega}) \right] H(\mathrm{e}^{j\frac{\omega-2\pi(i-1)}{L}}) X(\mathrm{e}^{j\frac{\omega-2\pi(i-1)}{L}})
\end{aligned} \tag{9.26}$$

Inspection of Eq. (9.26) reveals that to minimize

$$E[|e_i(k)|^2] = \frac{1}{2\pi} \int_{-\pi}^{\pi} |\bar{E}_i(\mathrm{e}^{j\omega})|^2 \mathrm{d}\omega$$

$\bar{W}_i(\mathrm{e}^{j\omega})$ has to be selected so that the three differences in the brackets on the right-hand side of Eq. (9.26) reduce to some small values. Moreover, we note that the frequency interval $-\pi \le \omega \le \pi$ may be divided into three distinct ranges. The first range, defined as $-\omega_1 \le \omega \le \omega_1$, is where there is no overlap between $H(\mathrm{e}^{j\frac{\omega-2\pi i}{L}})$ and its shifted versions, $H(\mathrm{e}^{j\omega-2\pi(i-1)/L})$ and $H(\mathrm{e}^{j\omega-2\pi(i+1)/L})$. In this range, the last two terms on the right-hand side of Eq. (9.26) are zero as this range coincides with the stop-bands of $H(\mathrm{e}^{j\omega-2\pi(i-1)/L})$ and $H(\mathrm{e}^{j\frac{\omega-2\pi(i+1)}{L}})$. The first term can also be made small by choosing $\bar{W}_i(\mathrm{e}^{j\omega})$ to be close to the plant response, $W_{\mathrm{o}}(\mathrm{e}^{j\frac{\omega-2\pi i}{L}})$, for $-\omega_1 \le \omega \le \omega_1$. It is important to note that a selection of $L \le L_{\max}$ implies that $\omega_{ss} \le \omega_1$. This in turn means that the portions of the plant response that are picked up by the synthesis filters can be modeled well by the subband adaptive filters.

where α is a positive parameter that specifies the width of the transition band of the prototype filter of the filter bank as explained below. The parameter α, as noted above, is known as *roll-off factor*.

Substituting Eq. (9.37) in Eq. (9.36), we get

$$\omega_p = (1 - \alpha)\frac{\pi}{M} \tag{9.38}$$

Also, the width of the transition band of the prototype filter is obtained as

$$\omega_s - \omega_p = \frac{2\pi\alpha}{M} \tag{9.39}$$

Equation (9.34) explicitly states that the group-delay introduced by the filter bank is KM. In most subband adaptive filtering applications, we want to keep this delay as small as possible. On the other hand, the optimum K that results in maximum attenuation in the stop-band is obtained by choosing K so that KM is the nearest multiple of M to $N_a/2$. However, this delay is generally large and thus, we would instead strike a compromise between delay and stop-band attenuation. That is, we may accept a lower delay at the cost of lower stop-band attenuation.

For effective implementation of subband adaptive filters it is important to understand the effect of reduced delay on the performance of the analysis and synthesis filters and its overall impact on the performance of the adaptive filter. This can be best understood through an example.

Figure 9.11 shows the magnitude and group-delay responses of three filters that have been designed by the above method. The filter length, N_a, the number of subbands, M, and the roll-off factor, α, are set equal to 97, 4, and 0.25, respectively, and the three designs are differentiated by the parameter K. The separation of the pass-band, transition band, and stop-band can be clearly seen in the responses. In particular, we note that the transition bands in the three designs are the same and match the band edges predicted by Eq. (9.37) and Eq. (9.38). We also note that the price to be paid for achieving reduced delay is lower stop-band attenuation, an undesirable boost in the magnitude response in the transition band, and group-delay distortion in the transition and stop bands. However, the magnitude and group-delay responses in the pass-band remain nearly undistorted. This is a desirable feature of this design method that makes it very appropriate for designing analysis as well as synthesis filters in the application of subband adaptive filtering.

9.8 A Design Procedure for Subband Adaptive Filters

Since there are many compromises to be made in the overall design of subband adaptive filters, it is very hard to suggest a simple design procedure for such filters. In this section, we present a procedure that the author has found useful in his research work. This procedure is iterative in nature and its application requires some experience. Hence, a novice needs to do some experiments with that before he/she can use it for actual design.

To choose all the parameters necessary for setting up a subband adaptive filter, one may take the following steps:

1. Choose a value for the number of subbands, M.
2. Choose an integer parameter, J, in the range of $\frac{1}{2}$ to $\frac{2}{3}$ of M, and select the pass-band, transition band, and stop-band of the analysis and synthesis prototype filters, as shown

Table 9.1 Summary of the two designs of analysis–synthesis prototype filters.

Parameters	Conventional delay	Low delay
K_a	5	3
K_s	3	1
N_a	191	289
N_s	193	353
Δ	192	96
E_a	1.4×10^{-6}	5.0×10^{-7}
E_s	4.1×10^{-7}	4.6×10^{-7}

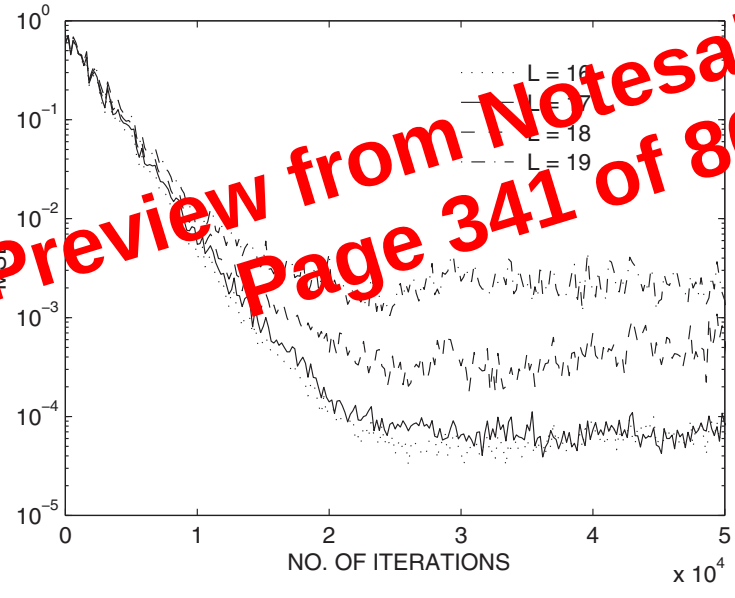

Figure 9.13 Learning curves of the subband adaptive filter for different values of the decimation factor, L.

signals do not suffer from any aliasing. On the contrary, $L = 19$ corresponds to the case where the decimated subband signals are fully aliased over the transition bands of their respective analysis filters. However, the signals in their pass-bands do not suffer from any serious aliasing, except that due to nonideal stop-band attenuations that are negligible. The case $L = 17$ does suffer from aliasing in transition bands, although relatively low. These results clearly confirm our earlier conjecture that in the selection of the decimation factor, L, a small amount of aliasing is acceptable.

10

IIR Adaptive Filters

In our study of adaptive filters in the previous chapters, we always limited ourselves to filters with finite-impulse response (FIR). The main feature of FIR filters, which has made them the most attractive structure in the application of adaptive filters, is that they are nonrecursive. That is, the filter output is computed based on only a finite number of input samples. This, as we noted in the previous chapters, results in a quadratic mean-squared error (MSE) performance surface, allowing us to use any of the simple gradient-based algorithms for finding the optimum coefficients (tap weights) of the filter.

The use of *recursive* or infinite-impulse response (IIR) filters, on the other hand, have been less popular in the realization of adaptive filters for the following reasons:

1. IIR filters can easily become unstable because their poles may get shifted out of the unit circle (i.e., $|z| = 1$, in the z-plane) during the adaptation process.
2. The performance function (e.g., MSE as a function of filter coefficients) of an IIR filter, usually, has many local minima points.

The problem of instability is usually dealt with by checking the filter coefficients after each adaptation step and limiting them to the range that results in a stable transfer function. This, in general, is a difficult job and adds additional complexity, which in many cases becomes significant when the filter order is large. This additional complexity tends to nullify the computational advantage provided by the recursive nature of these filters.

Because of the multimodal nature of their performance surfaces, convergence of the IIR adaptive filters to their global minima is not guaranteed. The following approaches are usually used to deal with this problem:

(i) Local minima are usually observed when the criterion used to adjust the filter coefficients is MSE. A modification to this criterion leads to quadratic performance surfaces similar to those of FIR filters, thereby eliminating the problem of local minima. This modification results in a special implementation of IIR adaptive filters known as the *equation error method*. The details of this method are discussed in Section 10.2. In contrast, the conventional formulation of IIR adaptive filters based on the Wiener filter theory, which may suffer from the problem of local minima, is referred to as the *output error method*. This is discussed in Section 10.1.

Adaptive Filters: Theory and Applications, Second Edition. Behrouz Farhang-Boroujeny.

Table 10.1 Summary of the output error LMS algorithm.

Input:	Tap-weight vector, $\mathbf{w}(n) = [a_0(n)\ a_1(n) \cdots a_N(n)\ b_1(n) \cdots b_M(n)]^{\mathrm{T}}$, Input vector, $\mathbf{u}(n) = [x(n)\ x(n-1) \cdots x(n-N)\ y(n-1) \cdots y(n-M)]^{\mathrm{T}}$, the previous samples of $\alpha_0(n)$ and $\beta_1(n)$, and Desired output, $d(n)$.
Output:	Filter output, $y(n)$, Tap-weight vector update, $\mathbf{w}(n+1)$, and the samples of $\alpha_0(n)$ and $\beta_1(n)$ for next iteration.

1. Filtering:
   $y(n) = \mathbf{w}^{\mathrm{T}}(n)\mathbf{u}(n)$
2. Error estimation:
   $e(n) = d(n) - y(n)$
3. $\boldsymbol{\eta}(n)$ update:
   $\alpha_0(n) = x(n) + \sum_{l=1}^{M} b_l(n)\alpha_0(n-l)$
   $\beta_1(n) = y(n) + \sum_{l=1}^{M} b_l(n)\beta_1(n-l)$
   $\boldsymbol{\eta}(n) = [\alpha_0(n)\ \alpha_0(n-1) \cdots \alpha_0(n-N)\ \beta_1(n) \cdots \beta_1(n-M)]^{\mathrm{T}}$
4. Tap-weight vector adaptation:
   $\mathbf{w}(n+1) = \mathbf{w}(n) + 2\mu e(n)\boldsymbol{\eta}(n)$

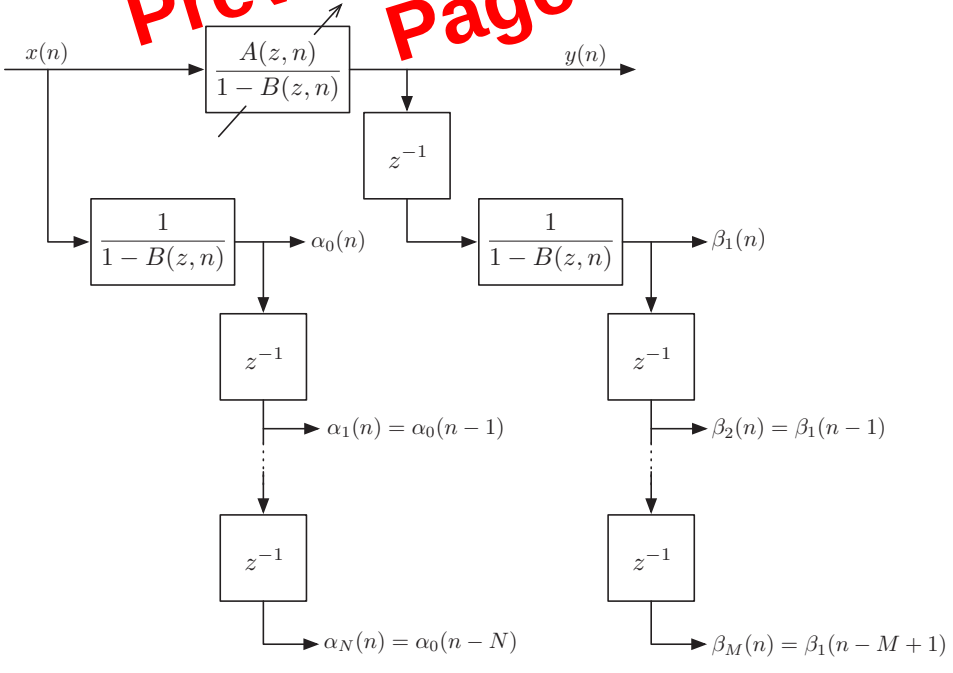

Figure 10.3 Simplified implementation of IIR adaptive filter using the output error adaptation method.

$$= \frac{1}{2\pi j} \oint \frac{dz}{z - 0.5}$$
$$= \text{residue of } \frac{1}{z - 0.5} \text{ at } z = 0.5$$
$$= 1$$

Similarly, we also obtain

$$E[d^2(n)] = \frac{4}{3} + \sigma_o^2, \quad E[x(n)d(n-1)] = E[d(n-1)x(n)] = 0$$

and

$$E[d(n)d(n-1)] = \frac{2}{3}$$

Substituting these results in Eq. (10.34) and solving for $\mathbf{w}$, we obtain

$$a_{0,e} = 1 \quad \text{and} \quad b_{1,e} = \frac{1}{2} \cdot \frac{1}{1 + 3\sigma_o^2/4}$$

where the subscript "e" signifies that the solutions correspond to the equation error method. We note that, in this particular case, $a_{0,e}$ is unbiased, that is, it is equal to its optimum value. However, $b_{1,e}$ is different from its optimum value in the Wiener filter. The amount of bias in $b_{1,e}$ is

$$b_{1,e} - b_{1,o} = \frac{1}{2} \cdot \frac{3\sigma_o^2/4}{1 + 3\sigma_o^2/4}$$

This bias is negligible when σ_o^2 is small. However, it becomes significant as σ_o^2 increases.

Further study of this example shows that when $x(n)$ is colored (nonwhite), both $a_{0,e}$ and $b_{1,e}$ are biased and the amount of bias, as we expect, increases with σ_o^2. This is left as an exercise for the reader (see Problem P10.2).

10.3 Case Study I: IIR Adaptive Line Enhancement

As was noted earlier in this chapter, adaptive line enhancement is a special problem that can be best solved by using IIR filters. In this section, we consider a special second-order IIR transfer function that was first proposed by David *et al.* (1983) and subsequently used and developed further by the same authors and others Ahmed *et al*. (1984); Cupo and Gitlin (1989); Hush *et al*. (1986); Regalia (1991); Cho and Lee (1993), and Farhang-Boroujeny and Wang (1997).

Figure 10.5 depicts the block diagram of the adaptive line enhancer (ALE) that we wish to study in this section. Here, $W(z)$ is an IIR filter with the transfer function

$$W(z) = \frac{(1-s)(w - z^{-1})}{1 - (1+s)wz^{-1} + sz^{-2}} \tag{10.35}$$

This is a narrow-band filter that may be used to extract a portion of the spectrum of the input, $x(n)$. When $x(n)$ is the sum of a narrow-band and a wide-band processes and $W(z)$ is centered around the narrow-band part of $x(n)$, the output of $W(z)$ will contain mainly

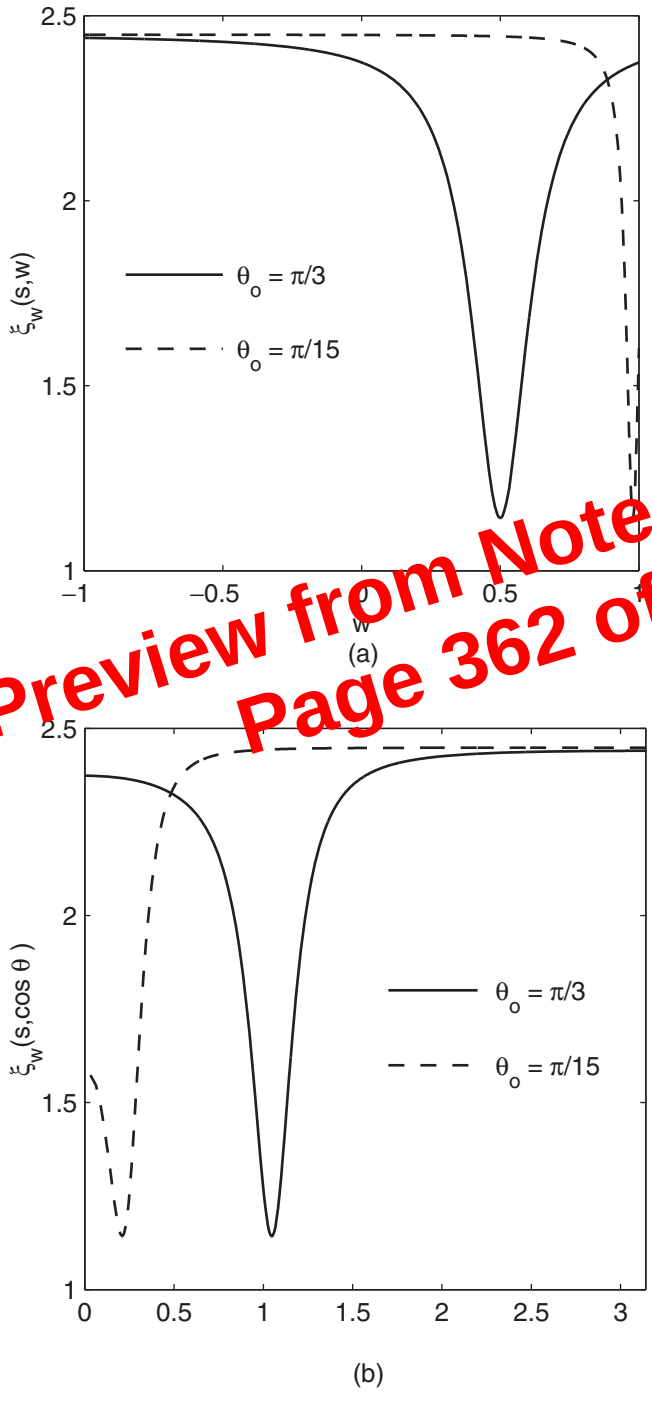

Figure 10.8 (a) Plots showing the variation of the performance function $\xi_w(s, w)$ as θ_o approaches 0 and (b) plots showing reduced sensitivity of the performance function $\xi_w(s, \cos\theta)$ to variations in θ_o.

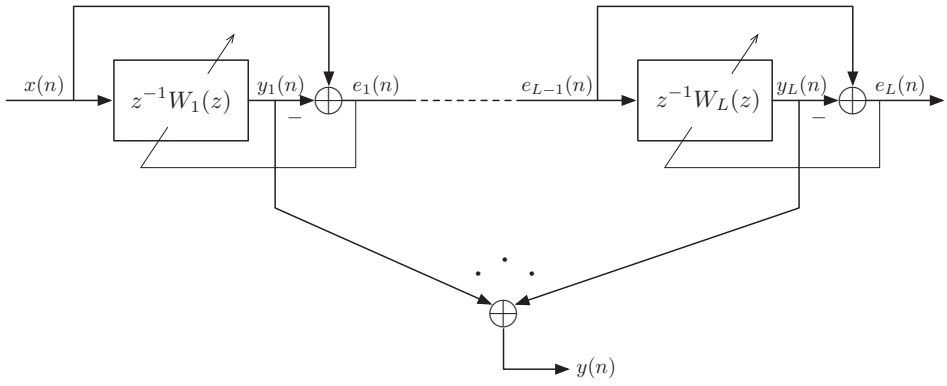

Figure 10.10 Cascaded IIR ALE for enhancement of multiple sinusoids buried in white noise.

Figure 10.11 illustrates the performance of the cascaded IIR ALE when Algorithm 2 is used. The input signal consists of the sum of four sinusoids in additive white Gaussian noise, and is given by

$$x(n) = \sin(\omega_1 n + \phi_1) + 2\sin(\omega_2 n + \phi_2) + 0.25\sin(\omega_3 n + \phi_3) + 0.5\sin(\omega_4 n + \phi_4) + \nu(n)$$

where ω_1, ω_2, ω_3, and ω_4 are equal to $\pi/1.8$, $\pi/3.5$, $\pi/6$, and $\pi/12$, respectively, ϕ_1 to ϕ_4 are random phases that are selected in the beginning of each simulation trial and remain fixed during that trial, and $\sigma_\nu^2 = 0.25$. This value corresponds to SNRs of 3, 9, -9, and -3 dB, respectively, for the individual sinusoids. As the energy of the input signals to successive stages of the ALE are different, the step-sizes of each stage are normalized to the energy of the input signal to that stage. The equations used for this purpose are $\mu_{s,i} = 0.0005/\hat{\sigma}_{x_i}^2$ and $\mu_{\theta,i}(n) = 0.01(1 - s(n))^3/\hat{\sigma}_{x_i}^2$. In these equations, i refers to the stage number, and $\hat{\sigma}_{x_i}^2$ is an estimate of the energy of the input signal, $x_i(n)$, to the ith stage. The following recursive equation is used for estimation of $\hat{\sigma}_{x_i}^2$

$$\hat{\sigma}_{x_i}^2(n) = 0.98\hat{\sigma}_{x_i}^2(n-1) + 0.02x_i^2(n).$$

The parameters $s_i(n)$'s are allowed to change between 0.25 and 0.9, and the threshold level used for activating or deactivating the adaptation of the following stages is set at 0.85. The results in Figure 10.11 show that this mechanism works very well. It may also be noted that the first stage is tuned to the strongest sinusoid (ω_2), and the last stage is tuned to the weakest one (ω_3). Such observation is intuitively sound. The MATLAB programs used to generate the results of this section are available on an accompanying website. The reader is encouraged to examine these programs and run further simulations to learn more about the line enhancer as well as the difficulties that one may encounter in using IIR adaptive filters. It would be also interesting to compare the behavior of FIR and IIR line enhancers. This is left as an exercise for interested readers.

matches the desired target response as close as possible. In particular, we are interested in matching the combined response of the channel and equalizer, that is,

$$\eta_{\mathrm{a}}(t) = \int_{-\infty}^{\infty} w_{\mathrm{a}}(\tau) h_{\mathrm{a}}(t-\tau)\mathrm{d}\tau \tag{10.48}$$

with the target response at sampling instants separated by the bit interval, T. As was noted in Chapter 1 (Section 1.6.2), the target response in magnetic channels is usually one of the class-IV partial responses characterized by the transfer functions

$$\Gamma(z) = z^{-\Delta}(1+z^{-1})^K(1-z^{-1}) \tag{10.49}$$

where z^{-1} represents one-bit delay, Δ is a parameter that takes care of the delays introduced by the channel and equalizer, and K is an integer greater than or equal to 1. The choice of K depends on the recording density, D. The value of K also determines the complexity of the detector. The commonly used values of K are 1, 2, and 3.

Next, we go through a sequence of discussions that lead us to a design methodology, using the results of this chapter as well as the previous chapters, for designing analog equalizers in the application of magnetic recording channels.

10.4.1 Channel Discretization

As all of the derivations in this book are based on sampled signals, we would like to replace the continuous time channel and equalizer impulse responses, $h_{\mathrm{a}}(t)$ and $w_{\mathrm{a}}(t)$, respectively, by their associated discrete-time counterparts. Define the sequences

$$h_i = h_{\mathrm{a}}(iT_{\mathrm{s}}) \quad \text{and} \quad w_i = w_{\mathrm{a}}(iT_{\mathrm{s}})$$

where T_{s} is the sampling period. When T_{s} is sufficiently small, we obtain, from Eq. (10.48)

$$\eta_i = \eta_{\mathrm{a}}(iT_{\mathrm{s}}) \approx T_{\mathrm{s}} \cdot (h_i \star w_i) \tag{10.50}$$

where $\star$ denotes convolution. The identity (10.50) follows from Eq. (10.48) by setting $t = iT_{\mathrm{s}}$ and approximating the integration on the right-hand side of Eq. (10.48) by a summation.

We note that the accuracy of the approximation used in Eq. (10.50) depends on the value of T_{s}. In practice, T_{s} has to be selected a few times smaller than the bit interval, T, for the results to be reasonably accurate. In the design procedure that we develop here we select T_{s} so that $T = LT_{\mathrm{s}}$, where L is an integer greater than 1. We call L the oversampling factor. Reasonable values of L are in the range of 4 to 10.

In the rest of our discussion, we assume that the time scale is normalized so that $T_{\mathrm{s}} = 1$. We also ignore the nonexactness of Eq. (10.50) and thus obtain

$$\eta_i = h_i \star w_i \tag{10.51}$$

as the discrete-time combined response of the channel and equalizer.

P10.10 Study the transfer function between the input, $x(n)$, and output, $y(n)$, of Figure 10.10 and show that this is that of a filter with L narrow bands.

P10.11 Study the transfer function between the input, $x(n)$, and output error, $e_L(n)$, of Figure 10.10 and show that this is that of a filter with L notches.

P10.12 In line enhancers, signal enhancement is defined as the ratio of the SNR at the enhancer output, to the SNR at its input. For the IIR ALE that was discussed in Section 10.3 show that when $x(n)$ is given by Eq. (10.39)

$$\text{Signal enhancement of IIR ALE} = \frac{1+s}{1-s}$$

P10.13 Work out a detailed derivation of Eq. (10.59).

P10.14 For the magnetic recording channel, which is discussed in Section 10.4, show that the mean-squared value of the sum of residual ISI and noise at the equalizer output is $\sum_i (\eta_{iL} - \gamma_i)^2 + \sigma_v^2 \sum_i w_i^2$. Thus, justify the use of definition Eq. (10.71).

P10.15 From our discussion in Section 10.2, we recall that the output error, $e(n)$, and equation error, $e'(n)$, are related through the transfer function $1 - B(z)$. Assuming that a relatively good estimate of $B(z)$ (say, $\hat{B}(z)$) is available, it is proposed that the setup of Figure P10.15 may be used to obtain better estimates of $A(z)$ and $B(z)$ compared with what could be achieved by the original equation error setup of Figure 10.4. Elaborate on this diagram and explain why this setup may give better estimates of $A(z)$ and $B(z)$, as compared with that in Figure 10.4. Also, develop an LMS algorithm for adaptation of $A(z)$ and $B(z)$ in this setup.

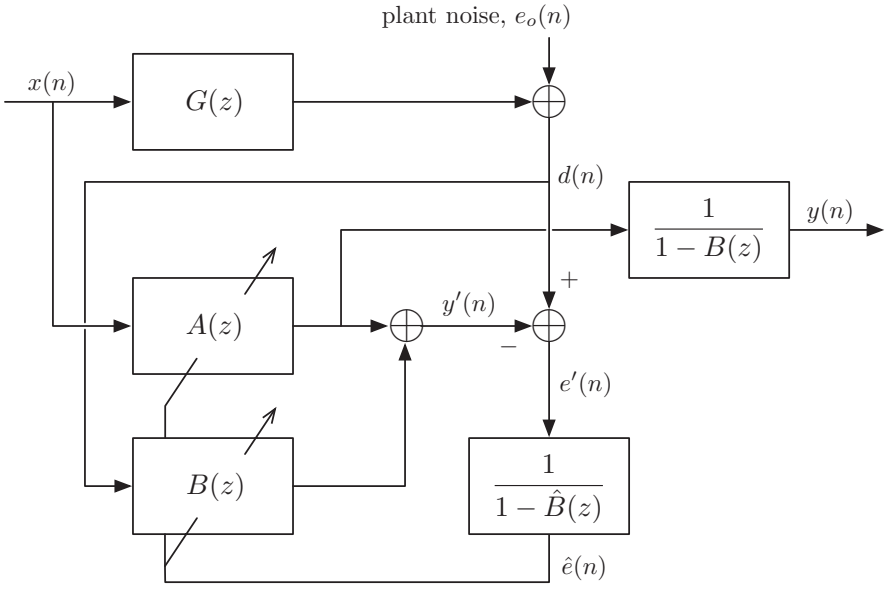

Figure P10.15

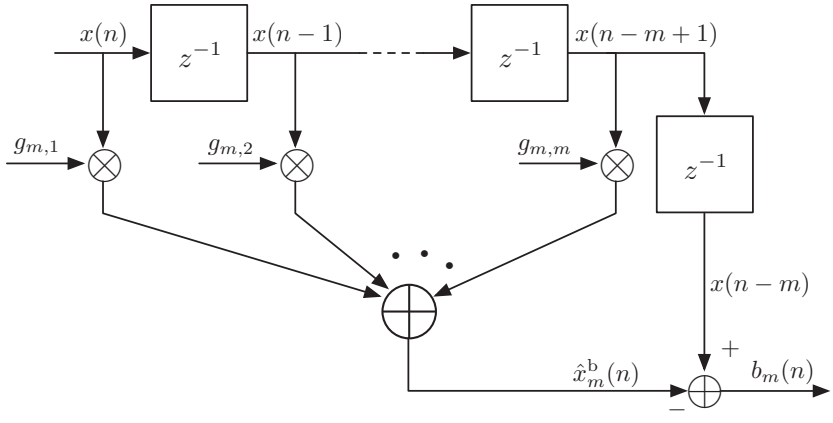

Figure 11.2 Backward linear predictor.

is the backward prediction error and

$$\hat{x}_m^{\mathrm{b}}(n) = \sum_{i=1}^{m} g_{m,i}\, x(n-i+1) = \mathbf{g}_m^{\mathrm{T}} \mathbf{x}_m(n) \tag{11.12}$$

is the mth-order backward prediction of the input sample $x(n-m)$. This is a conventional Wiener filtering problem with the input vector $\mathbf{x}_m(n)$ and desired output $x(n-m)$. Hence, the corresponding Wiener–Hopf equation is obtained by direct substitution of $x(n-m)$ for $d(n)$ and $\mathbf{x}_m(n)$ for $\mathbf{x}(n)$ in Eqs. (3.10) and (3.11), and recalling Eq. (3.24). The result is

$$\mathbf{R}\mathbf{g}_m = \mathbf{r}_{\mathrm{b}} \tag{11.13}$$

where $\mathbf{R} = E[\mathbf{x}_m(n)\mathbf{x}_m^{\mathrm{T}}(n)]$, $\mathbf{r}_{\mathrm{b}} = E[x(n-m)\mathbf{x}_m(n)]$.

Since $x(n)$ is stationary, the correlation matrix $\mathbf{R}$ in Eq. (11.13) is the same matrix as that in Eq. (11.5). However, the vector $\mathbf{r}_{\mathrm{b}}$ on the right-hand side of Eq. (11.13) is different from the vector $\mathbf{r}$ in Eq. (11.5). Using the definition (11.7), we obtain

$$\mathbf{r}_{\mathrm{b}} = \begin{bmatrix} r(m) \\ r(m-1) \\ \vdots \\ r(1) \end{bmatrix} \tag{11.14}$$

Comparing Eqs. (11.14) and (11.8), we note that $\mathbf{r}_{\mathrm{b}}$ is same as the vector $\mathbf{r}$ with its elements arranged in the reverse order.

When the tap weights of the backward predictor are optimized according to Eq. (11.13),

$$\begin{aligned} P_m^{\mathrm{b}} &= E[x^2(n-m)] - \mathbf{r}_{\mathrm{b}}^{\mathrm{T}}\mathbf{g}_m \\ &= E[x^2(n-m)] - \mathbf{r}_{\mathrm{b}}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{r}_{\mathrm{b}} \end{aligned} \tag{11.15}$$

11.3 Relationship Between Forward and Backward Predictors

We now show that there is a close relationship between the tap-weight vectors of the forward and backward linear predictors of a process $x(n)$. To see this, we substitute Eqs. (11.8) and (11.9) in Eq. (11.5) and write the result in scalar form as

$$\sum_{i=1}^{m} r(i-j)a_{m,i} = r(j), \quad \text{for } j = 1, 2, \ldots, m \tag{11.16}$$

where we have used the property $r(i-j) = r(j-i)$. Also, substitution of Eqs. (11.8) and (11.14) in to Eq. (11.13) gives

$$\sum_{i=1}^{m} r(i-j)g_{m,i} = r(m+1-j), \quad \text{for } j = 1, 2, \ldots, m \tag{11.17}$$

Next, we let $i = m+1-k$ and $j = m+1-l$ in Eq. (11.17) and use $r(k-l) = r(l-k)$ to obtain

$$\sum_{k=1}^{m} r(k-l)g_{m,m+1-k} = r(l), \quad \text{for } l = 1, 2, \ldots, m \tag{11.18}$$

Replacing k and l in Eq. (11.18) by i and j, respectively, and comparing the result with Eq. (11.16), we get

$$a_{m,i} = g_{m,m+1-i}, \quad \text{for } i = 1, 2, \ldots, m \tag{11.19}$$

or

$$g_{m,i} = a_{m,m+1-i}, \quad \text{for } i = 1, 2, \ldots, m \tag{11.20}$$

This result shows that *the optimum tap weights of the mth-order forward predictor of a wide sense stationary process $x(n)$ are the same as the optimum tap weights of the corresponding backward predictor, but in the reverse order*. Thus, we may write

$$f_m(n) = x(n) - \sum_{i=1}^{m} a_{m,i}x(n-i) \tag{11.21}$$

and

$$b_m(n) = x(n-m) - \sum_{i=1}^{m} a_{m,m+1-i}x(n-i+1) \tag{11.22}$$

11.4 Prediction-Error Filters

The forward predictor of Figure 11.1 uses an m-tap transversal filter to get an estimate of the present sample $x(n)$ of a sequence based on its past m samples $x(n-1), x(n-2), \ldots, x(n-m)$. The mth-order forward prediction-error filter for a sequence $x(n)$ is defined as the filter whose input is $x(n)$ and the forward prediction error $f_m(n)$ is its output. Figure 11.3 depicts a block schematic diagram showing how forward predictor and forward prediction-error filter are related.

11.6 Derivation of Lattice Structure

In this section, we present a derivation of lattice structure for prediction-error filters. A distinct feature of lattice structure, as we will show in this section, is that it is a direct implementation of the *order-update equations* for computing mth-order forward and backward prediction errors from the forward and backward prediction errors of order $m-1$. This is not possible in the transversal structure case. To derive these order-update equations and thereby the structure of lattice filters, we start with the forward prediction error for an $(m+1)$th-order predictor:

$$f_{m+1}(n) = x(n) - \sum_{i=1}^{m+1} a_{m+1,i}\, x(n-i) \tag{11.30}$$

The summation on the right-hand side of Eq. (11.30) can be rearranged as

$$\sum_{i=1}^{m+1} a_{m+1,i}\, x(n-i) = \sum_{i=1}^{m} a_{m+1,i}\, x(n-i) + a_{m+1,m+1}\, x(n-m-1) \tag{11.31}$$

From Eq. (11.22), we get

$$x(n-m-1) = b_m(n-1) + \sum_{i=1}^{m} a_{m,m+1-i}\, x(n-i) \tag{11.32}$$

Substituting Eq. (11.32) in Eq. (11.31), we obtain

$$\begin{aligned}\sum_{i=1}^{m+1} a_{m+1,i}\, x(n-i) &= \sum_{i=1}^{m} (a_{m+1,i} + a_{m+1,m+1} a_{m,m+1-i})\, x(n-i) \\ &\quad + a_{m+1,m+1}\, b_m(n-1) \\ &= \sum_{i=1}^{m} a'_{m,i}\, x(n-i) + \kappa_{m+1}\, b_m(n-1)\end{aligned} \tag{11.33}$$

where

$$\kappa_{m+1} = a_{m+1,m+1} \tag{11.34}$$

and

$$a'_{m,i} = a_{m+1,i} + \kappa_{m+1}\, a_{m,m+1-i}, \quad \text{for } i = 1, 2, \ldots, m \tag{11.35}$$

The above development shows that any linear combination of the input samples $x(n-1), x(n-2), \ldots, x(n-m-1)$ can also be obtained as a linear combination of $x(n-1), x(n-2), \ldots, x(n-m)$ and $b_m(n-1)$. We also note that the summation on the left-hand side of Eq. (11.33) is the $(m+1)$th-order forward prediction of $x(n)$, that is, $\hat{x}^{\mathrm{f}}_{m+1}(n)$. We may thus argue that the estimate $\hat{x}^{\mathrm{f}}_{m+1}(n)$ can also be obtained as a linear combination of the past m samples of $x(n)$ and the backward prediction error $b_m(n-1)$, that is,

$$\hat{x}^{\mathrm{f}}_{m+1}(n) = \sum_{i=1}^{m} a'_{m,i}\, x(n-i) + \kappa_{m+1}\, b_m(n-1) \tag{11.36}$$

where the last equality follows from the identity $E[f_m(n)x(n-i)] = 0$, for $i = 1, 2, \ldots, m$. Substituting Eq. (11.21) in Eq. (11.88), we obtain

$$\kappa_{m+1} P_m = E\left[\left(x(n) - \sum_{i=1}^{m} a_{m,i} x(n-i)\right) x(n-m-1)\right]$$

$$= r(m+1) - \sum_{i=1}^{m} a_{m,i} r(m+1-i) \tag{11.89}$$

or

$$\kappa_{m+1} = \frac{r(m+1) - \sum_{i=1}^{m} a_{m,i} r(m+1-i)}{P_m} \tag{11.90}$$

Thus, given the autocorrelation coefficients $r(1), r(2), \ldots, r(m+1)$, the mth-order transversal predictor coefficients $a_{m,i}$'s, and P_m, one can calculate κ_{m+1} according to Eq. (11.90). The order-update equations (11.84) are then used to obtain the coefficients of the $(m+1)$th-order transversal predictor, $a_{m+1,i}$'s. Equation (11.66) is used to obtain P_{m+1} for the next iteration. Table 11.3 summarizes these results. This is called *Levinson–Durbin algorithm*. The most important feature of the Levinson–Durbin algorithm is its computational efficiency. Careful examination of Table 11.3 shows that the implementation of Levinson–Durbin algorithm requires about M^2 multiplications/divisions and the same number of additions/subtractions, where M is the order of predictor. This must be compared with M^3 which is the order of computations required for solving a system of M linear equations without exploiting the structure in the system. The special structure that is exploited here is the symmetric Toeplitz nature of the autocorrelation matrix $\mathbf{R}$. By definition, the autocorrelation matrix of any process (stationary or nonstationary) is symmetric. But, if the process $x(n)$ is stationary (at least wide sense), then the autocorrelation matrix becomes Toeplitz, in addition to being symmetric. That is, all the elements along any given diagonal are the same. For example, the kth subdiagonal and super-diagonal will be constituted by the autocorrelation at lag k.

Table 11.3 Levinson–Durbin algorithm.

Given: $r(0), r(1), \ldots, r(M)$
Required: $a_{M,1}, a_{M,2}, \ldots, a_{M,M}$ and $\kappa_1, \kappa_2, \ldots, \kappa_M$

$P_0 = r(0)$
$\kappa_1 = r(1)/P_0$
$a_{1,1} = \kappa_1$
$P_1 = (1 - \kappa_1^2) P_0$
for $m = 1$ to $M - 1$
  $\kappa_{m+1} = \frac{r(m+1) - \sum_{i=1}^{m} a_{m,i} r(m+1-i)}{P_m}$
  $a_{m+1,i} = a_{m,i} - \kappa_{m+1} a_{m,m+1-i}$, for $i = 1, 2, \ldots, m$
  $a_{m+1,m+1} = \kappa_{m+1}$
  $P_{m+1} = (1 - \kappa_{m+1}^2) P_m$
end

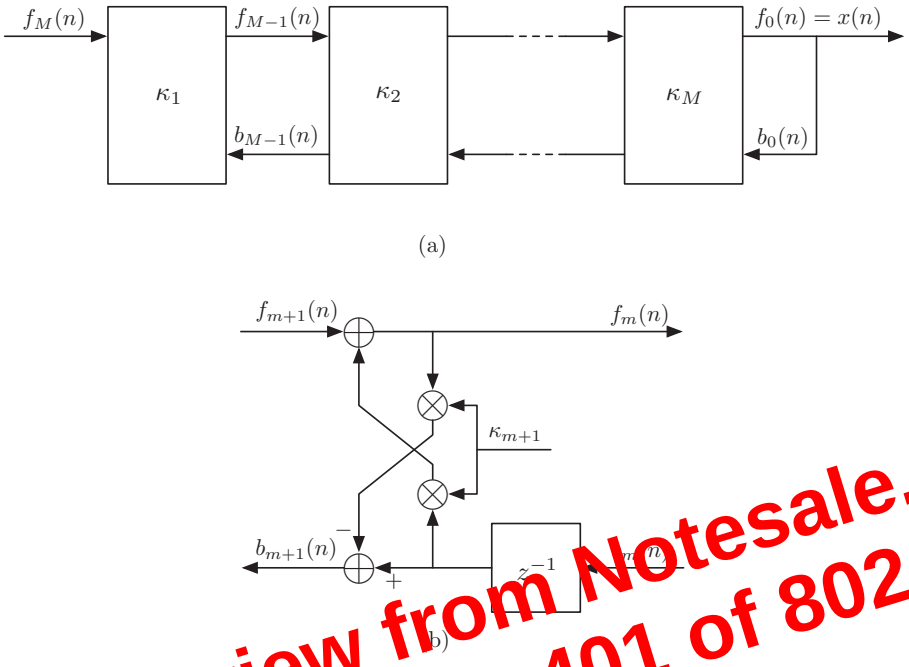

Figure 11.9 Lattice all-pole filter: (a) overall structure and (b) details of one stage.

We note that the denominator of $G(z)$ is assumed to be the system function of a prediction-error filter. This, as was noted in the last section, is not restrictive, as this condition only limits the poles of $G(z)$ to remain within the unit circle in the z-plane. In other words, the condition imposed on $G(z)$ is just to guarantee its stability.

To develop a lattice structure for $G(z)$, we first rearrange Eq. (11.75) as

$$\mathbf{w} = \mathbf{L}^{\mathrm{T}}\mathbf{c} \tag{11.100}$$

Next, we define $\mathbf{z} = [1 \quad z^{-1} \quad z^{-2} \cdots z^{-(N-1)}]^{\mathrm{T}}$, where z is the z-domain complex variable, multiply the transpose of both sides of Eq. (11.100) from right by $\mathbf{z}$ and replace $N-1$ by M, to obtain

$$W(z) = \sum_{i=0}^{M} c_i H_{b_i}(z) \tag{11.101}$$

where c_i's are the elements of vector $\mathbf{c}$,

$$W(z) = \sum_{i=0}^{M} w_i z^{-i} \tag{11.102}$$

and $H_{b_i}(z)$ is defined as in Eq. (11.80). Equation (11.101) shows that any arbitrary order M FIR system function $W(z)$ can equivalently be realized as a linear combination of the backward prediction-error filter system functions $H_{b_0}(z), H_{b_1}(z), \ldots, H_{b_M}(z)$.

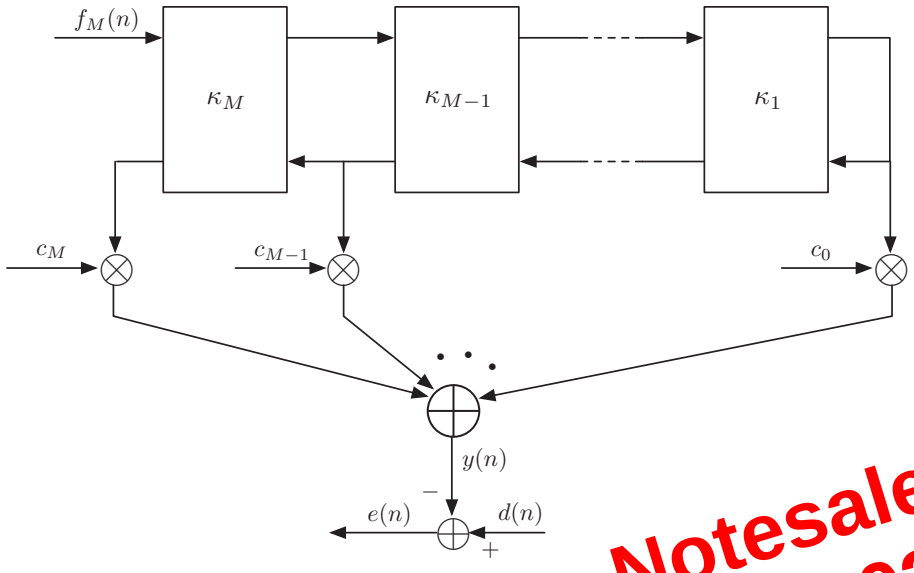

Figure 11.9 Pole-zero lattice for an arbitrary system function.

Using Eqs. (11.101) and (11.102), we obtain

$$G(z) = \sum_{i=0}^{M} c_i \frac{H_{b_i}(z)}{H_{f_M}(z)} \tag{11.103}$$

Furthermore, with reference to Figure 11.8a, we note that $H_{b_i}(z)/H_{f_M}(z)$ is the transfer function relating $f_M(n)$ and $b_i(n)$. It is obtained as the cascade of the transfer function between $f_M(n)$ and $x(n)$, that is, $1/H_{f_M}(z)$, and the transfer function between $x(n)$ and $b_i(n)$, that is, $H_{b_i}(z)$. Using these results, we obtain Figure 11.9 as lattice realization of the system function $G(z)$.

11.13 Adaptive Lattice Filter

In this section, an LMS algorithm for adaptive adjustment of the parameters of the lattice joint process estimator, given in Figure 11.7, is developed. Simulation results and some discussions on the performance of adaptive lattice filters are provided in the next section.

As the prediction error power becomes minimum when the predictor coefficients are chosen optimally, the optimum PARCOR coefficient κ_m of the mth stage of a lattice predictor is obtained by minimizing the cost function

$$\xi_{p,m} = E[f_m^2(n) + b_m^2(n)] \tag{11.104}$$

The cost function $\xi_{p,m}$ is equivalent to either of the cost functions $P_m^{\mathrm{f}} = E[f_m^2(n)]$ and $P_m^b = E[b_m^2(n)]$, since forward and backward predictors of the same order share the same set of coefficients and also the same level of minimum mean-squared error. By defining

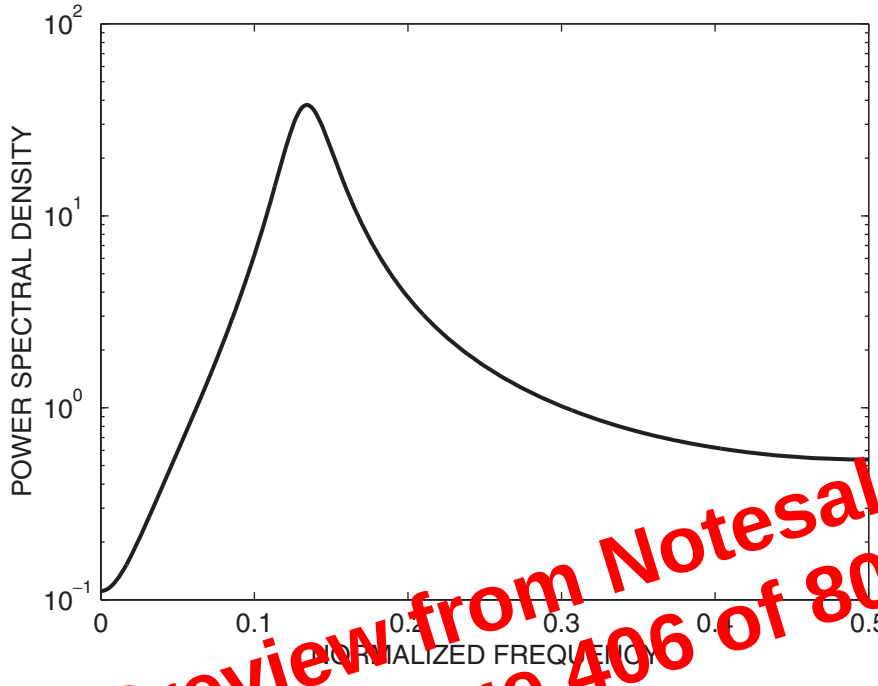

Figure 11.11 Power spectral density of the input process for the modeling problem simulations.

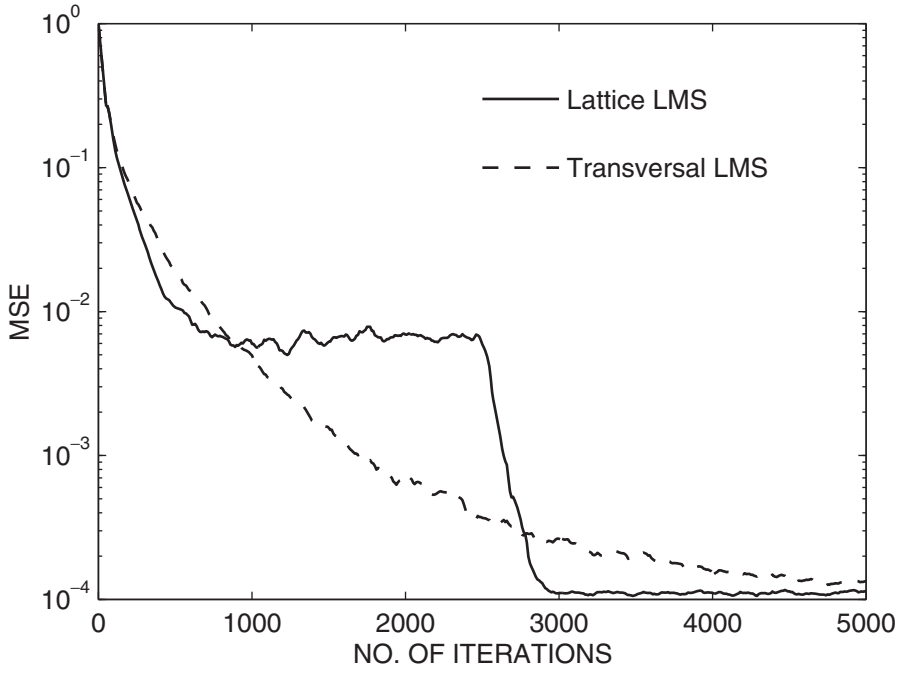

Figure 11.12 Learning curves showing convergence of the transversal and lattice LMS applied to modeling problem of Figure 11.10.

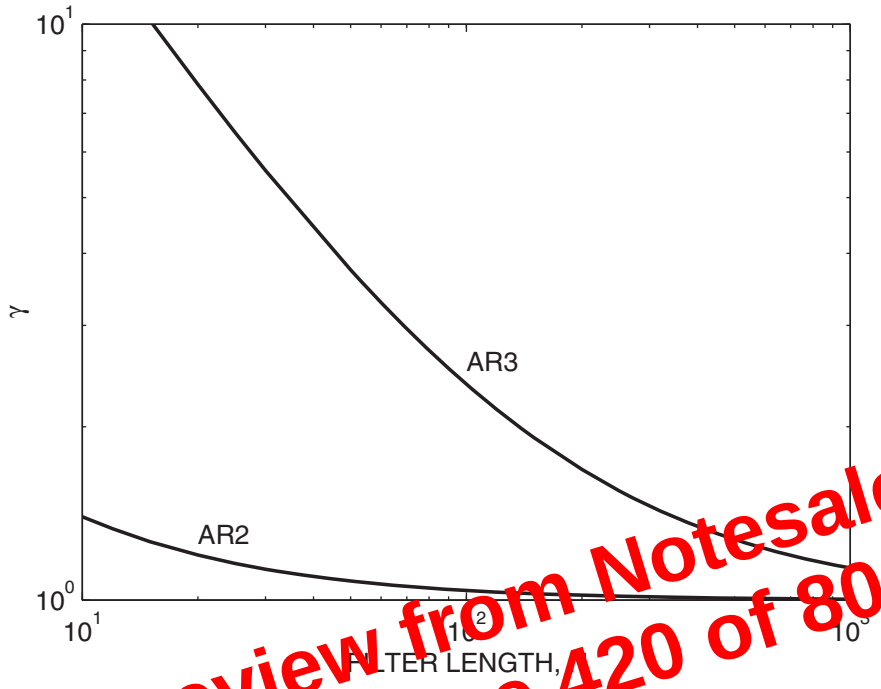

Figure 11.15 Variation of the parameter γ as a function of filter length for $x_2(n)$ (AR2) and $x_3(n)$ (AR3).

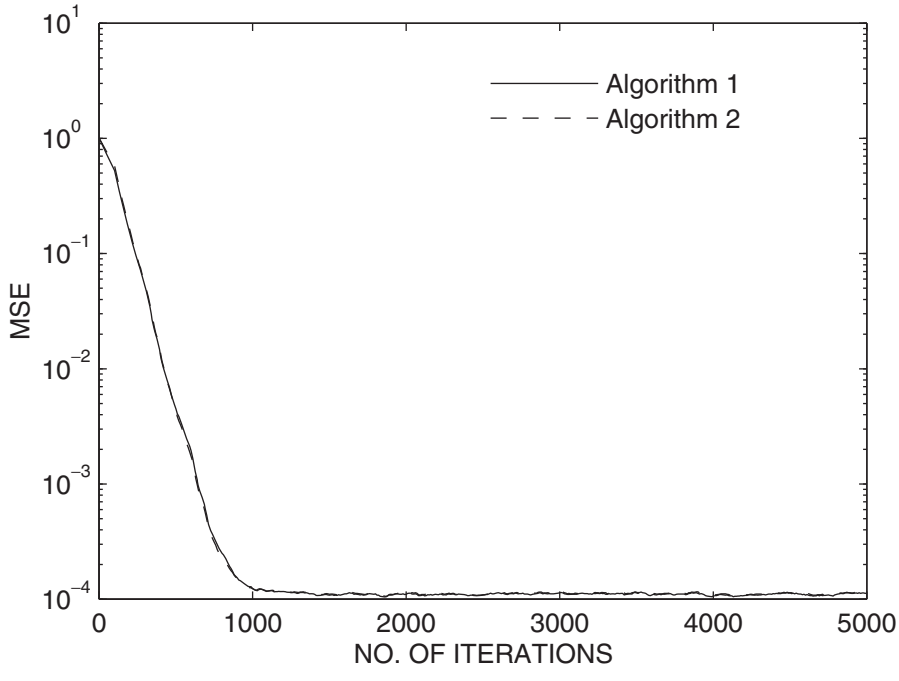

Figure 11.16 MSE versus iteration number for $x_2(n)$, $N = 30$ and with the AR model of input assumed known.

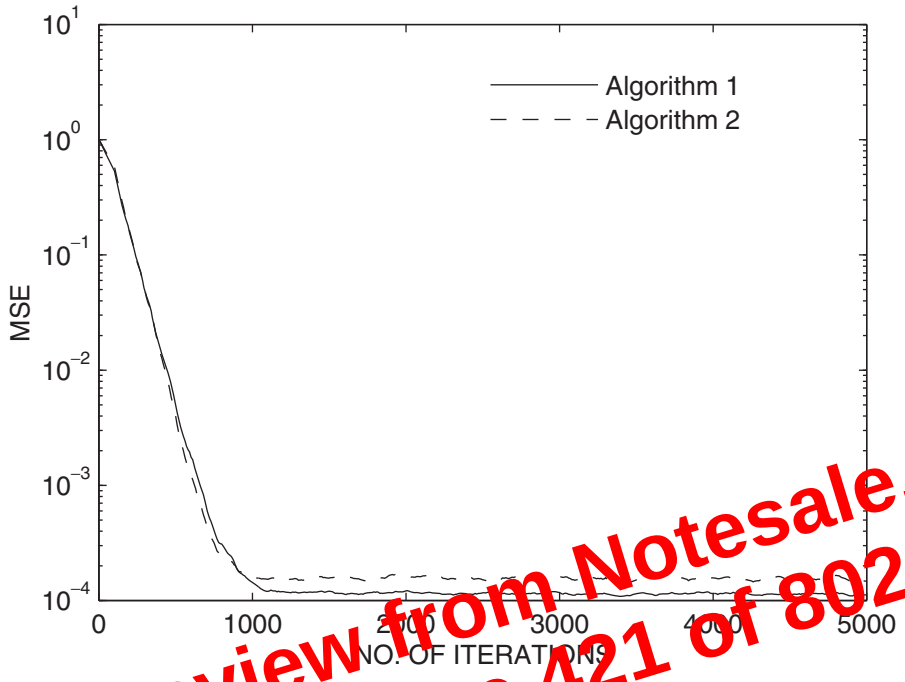

Figure 11.17 MSE versus iteration number for $x_3(n)$, $N = 30$ and with the AR model of input assumed known.

both algorithms should approach to about the same misadjustment, in the case of $x_2(n)$. However, their performance may be significantly different, in the case of $x_3(n)$. To be more exact, from the data used for generating Figure 11.15, we have $\gamma = 1.13$, for $x_2(n)$, and $\gamma = 5.57$, for $x_3(n)$, for $N = 30$. Using these and Eqs. (11.132) and (11.148), we obtain the following:

For $x_2(n)$: $\frac{\mathcal{M}_2}{\mathcal{M}_1} = 1.147.$

For $x_3(n)$: $\frac{\mathcal{M}_2}{\mathcal{M}_1} = 11.32.$

Careful examination of the numerical values which have been obtained by simulations show that for the $x_2(n)$ process $\frac{\mathcal{M}_2}{\mathcal{M}_1} = 1.152$. This matches well with the above ratio. However, for the $x_3(n)$ process the simulation results give $\frac{\mathcal{M}_2}{\mathcal{M}_1} = 3.85$. This which does not match the above theoretical ratio, may be explained as follows. Careful examination of the numerical results in simulations reveals that there are only a few terms in $\mathbf{u}_a(n)$ that have a major effect on the degradation of Algorithm 2, when compared with Algorithm 1. These terms, which greatly disturb the first and last few elements of the tap-weight vector $\mathbf{w}(n)$, are so large that their contribution violates the independence assumption 2 of the previous section. As a result, the theoretical derivation that led to Eq. (11.148) may not be valid unless the step-size parameter μ is set to a very small value so that the latter assumption could be justified. Nevertheless, the developed theory is able to predict

P11.26 Consider the difference equation (11.114) and note that it may be written as

$$x(n) = \mathbf{x}_M^{\mathrm{T}}(n-1)\mathbf{h} + v(n) \qquad \text{(P11.26.1)}$$

where $\mathbf{x}_M(n) = [x(n-1)\ x(n-2)\cdots x(n-M)]^{\mathrm{T}}$ and $\mathbf{h} = [h_1\ h_2\cdots h_M]^{\mathrm{T}}$.

(i) Starting with Eq. (P11.26.1) show that the vector $\mathbf{h}$ is related to the autocorrelation coefficients $r(0), r(1), \ldots, r(M)$ of $x(n)$ according to the equation

$$\mathbf{R}_M\mathbf{h} = \mathbf{r}_M$$

where

$$\mathbf{R}_M = \begin{bmatrix} r(0) & r(1) & \cdots & r(M-1) \\ r(1) & r(0) & \cdots & r(M-2) \\ \vdots & \vdots & \ddots & \vdots \\ r(M-1) & r(M-2) & \cdots & r(0) \end{bmatrix}$$

and

$$\mathbf{r}_M = [r(1)\ r(2)\cdots r(M)]^{\mathrm{T}}$$

(ii) Show that for any m

$$r(m) = \sum_{i=1}^{M} h_i r(m-i)$$

(iii) By combining the results of (i) and (ii) show that for any $M' > M$,

$$\mathbf{R}_{M'}\mathbf{h}' = \mathbf{r}_{M'}$$

where $\mathrm{h}' = \begin{bmatrix} \mathbf{h} \\ \mathbf{0} \end{bmatrix}$ and $\mathbf{0}$, here, is the length $M' - M$ zero column vector.

(iv) Use the above results to justify the validity of the results presented in Eqs. (11.115) and (11.116).

P11.27 Suggest a lattice structure for realization of the transfer function $W(z)$ of the IIR line enhancer of Section 10.3 and obtain its coefficients in terms of the parameters s and w.

P11.28 Give a detailed derivation of Eq. (11.123).

P11.29 Give a derivation of Eq. (11.147) from Eq. (11.146).

P11.30 Recall that the unconstrained PFBLMS algorithm of Chapter 8 converges very slowly when the partition length, M, and the block length, L, are equal. In Section 8.4, it was noted the slow convergence of PFBLMS algorithm can be improved by choosing M a few times larger than L. We also noticed that the frequency domain processing involved in the implementation of the PFBLMS algorithm may be viewed as a parallel bank of a number of transversal filters, each belonging to one of the frequency bins. Furthermore, when the number of frequency bins is large, these filters operate (converge) almost independent of one another. Noting these, the following alternative solution may be proposed to

Using Eqs. (12.6) and (12.10), we obtain

$$\frac{\partial \zeta(n)}{\partial w_i(n)} = 2\sum_{k=1}^{n} e_n(k)\frac{\partial e_n(k)}{\partial w_i(n)} \tag{12.18}$$

Using the identity

$$e_n(k) = d(k) - \sum_{i=0}^{N-1} w_i(n)x_i(k) \tag{12.19}$$

to evaluate the second factor on the right-hand side of Eq. (12.18), we get

$$\frac{\partial \zeta(n)}{\partial w_i(n)} = -2\sum_{k=1}^{n} e_n(k)x_i(k) \tag{12.20}$$

Furthermore, we note that when $\mathbf{w}(n) = \hat{\mathbf{w}}(n)$,

$$\frac{\partial \zeta(n)}{\partial w_i(n)} = 0, \quad \text{for} \quad i = 0, 1, \ldots, N-1 \tag{12.21}$$

Using Eqs. (12.20) and (12.21), we find that when $\mathbf{w}(n) = \hat{\mathbf{w}}(n)$, the following identities hold:

$$\sum_{k=1}^{n} \hat{e}_n(k)x_i(k) = 0, \quad \text{for} \quad i = 0, 1, \ldots, N-1 \tag{12.22}$$

where $\hat{e}_n(k)$ is the optimized estimation error in the least-squares sense. This result, which is equivalent to Eq. (12.17), is known as the *principle of orthogonality* in the least-squares formulation. We define the vectors

$$\hat{\mathbf{e}}(n) = [\hat{e}_n(1)\,\hat{e}_n(2)\cdots\hat{e}_n(n)]^{\mathrm{T}} \tag{12.23}$$

and

$$\mathbf{x}_i(n) = [x_i(1)\,x_i(2)\cdots x_i(n)]^{\mathrm{T}} \tag{12.24}$$

and note that Eq. (12.22) may also be expressed in terms of these vectors as

$$\hat{\mathbf{e}}^{\mathrm{T}}(n)\mathbf{x}_i(n) = 0, \quad \text{for} \quad i = 0, 1, \ldots, N-1 \tag{12.25}$$

We note that the left-hand side of Eq. (12.25) is the *inner product* of $\hat{\mathbf{e}}(n)$ and $\mathbf{x}_i(n)$, thus the name *principle of orthogonality*.

Comparison of Eqs. (12.17) and (12.25) reveals that the definition of orthogonality is in terms of statistical averages in Wiener filtering, whereas it is in terms of inner products of data vectors in the case of least-squares estimation. By dividing both sides of Eq. (12.22) by n, we see that we can also use time averages to define orthogonality in the least-squares case:

$$\frac{1}{n}\sum_{k=1}^{n} \hat{e}_n(k)x_i(k) = 0, \quad \text{for} \quad i = 0, 1, \ldots, N-1 \tag{12.26}$$

An immediate corollary to the principle of orthogonality is that *when the tap weights of a filter are optimized in the least-squares sense, the filter output and its optimized estimation error are orthogonal*. That is,

$$\hat{\mathbf{e}}^{\mathrm{T}}(n)\hat{\mathbf{y}}(n) = 0 \tag{12.27}$$

where $\hat{\mathbf{y}}(n)$ is the vector of the output samples of the filter when $\mathbf{w}(n) = \hat{\mathbf{w}}(n)$. This follows immediately if we note that Eq. (12.8) may also be written as

$$\mathbf{y}(n) = \sum_{i=0}^{N-1} w_i(n)\mathbf{x}_i(n) \tag{12.28}$$

and use the identity (12.25) to obtain Eq. (12.27).

Example 12.1

Consider the case where $n = 3$,

$$\mathbf{X}(3) = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 2 & 0.1 \end{bmatrix}$$

and

$$\mathbf{d}(3) = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

We wish to find $\hat{\mathbf{w}}(3)$, $\hat{\mathbf{y}}(3)$, and $\hat{\mathbf{e}}(3)$ and to confirm the principle of orthogonality. We have

$$\boldsymbol{\Psi}(3) = \mathbf{X}(3)\mathbf{X}^{\mathrm{T}}(3) = \begin{bmatrix} 5 & 4 \\ 4 & 5.01 \end{bmatrix}$$

and

$$\boldsymbol{\theta}(3) = \mathbf{X}(3)\mathbf{d}(3) = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

Thus,

$$\hat{\mathbf{w}}(3) = \begin{bmatrix} 5 & 4 \\ 4 & 5.01 \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \frac{1}{9.05} \begin{bmatrix} 5.01 & -4 \\ -4 & 5 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \frac{1}{9.05} \begin{bmatrix} 9.01 \\ -9 \end{bmatrix}$$

$$\begin{aligned} \hat{\mathbf{y}}(3) &= \hat{w}_0(3)\mathbf{x}_0(3) + \hat{w}_1(3)\mathbf{x}_1(3) \\ &= \frac{9.01}{9.05} \begin{bmatrix} 2 \\ 1 \\ 0 \end{bmatrix} - \frac{9}{9.05} \begin{bmatrix} 1 \\ 2 \\ 0.1 \end{bmatrix} = \frac{1}{9.05} \begin{bmatrix} 9.02 \\ -8.99 \\ -0.9 \end{bmatrix} \end{aligned}$$

and

$$\hat{\mathbf{e}}(3) = \mathbf{d}(3) - \hat{\mathbf{y}}(3) = \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix} - \frac{1}{9.05} \begin{bmatrix} 9.02 \\ -8.99 \\ -0.9 \end{bmatrix} = \frac{1}{9.05} \begin{bmatrix} 0.03 \\ -0.06 \\ 0.90 \end{bmatrix}$$

Substituting Eq. (12.92) in Eq. (12.93), we obtain

$$\mathcal{M}_{\text{RLS}} = \frac{1-\lambda}{1+\lambda} N \tag{12.94}$$

12.5.5 *Initial Transient Behavior of the RLS Algorithm*

Much of the benefit of the RLS algorithm is attributed to the fact that it shows very fast convergence when it is started from a rest condition with the initial values of $\mathbf{w}(0) = \mathbf{0}$ and $\boldsymbol{\Psi}^{-1}(0) = \delta^{-1}\mathbf{I}$. This fast convergence is observed only after the first N samples of the input and desired output sequences are processed. In typical implementations of the RLS algorithm, we always find that the MSE of adaptive filter converges to a level close to its minimum value within a small number of iterations (usually two to three times the filter length) and then it proceeds with a fine-tuning process that may last much longer before the MSE reaches its steady-state value. The initial transient behavior of the RLS algorithm can be best explained through a numerical example (computer simulation).

As a numerical example, here, we apply the RLS algorithm to the system modeling setup of Section 6.4.1. Thus, a comparison of the RLS and LMS algorithms can be made. Figure 12.1 presents the schematic diagram of the modeling setup. The common input, $x(n)$, to the plant, $W_o(z)$, and adaptive filter, $W(z)$, is obtained by passing a unit variance white Gaussian sequence, $\nu(n)$, through a filter with the system function $H(z)$. The plant noise, as before, is denoted by $e_o(n)$. It is assumed to be a white noise process independent of $x(n)$.

For our experiments, as in Section 6.4.1, we select $\sigma_o^2 = 0.001$ and

$$W_o(z) = \sum_{i=0}^{7} z^{-i} - \sum_{i=8}^{14} z^{-i} \tag{12.95}$$

The length of the adaptive filter, N, is chosen equal to the length of $W_o(z)$, that is, $N = 15$. Also, we present results of simulations for two choices of input that are characterized by

$$H(z) = H_1(z) = 0.35 + z^{-1} - 0.35z^{-2} \tag{12.96}$$

and

$$H(z) = H_2(z) = 0.35 + z^{-1} + 0.35z^{-2} \tag{12.97}$$

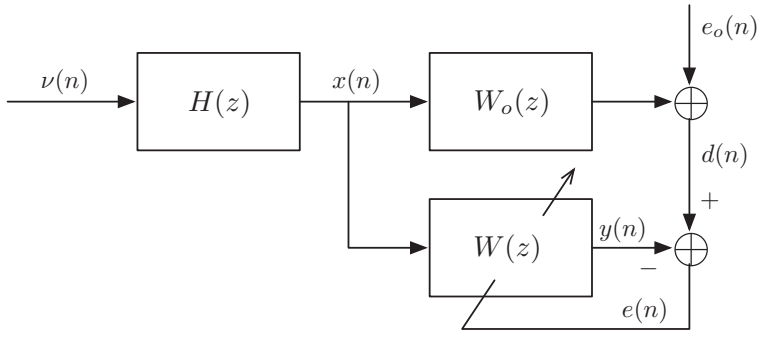

Figure 12.1 Adaptive modeling of an FIR plant.

2. Once the PARCOR coefficients are optimized, the backward prediction errors $b_{0,n}(k)$, $b_{1,n}(k)$, ..., $b_{N,n}(k)$ are orthogonal with one another, in the sense that

$$\sum_{k=1}^{n} \lambda^{n-k} b_{i,n}(k) b_{j,n}(k) = 0 \tag{13.25}$$

for any pair of unequal i and j in the range of 0 to $N-1$.

3. The regressor coefficients $c_0(n), c_1(n), \ldots, c_{N-1}(n)$ may also be optimized in a sequential manner. That is, first $c_0(n)$ is optimized by minimizing $\zeta_1^{ee}(n)$. We then hold $c_0(n)$, run the sequence $\{x(1), x(2), \ldots, x(n)\}$ through the first stage of the lattice, and optimize $c_1(n)$ so that $\zeta_2^{ee}(n)$ is minimized. This process continues for the rest of the joint process estimator coefficients as well. To summarize, the regressor coefficients $c_0(n)$, $c_1(n)$, ..., $c_{N-1}(n)$ are obtained according to the following equations:

$$c_m(n) = \frac{\sum_{k=1}^{n} \lambda^{n-k} e_{m,n}(k) b_{m,n}(k)}{\sum_{k=1}^{n} \lambda^{n-k} b_{m,n}^2(k)} \tag{13.26}$$

for $m = 0, 1, \ldots, N-1$.

Equations (13.23), (13.24), and (13.26), although fundamental in providing a clear understanding of the underlying principles in the development of the least-squares lattice algorithm, cannot be used for the computation of the lattice coefficients in an adaptive application, as their computational complexity grows with the number of data samples, n. As in the case of the standard RLS algorithm, the problem is solved by finding a set of equations that update the filter coefficients in a recursive manner. This is the subject of the next section.

13.4 RLSL Algorithm

In this section, we go through a systematic step-by-step procedure to develop the recursions necessary for the derivation of the RLSL algorithm. The development of the RLSL algorithm involves a large number of variables, compared to any of the algorithms that we have derived/discussed thus far in this book. Because of this, it is often difficult for a novice to the topic to follow these equations. Thus, choosing the right set of notations that reduce this burden is crucial to the development of a readable material on this topic. Bearing this in mind, our discussion on the RLSL algorithm begins with an introduction to the notations and some preliminaries. The derivations will be followed thereafter.

13.4.1 Notations and Preliminaries

In the past few sections, we introduced a number of notations for formulating the least-squares solutions in the cases of forward and backward transversal predictors as well as lattice joint process estimator. Here, we introduce some more notations and also some new definitions that are necessary for the derivations that follow.

Adding the results of Eqs. (13.61) and (13.62), we obtain

$$\mathbf{A} + \mathbf{B} = \mathbf{I}_{m+1}.$$

This completes the proof of Eq. (13.56).

Premultiplying and postmultiplying Eq. (13.56) by $\mathbf{x}_{m+1}^{\mathrm{T}}(n)$ and $\mathbf{x}_{m+1}(n)$, respectively, and noting that

$$\mathbf{x}_{m+1}(n) = \begin{bmatrix} \mathbf{x}_m(n) \\ x(n-m) \end{bmatrix} \quad \text{and} \quad \mathbf{k}_m(n) = \boldsymbol{\Psi}_m^{-1}(n)\mathbf{x}_m(n)$$

we obtain

$$\mathbf{x}_{m+1}^{\mathrm{T}}(n)\mathbf{k}_{m+1}(n) = \mathbf{x}_m^{\mathrm{T}}(n)\mathbf{k}_m(n) + \frac{(\tilde{\mathbf{g}}_m^{\mathrm{T}}(n)\mathbf{x}_{m+1}(n))^2}{\zeta_m^{bb}(n)} \tag{13.63}$$

Using Eq. (13.40), we obtain

$$\begin{aligned} \tilde{\mathbf{g}}_m^{\mathrm{T}}(n)\mathbf{x}_{m+1}(n) &= x(n-m) - \mathbf{g}_m^{\mathrm{T}}(n)\mathbf{x}_m(n) \\ &= b_{m,n}(n) \end{aligned} \tag{13.64}$$

Finally, substituting Eq. (13.64) in Eq. (13.63), subtracting both sides of the result from 1, and recalling Eq. (13.53), we obtain

$$\gamma_{m+1}(n) = \gamma_m(n) - \frac{b_{m,n}^2(n)}{\zeta_m^{bb}(n)} \tag{13.65}$$

13.4.5 *Update Equation for Cross-Correlations*

The recursions that are remaining to complete the derivation of the RLSL algorithm are the update equations for cross-correlations $\zeta_m^{fb}(n)$ and $\zeta_m^{be}(n)$.

Recall the RLS recursion for the mth-order forward transversal predictor

$$\mathbf{a}_m(n) = \mathbf{a}_m(n-1) + \mathbf{k}_m(n-1)f_{m,n-1}(n) \tag{13.66}$$

where $\mathbf{k}_m(n-1)$ and $f_{m,n-1}(n)$, as defined earlier, are the gain vector and *a priori* estimation error of the forward predictor, respectively. The samples of the *a posteriori* estimation error of the forward predictor, for $k = 1, 2, \ldots, n$, are given by

$$f_{m,n}(k) = x(k) - \mathbf{a}_m^{\mathrm{T}}(n)\mathbf{x}_m(k-1) \tag{13.67}$$

Substituting Eq. (13.66) in Eq. (13.67) and rearranging, we obtain

$$f_{m,n}(k) = f_{m,n-1}(k) - \mathbf{k}_m^{\mathrm{T}}(n-1)\mathbf{x}_m(k-1)f_{m,n-1}(n) \tag{13.68}$$

where

$$f_{m,n-1}(k) = x(k) - \mathbf{a}_m^{\mathrm{T}}(n-1)\mathbf{x}_m(k-1) \tag{13.69}$$

for $k = 1, 2, \ldots, n$ are samples of the *a priori* forward prediction error.

In addition, recall the RLS recursion for the mth-order backward predictor

$$\mathbf{g}_m(n) = \mathbf{g}_m(n-1) + \mathbf{k}_m(n)b_{m,n-1}(n) \tag{13.70}$$

On the other hand, post- and premultiplying Eq. (13.91) by $\mathbf{x}_{N+1}(n)$ and $\mathbf{x}_{N+1}^{\mathrm{T}}(n)$, respectively, subtracting both sides of the result from unity, and recalling Eq. (13.53), we obtain

$$\gamma_{N+1}(n) = \gamma_N(n-1) - \frac{f_{N,n}^2(n)}{\zeta_N^{ff}(n)} \tag{13.94}$$

Substituting Eq. (13.94) in Eq. (13.93) and dividing both sides of the result by $\gamma_{N+1}(n)$, we get

$$\bar{\mathbf{k}}_{N+1}(n) = \begin{bmatrix} 0 \\ \bar{\mathbf{k}}_N(n-1) \end{bmatrix} + \frac{f_{N,n}(n)}{\gamma_{N+1}(n)\zeta_N^{ff}(n)} \tilde{\mathbf{a}}_N(n-1) \tag{13.95}$$

Moreover, combining Eqs. (13.94) and (13.45), it is straightforward to show that (Problem P13.17)

$$\gamma_{N+1}(n)\zeta_N^{ff}(n) = \lambda\gamma_N(n-1)\zeta_N^{ff}(n-1) \tag{13.96}$$

Finally, substituting Eq. (13.96) in Eq. (13.95) and using Eq. (13.52), we obtain

$$\bar{\mathbf{k}}_{N+1}(n) = \begin{bmatrix} 0 \\ \bar{\mathbf{k}}_N(n-1) \end{bmatrix} + \lambda^{-1} \frac{f_{N,n-1}(n)}{\zeta_N^{ff}(n-1)} \tilde{\mathbf{a}}_N(n-1) \tag{13.97}$$

This recursion gives a time as well as order-update of the normalized gain vector. Next, we develop another recursion that keeps the time index of the normalized gain vector fixed at n, but reduces its length from $N+1$ to N. This also leads to a time update of the tap-weight vector of the backward predictor.

Backward Prediction

Consider Eq. (13.56) with $m = N$. Then, postmultiplying it by $\mathbf{x}_{N+1}(n)$ and recalling Eqs. (13.40) and (13.88), we obtain

$$\gamma_{N+1}(n)\bar{\mathbf{k}}_{N+1}(n) = \gamma_N(n) \begin{bmatrix} \bar{\mathbf{k}}_N(n) \\ 0 \end{bmatrix} + \frac{b_{N,n}(n)}{\zeta_N^{bb}(n)} \tilde{\mathbf{g}}_N(n) \tag{13.98}$$

Equating the last elements of the vectors on both sides of Eq. (13.98) and rearranging, we obtain

$$\bar{\mathbf{k}}_{N+1,N+1}(n) = \frac{b_{N,n}(n)}{\gamma_{N+1}(n)\zeta_N^{bb}(n)} \tag{13.99}$$

where $\bar{\mathbf{k}}_{N+1,N+1}(n)$ denotes the last element of $\bar{\mathbf{k}}_{N+1}(n)$. On the other hand, combining Eqs. (13.46) and (13.65), and replacing m by N, it is straightforward to show that (Problem P13.18)

$$\gamma_{N+1}(n)\zeta_N^{bb}(n) = \lambda\gamma_N(n)\zeta_N^{bb}(n-1) \tag{13.100}$$

Substituting Eq. (13.100) in Eq. (13.99), recalling Eq. (13.54), and rearranging the result, we get

$$b_{N,n-1}(n) = \lambda\zeta_N^{bb}(n-1)\bar{\mathbf{k}}_{N+1,N+1}(n) \tag{13.101}$$

(i)

$$\sum_{k=1}^{n} \lambda^{n-k} b_{m-1,n-1}(k-1) f_{m,n}(k) = 0$$

(ii)

$$\sum_{k=1}^{n} \lambda^{n-k} f_{m-1,n}(k) b_{m,n}(k) = 0$$

P13.8 Consider a linear adaptive filter with tap-input vectors $\mathbf{x}(1)$, $\mathbf{x}(2)$, ..., $\mathbf{x}(n)$, and desired output sequence

$$d(k) = \begin{cases} 1, & k = n \\ 0, & k = 1, 2, \ldots, n-1 \end{cases}$$

Find the least-squares error sum of this filter and show that it is equal to the conversion factor $\gamma(n)$ as given by Eq. (13.49) or (13.50). Then, prove that

$$0 \leq \gamma(n) \leq 1$$

P13.9 Show that in a lattice structure, at any instant of time, n,

$$\gamma_{m+1}(n) \leq \gamma_m(n)$$

P13.10 Prove that

$$\frac{\zeta_m^{ff}(n)}{\zeta_{m-1}^{ff}(n)} = \frac{\zeta_m^{bb}(n)}{\zeta_{m-1}^{bb}(n-1)} = 1 - \kappa_m^{\mathrm{f}}(n)\kappa_m^{\mathrm{b}}(n)$$

P13.11 Use the result of Problem P13.10 to derive the following update equations for the least-squares sums $\zeta_m^{ff}(n)$ and $\zeta_m^{bb}(n)$:

$$\zeta_m^{ff}(n) = \zeta_{m-1}^{ff}(n) - \frac{(\zeta_{m-1}^{fb}(n))^2}{\zeta_{m-1}^{bb}(n-1)}$$

and

$$\zeta_m^{bb}(n) = \zeta_{m-1}^{bb}(n-1) - \frac{(\zeta_{m-1}^{fb}(n))^2}{\zeta_{m-1}^{ff}(n)}$$

P13.12 Obtain the normal equation that results from the least-square optimization of the *a posteriori* estimation error $e_{N,n}(k)$ of the joint process estimator of Figure 13.3 and show that this leads to the following set of independent equations:

$$c_m(n) = \frac{\sum_{k=1}^{n} \lambda^{n-k} b_{m,n}(k) d(k)}{\sum_{k=1}^{n} \lambda^{n-k} b_{m,n}^2(k)}$$

for $m = 0, 1, \ldots, N-1$. Then, use the orthogonality of the backward errors $b_{m,n}(k)$, for $m = 0, 1, \ldots, N$, to convert these equations to those given in Eq. (13.26).

P13.13 Give a detailed proof of Eq. (13.81).

P13.14 Derive the update equations of $\kappa_{m+1}^{\mathrm{b}}(n)$ and $c_m(n)$ that have appeared in Table 13.2.

P13.15 Prove the following identity

$$\boldsymbol{\Psi}_{m+1}^{-1}(n) = \begin{bmatrix} 0 & \mathbf{0}_m^{\mathrm{T}} \\ \mathbf{0}_m & \boldsymbol{\Psi}_m^{-1}(n-1) \end{bmatrix} + \frac{1}{\zeta_m^{ff}(n)} \tilde{\mathbf{a}}_m(n) \tilde{\mathbf{a}}_m^{\mathrm{T}}(n)$$

P13.16 Show that

$$\gamma_m(n) = \gamma_m(n-1) + \frac{b_{m,n}^2(n)}{\zeta_m^{bb}(n)} - \frac{f_{m,n}^2(n)}{\zeta_m^{ff}(n)}$$

P13.17 Give a detailed derivations of Eq. (13.96).

P13.18 Give a detailed derivations of Eq. (13.100).

P13.19 Use the result of Problems P13.17 and P13.18 to obtain a time-update equation relating $\gamma_m(n)$ and $\gamma_m(n-1)$.

P13.20 Explore the possibility of rearranging the recursions/equations in Table 13.1 in terms of *a priori* estimation errors. Thus, suggest an alternative implementation RLSL algorithm using the *a priori* estimation errors.

Using these and Eq. (14.16) in Eq. (14.17), we obtain

$$2\xi_{\text{ex}} = 2\text{tr}[\boldsymbol{\mu}\mathbf{R}]\xi_{\text{ex}} + 4\text{tr}[\boldsymbol{\mu}\mathbf{R}\mathbf{K}\mathbf{R}] + 2\sigma_{e_o}^2\text{tr}[\boldsymbol{\mu}\mathbf{R}] + \frac{1}{2}\text{tr}[\boldsymbol{\mu}^{-1}\mathbf{G}] \tag{14.18}$$

To arrive at a mathematically tractable result, we assume that the term $\text{tr}[\boldsymbol{\mu}\mathbf{R}\mathbf{K}\mathbf{R}]$ in Eq. (14.18) can be ignored. Numerical examples and computer simulations show that when N (the filter length) is large $\text{tr}[\boldsymbol{\mu}\mathbf{R}\mathbf{K}\mathbf{R}]$ is usually at least an order of magnitude smaller than $\text{tr}[\boldsymbol{\mu}\mathbf{R}]\xi_{\text{ex}}$. See Problem P14.1 for more exposure over this approximation. This leads to the following result:

$$\xi_{\text{ex}} = \frac{1}{1 - \text{tr}[\boldsymbol{\mu}\mathbf{R}]}\left(\sigma_{e_o}^2\text{tr}[\boldsymbol{\mu}\mathbf{R}] + \frac{1}{4}\text{tr}[\boldsymbol{\mu}^{-1}\mathbf{G}]\right) \tag{14.19}$$

Using Eq. (14.19) to evaluate the misadjustment of the generalized LMS algorithm, we obtain

$$\mathcal{M} = \frac{\xi_{\text{ex}}}{\xi_{\text{min}}} = \frac{1}{1 - \text{tr}[\boldsymbol{\mu}\mathbf{R}]}\left(\text{tr}[\boldsymbol{\mu}\mathbf{R}] + \frac{1}{4}\sigma_{e_o}^{-2}\text{tr}[\boldsymbol{\mu}^{-1}\mathbf{G}]\right) \tag{14.20}$$

where $\xi_{\text{min}} = \sigma_{e_o}^2$ is the minimum MSE of the filter that is obtained when $\mathbf{w}(n) = \mathbf{w}_o(n)$.

To relate this result to the results of the previous chapters, let us consider the case of conventional LMS algorithm. In this case, $\boldsymbol{\mu} = \mu\mathbf{I}$, where μ is a scalar step-size parameter. Substituting $\boldsymbol{\mu}$ by $\mu\mathbf{I}$ in Eq. (14.20), we obtain

$$\mathcal{M}_{\text{LMS}} = \frac{1}{1 - \mu\text{tr}[\mathbf{R}]}\left(\mu\text{tr}[\mathbf{R}] + \frac{1}{4}\sigma_{e_o}^{-2}\mu^{-1}\text{tr}[\mathbf{G}]\right) \tag{14.21}$$

It is instructive to note that when the plant tap-weight, $\mathbf{w}_o$, is time invariant, the correlation matrix $\mathbf{G}$ is zero, as $\boldsymbol{\epsilon}_o(n)$ is zero for all values of n. Thus, we obtain

$$\mathcal{M}_{\text{LMS}} = \frac{\mu\text{tr}[\mathbf{R}]}{1 - \mu\text{tr}[\mathbf{R}]} \tag{14.22}$$

This is exactly the result that we obtained in Chapter 6 – see Eq. (6.63). This observation shows that Eq. (14.20) is in fact a generalization of similar results that were obtained in the previous chapters. This includes the effect of plant variation and also the use of different step-size parameters at various taps. Moreover, we see that when the plant is time varying, there are two distinct terms contributing to the misadjustment of the LMS algorithm. Accordingly, we may write

$$\mathcal{M} = \mathcal{M}_1 + \mathcal{M}_2 \tag{14.23}$$

where

$$\mathcal{M}_1 = \frac{\text{tr}[\boldsymbol{\mu}\mathbf{R}]}{1 - \text{tr}[\boldsymbol{\mu}\mathbf{R}]} \tag{14.24}$$

and

$$\mathcal{M}_2 = \frac{\sigma_{e_o}^{-2}}{4} \cdot \frac{\text{tr}[\boldsymbol{\mu}^{-1}\mathbf{G}]}{1 - \text{tr}[\boldsymbol{\mu}\mathbf{R}]} \tag{14.25}$$

With reference to recursion (14.6) and the subsequent derivations, we find that $\mathcal{M}_1$ originates from the term $2\boldsymbol{\mu}e_o(n)\mathbf{x}(n)$ on the right-hand side of Eq. (14.6). This, clearly, is

It is instructive to note that Eq. (14.30) is intuitively sound. It suggests that those taps that have larger tap perturbation should be given larger step-size parameters. It also suggests normalization of the step-size parameters proportional to the inverse of the signal level at various taps. However, this normalization is different from the one commonly used in the step-normalized algorithms, where μ_i is selected proportional to the inverse of signal power at the respective tap, that is, proportional to $1/\sigma_{x_i}^2$. Moreover, Eq. (14.30) suggests that the step-size parameters should be reduced as the error level at the filter output increases – note that η^2 is equal to the MSE of the filter after it has converged.

The validity of Eq. (14.30) is subject to the condition that the optimum step-size parameters remain in a range that does not result in instability of the algorithm. For the case of the conventional LMS algorithm, where a single step-size parameter, μ, is employed, a useful and practically applicable upper bound for μ is the one derived in Chapter 6 and repeated below for convenience (6.74):

$$\mu < \frac{1}{3\text{tr}[\mathbf{R}]} \tag{14.31}$$

or, equivalently,

$$\mu\text{tr}[\mathbf{R}] < \frac{1}{3} \tag{14.32}$$

This result can be extended to the generalized LMS recursion Eq. (14.3) as follows.

Consider the recursion Eq. (14.3) and define $\tilde{\mathbf{x}}(n) = \boldsymbol{\mu}^{1/2}\mathbf{x}(n)$, where $\boldsymbol{\mu}^{1/2}$ is the diagonal matrix consisting of the square roots of the diagonal elements of $\boldsymbol{\mu}$. Then, multiplying both sides of Eq. (14.6) from left by $\boldsymbol{\mu}^{-1/2}$ and, also, defining $\tilde{\mathbf{v}}(n) = \boldsymbol{\mu}^{-1/2}\mathbf{v}(n)$ and $\tilde{\boldsymbol{\epsilon}}_o(n) = \boldsymbol{\mu}^{-1/2}\boldsymbol{\epsilon}_o(n)$, we obtain

$$\tilde{\mathbf{v}}(n+1) = (\mathbf{I} - 2\tilde{\mathbf{x}}(n)\tilde{\mathbf{x}}^{\mathrm{T}}(n))\tilde{\mathbf{v}}(n) + 2e_o(n)\tilde{\mathbf{x}}(n) - \tilde{\boldsymbol{\epsilon}}_o(n) \tag{14.33}$$

The recursion Eq. (14.33) is similar to the conventional LMS recursion with $\mu = 1$. Accordingly, Eq. (14.32) can be applied. Hence, we find that the stability of Eq. (14.33), and thus Eq. (14.6) or, equivalently, the recursion Eq. (14.3), is guaranteed if

$$\text{tr}[\tilde{\mathbf{R}}] < 1/3 \tag{14.34}$$

where

$$\begin{aligned}\tilde{\mathbf{R}} &= E[\tilde{\mathbf{x}}(n)\tilde{\mathbf{x}}^{\mathrm{T}}(n)] \\ &= E[\boldsymbol{\mu}^{1/2}\mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n)\boldsymbol{\mu}^{1/2}] \\ &= \boldsymbol{\mu}^{1/2}E[\mathbf{x}(n)\mathbf{x}^{\mathrm{T}}(n)]\boldsymbol{\mu}^{1/2} \\ &= \boldsymbol{\mu}^{1/2}\mathbf{R}\boldsymbol{\mu}^{1/2}\end{aligned} \tag{14.35}$$

Substituting this result in Eq. (14.34) and noting that $\text{tr}[\boldsymbol{\mu}^{1/2}\mathbf{R}\boldsymbol{\mu}^{1/2}] = \text{tr}[\boldsymbol{\mu}\mathbf{R}]$ (according to identity $\text{tr}[\mathbf{AB}] = \text{tr}[\mathbf{BA}]$), we get

$$\text{tr}[\boldsymbol{\mu}\mathbf{R}] < \frac{1}{3} \tag{14.36}$$

This is a sufficient condition that may be imposed on the algorithm step-size parameters, μ_i's, to guarantee the stability of the generalized LMS recursion Eq. (14.3).

When Eq. (14.36) holds, the minimum excess MSE of the filter, $\xi_{\text{ex,o}}$, is obtained by substituting Eq. (14.30) in Eq. (14.26). This gives

$$\xi_{\text{ex,o}} = \frac{1}{2 - \frac{1}{\eta}\sum_i \sigma_{\epsilon_{\text{o},i}}\sigma_{x_i}} \left(\frac{\sigma_{e_\text{o}}^2}{\eta} + \eta\right) \sum_i \sigma_{\epsilon_{\text{o},i}}\sigma_{x_i} \tag{14.37}$$

Substituting for η from Eq. (14.29) in Eq. (14.37), we get

$$\xi_{\text{ex,o}} = \frac{\sum_i \sigma_{\epsilon_{\text{o},i}}\sigma_{x_i} + \sqrt{\left(\sum_i \sigma_{\epsilon_{\text{o},i}}\sigma_{x_i}\right)^2 + 4\sigma_{e_\text{o}}^2}}{2} \sum_i \sigma_{\epsilon_{\text{o},i}}\sigma_{x_i} \tag{14.38}$$

14.5 Comparisons of Conventional Algorithms

In this section, we compare tracking behavior of various versions of the LMS algorithm in the context of the modeling problem discussed in the last few sections. Noting that the tracking behaviors of the RLS and LMS-Newton algorithms are about the same, the comparisons also cover the RLS algorithm. *The indicator of better tracking behavior (performance) is lower steady-state excess MSE.*

In order to prevent diverting into many possible cases, we concentrate on the comparison of the direct implementation of a transversal filter using the LMS algorithm, and its implementation in transform domain. We note that for a transversal filter $\mathbf{x}(n) = [x(n)\ x(n-1)\ \ldots\ x(n-N+1)]^\text{T}$, and for its transform-domain implementation $\mathbf{x}(n)$ is replaced by $\mathbf{x}_\mathcal{T}(n) = \mathcal{T}\mathbf{x}(n)$, where $\mathcal{T}$ is an orthonormal transformation matrix satisfying the condition[1]

$$\mathcal{T}\mathcal{T}^\text{T} = \mathbf{I} \tag{14.39}$$

where $\mathbf{I}$ is the identity matrix.

In addition, if $\boldsymbol{\epsilon}_\text{o}(n)$ represents the plant tap-weight increments in its transversal form, the corresponding increments in the transform domain are given by

$$\boldsymbol{\epsilon}_{\mathcal{T},\text{o}}(n) = \mathcal{T}\boldsymbol{\epsilon}_\text{o}(n) \tag{14.40}$$

We also define $\mathbf{R}_\mathcal{T} = E[\mathbf{x}_\mathcal{T}(n)\mathbf{x}_\mathcal{T}^\text{T}(n)]$ and $\mathbf{G}_\mathcal{T} = E[\boldsymbol{\epsilon}_{\mathcal{T},\text{o}}(n)\boldsymbol{\epsilon}_{\mathcal{T},\text{o}}^\text{T}(n)]$ and note that

$$\mathbf{R}_\mathcal{T} = \mathcal{T}\mathbf{R}\mathcal{T}^\text{T} \tag{14.41}$$

and

$$\mathbf{G}_\mathcal{T} = \mathcal{T}\mathbf{G}\mathcal{T}^\text{T} \tag{14.42}$$

The ith diagonal elements of $\mathbf{R}_\mathcal{T}$ and $\mathbf{G}_\mathcal{T}$ are denoted as $\sigma_{x_{\mathcal{T},i}}^2$ and $\sigma_{\epsilon_{\mathcal{T},\text{o},i}}^2$, respectively. Moreover, to simplify our discussion, yet with no loss of generality, we assume that the input sequence to the transversal filter is normalized to unit power, that is, $\sigma_{x_i}^2 = E\{|x(n-i)|^2\} = 1$, for $i = 0, 1, \ldots, N-1$. Then, the orthonormality of $\mathcal{T}$, that is, the

[1] To avoid complex-valued coefficients/variables in our formulations in this chapter, we only consider transformations with real-valued coefficients.

14.6 Comparisons Based on Optimum Step-Size Parameters

From the theoretical results of Section 14.4 and the definitions of $\mathbf{R}_{\mathcal{T}}$ and $\mathbf{G}_{\mathcal{T}}$, in the last section, we recall that when the optimum step-size parameters given by Eq. (14.30) are used, the excess MSE of the TDLMS algorithm is given by Eq. (14.38)

$$\xi_{\text{ex,o}}(\text{TDLMS}) = \frac{\Gamma_{\mathcal{T}} + \sqrt{\Gamma_{\mathcal{T}}^2 + 4\sigma_{e_o}^2}}{2}\Gamma_{\mathcal{T}} \tag{14.54}$$

where

$$\Gamma_{\mathcal{T}} = \sum_{i=0}^{N-1} \sigma_{\epsilon_{\mathcal{T},\text{o},i}} \sigma_{x_{\mathcal{T},i}} \tag{14.55}$$

We note that $\Gamma_{\mathcal{T}}$ is a function of $\mathbf{R}$, $\mathbf{G}$, and $\boldsymbol{\mathcal{T}}$.

We also note that when no transformation has been applied, but the optimum step-size parameters are used for different taps, the excess MSE of the LMS algorithm is given by

$$\xi_{\text{ex,o}}(\text{LMS}) = \frac{\Gamma_{\text{I}} + \sqrt{\Gamma_{\text{I}}^2 + 4\sigma_{e_o}^2}}{2}\Gamma_{\text{I}} \tag{14.56}$$

where

$$\Gamma_{\text{I}} = \sum_{i=0}^{N-1} \sigma_{\epsilon_{\text{o},i}} \sigma_{x_i} \tag{14.57}$$

Clearly, to achieve the best tracking performance of the TDLMS algorithm, one should find the matrix $\boldsymbol{\mathcal{T}}$, which minimizes $\Gamma_{\mathcal{T}}$. A general solution to this problem appears to be difficult. We thus limit ourselves to a few particular cases whose study is found to be instructive. The following lemma is widely used in the study of the cases that follows.

Lemma 14.1 *Consider the diagonal matrix* $\boldsymbol{\Lambda} = \text{diag}\,(\lambda_0, \lambda_1, \ldots, \lambda_{N-1})$, *where* λ_i*'s are all real and nonnegative. If* $\boldsymbol{\mathcal{T}}$ *is an orthonormal matrix, that is,* $\boldsymbol{\mathcal{T}}\boldsymbol{\mathcal{T}}^T = \mathbf{I}$, *and* $\mathbf{S} = \boldsymbol{\mathcal{T}}\boldsymbol{\Lambda}\boldsymbol{\mathcal{T}}^{\text{T}}$, *then the following inequality always holds:*

$$\sum_{i=0}^{N-1} \sqrt{\lambda_i} \le \sum_{i=0}^{N-1} \sqrt{s_{ii}} \tag{14.58}$$

where s_{ii} *is the* i*th diagonal element of* $\mathbf{S}$.

Proof. We first note that for $x \ge 0$, $f(x) = \sqrt{x}$ is a concave function. In addition, according to the theory of the convex functions, Rockafellar (1970), if $f(x)$ is a concave function and $\zeta_0, \zeta_1, \ldots, \zeta_{N-1}$ are a set of nonnegative numbers that satisfy $\sum_{i=0}^{N-1} \zeta_i = 1$, then for any set of numbers $x_0, x_1, \ldots, x_{N-1}$ in the domain of $f(x)$, the following inequality holds

$$\sum_{i=0}^{N-1} \zeta_i f(x_i) \le f\left(\sum_{i=0}^{N-1} \zeta_i x_i\right) \tag{14.59}$$

Eq. (14.79), are presented. We note that both implementations of the VSLMS algorithm converge to about the same excess MSE as the case where optimal step-size parameters are used. This clearly illustrates the optimal tracking behavior of the VSLMS algorithm, as was predicted earlier in this section. We also note that there is very little difference between the behavior of the recursion Eq. (14.78) and its sign counterpart, Eq. (14.79).

Figure 14.5 presents the results showing how the VSLMS algorithm track variation of $\mu_{o,10}(n)$, that is, the optimum step-size parameter of the 10th tap of the adaptive filter. The results are given for the recursion Eq. (14.78) and also its sign counterpart, Eq. (14.79). The results show that the VSLMS algorithm converges to the optimum step-size parameters, thus achieving a close to optimum tracking behavior. Further experiments have confirmed this optimum performance even when the adaptive filter input is colored (Mathews and Xie, 1993; Farhang-Boroujeny and Gazor, 1994).

Computation of the optimum step-size parameters, which are used to obtain the results of Figures 14.4 and 14.5, is carried out by finding σ_{x_i} and $\sigma_{\epsilon_{o,i}}$ first, and then substituting them in Eq. (14.30). Noting that the filter input is a binary sequence, we get $\sigma_{x_i} = 1$. To evaluate $\sigma_{\epsilon_{o,i}}$, we recall from Eq. (14.91) that the path gains, $a_1(nT_s)$ and $a_2(nT_s)$ are generated using the recursions

$$a_k((n+1)T_s) = \alpha a_k(nT_s) + \sqrt{1-\alpha^2}\nu_k(n+1), \quad \text{for } k = 1, \ 2 \tag{14.93}$$

where $\nu_1(n+1)$ and $\nu_2(n+1)$ are two independent unit-variance zero-mean Gaussian white sequences. Assuming that α is smaller but close to 1, we find that

$$1 - \alpha \approx \sqrt{1-\alpha^2}$$

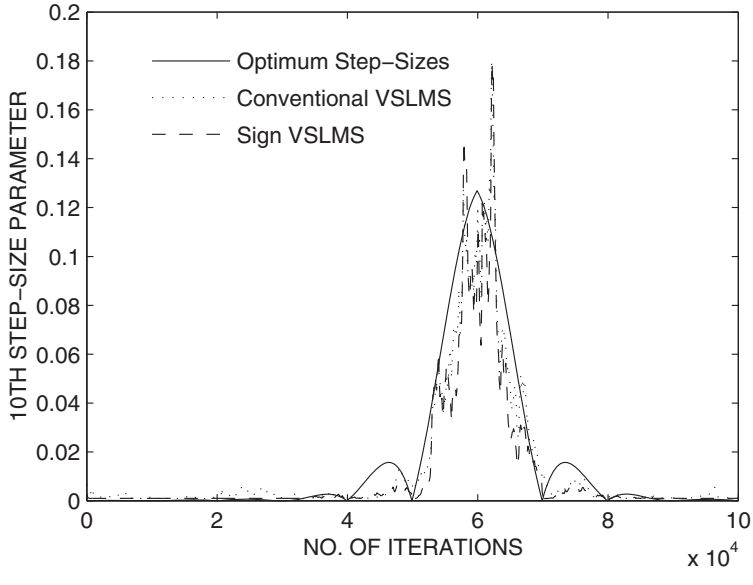

Figure 14.5 A typical simulation result showing that the VSLMS algorithm closely track the optimum step-size parameters given by Eq. (14.30).

P14.14 Develop a simulation program to study the convergence behavior of the VSLMS algorithm in the channel modeling application that was discussed in Section 14.7.4, when a common step-size parameter is used for all taps. Compare your results with those of Figures 14.3–14.5, and discuss your observations.

P14.15 Develop a simulation program to study the convergence behavior of the RLS algorithm with variable forgetting factor (Table 14.2) in the channel modeling application that was discussed in Section 14.7.4. Compare your results with those of Figures 14.3–14.5 and also your results in Problem P14.14. Discuss your observations.

echo canceller, $e(n)$ is the signal sent to the far-end room. Moreover, it may be written as

$$\begin{aligned} e(n) &= d(n) - \hat{y}(n) \\ &= e_{\mathrm{o}}(n) + \mathbf{w}_{\mathrm{o}}^{\mathrm{T}}\mathbf{x}(n) - \mathbf{w}^{\mathrm{T}}(n)\mathbf{x}(n) \\ &= e_{\mathrm{o}}(n) - \mathbf{v}^{\mathrm{T}}(n)\mathbf{x}(n) \end{aligned} \tag{15.9}$$

where $e_{\mathrm{o}}(n)$ is the signal corresponding to the sound activities in the near-end room, excluding the far-end signal that is broadcast in the room through the loudspeaker. We also note that $\mathbf{v}^{\mathrm{T}}(n)\mathbf{x}(n)$ is an undesirable residual echo. Assuming that $e_{\mathrm{o}}(n)$ has a zero mean, and defining

$$r(n) = \mathbf{v}^{\mathrm{T}}(n)\mathbf{x}(n) \tag{15.10}$$

Equation (15.8) reduces to

$$E[\|\mathbf{v}(n+1)\|^2] = \|\mathbf{v}(n)\|^2 + \tilde{\mu}^2 \frac{E[e^2(n)]}{\mathbf{x}^{\mathrm{T}}(n)\mathbf{x}(n)} - 2\tilde{\mu}\frac{r^2(n)}{\mathbf{x}^{\mathrm{T}}(n)\mathbf{x}(n)} \tag{15.11}$$

The goal is to find the value of $\tilde{\mu}$ that minimizes $E[\|\mathbf{v}(n+1)\|^2]$. This, obviously, is obtained by solving the equation

$$\frac{\partial E[\|\mathbf{v}(n+1)\|^2]}{\partial \tilde{\mu}} = 0 \tag{15.12}$$

This leads to

$$\begin{aligned} \tilde{\mu}_{\mathrm{opt}}(n) &= \frac{r^2(n)}{E[e^2(n)]} \\ &= \frac{r^2(n)}{E[e_{\mathrm{o}}^2(n)] + r^2(n)} \end{aligned} \tag{15.13}$$

where the second identity is obtained using Eq. (15.80) and recalling that $e_{\mathrm{o}}(n)$ and $r(n) = \mathbf{v}^{\mathrm{T}}(n)\mathbf{x}(n)$ are uncorrelated.

It is important to make some comments on the significance of the optimum step-size parameter (15.13).

- When $E[e_{\mathrm{o}}^2(n)] = 0$, that is, when $e_{\mathrm{o}}(n) = 0$ one finds that $\tilde{\mu}_{\mathrm{opt}}(n) = 1$. In other words, if one thinks of the echo canceller as a modeling problem just as the one in Figure 6.5, when there is no noise at the output of the plant, $\tilde{\mu}_{\mathrm{opt}}(n) = 1$ leads to an optimum performance of the NLMS algorithm.
- The optimum step-size parameter decreases as $e_{\mathrm{o}}(n)$ increases in power.
- At the beginning of adaptation, when $\mathbf{w}(n) = 0$, $r^2(n)$ is generally large and thus, unless $E[e_{\mathrm{o}}^2(n)]$ is also large, $\tilde{\mu}_{\mathrm{opt}}(n)$ will be approximately equal to 1.
- As the algorithm iterates, $r^2(n)$ reduces, on average, and thus, in the presence of some nonzero $E[e_{\mathrm{o}}^2(n)]$, $\tilde{\mu}_{\mathrm{opt}}(n)$ decreases. This, clearly, is a sound strategy for reducing the filter misadjustment and/or misalignment.
- In case $\mathbf{w}_{\mathrm{o}}$ changes, $r^2(n)$ increases and, hence, $\tilde{\mu}_{\mathrm{opt}}(n)$ will increase to track the change of $\mathbf{w}_{\mathrm{o}}$.

Table 15.3 Summary of the variable step-size normalized PFBLMS (VSNPFBLMS) algorithm for echo cancellation.

Input: Tap-weight vectors, $\mathbf{w}_{\mathcal{F},l}(k)$, $l = 0, 1, \ldots, P-1$,
extended input vector,
$\tilde{\mathbf{X}}_0(k) = [x(kL-M)\; x(kL-M+1) \cdots x(kL+L-1)]^{\mathrm{T}}$,
the past frequency domain vectors of input, $\mathbf{x}_{\mathcal{F},0}(k-l)$, for $l = 1, 2, \ldots, (P-1)p$,
and the desired output vector,
$\mathbf{d}(k) = [d(kL)\; d(kL+1) \cdots d(kL+L-1)]^{\mathrm{T}}$.

Output: Tap-weight vector update, $\mathbf{w}_{\mathcal{F},l}(k+1)$, $l = 0, 1, \ldots, P-1$,
power estimates $\sigma_{\mathrm{d}}^2(k)$, $\sigma_{\hat{y}}^2(k)$, and $\sigma_e^2(k)$
The echo canceller output is $e(n)$.

1. Filtering:

$$\mathbf{x}_{\mathcal{F},0}(k) = \mathrm{FFT}(\tilde{\mathbf{X}}_0(k))$$

$$\hat{\mathbf{y}}(k) = \text{the last } L \text{ elements of IFFT}\left(\sum_{l=0}^{P-1} \mathbf{w}_{\mathcal{F},l}(k) \odot \mathbf{x}_{\mathcal{F},0}(k-pl)\right)$$

2. Error estimation:

$$\mathbf{e}(k) = \mathbf{d}(k) - \hat{\mathbf{y}}(k)$$

3. Step-size calculation:

$$\sigma_{\mathrm{d}}^2(k) = \alpha\sigma_{\mathrm{d}}^2(k-1) + (1-\alpha)(\mathbf{d}^{\mathrm{H}}(k)\mathbf{d}(k))$$

$$\sigma_{\hat{y}}^2(k) = \alpha\sigma_{\hat{y}}^2(k-1) + (1-\alpha)(\hat{\mathbf{y}}^{\mathrm{H}}(k)\hat{\mathbf{y}}(k))$$

$$\sigma_e^2(k) = \alpha\sigma_e^2(k-1) + (1-\alpha)(\mathbf{e}^{\mathrm{H}}(n)\mathbf{e}(k))$$

$$\tilde{\mu}(k) = 1 - \frac{\sigma_{\mathrm{d}}^2(k) - \sigma_{\hat{y}}^2(k)}{\sigma_e^2(k)}$$

if $\tilde{\mu}(k) > 1$, let $\tilde{\mu}(k) = 1$
if $\tilde{\mu}(k) < 0$, let $\tilde{\mu}(k) = 0$
for $i = 0$ to $M' - 1$

$$\mu_i(k) = \tilde{\mu}(k) / \sum_{l=0}^{P-1} x_{\mathcal{F},0,i}^2(k-pl)$$

$$\boldsymbol{\mu}(k) = [\mu_0(k)\; \mu_1(k) \cdots \mu_{M'-1}(k)]^{\mathrm{T}}$$

4. Tap-weight adaptation:

$$\mathbf{e}_{\mathcal{F}}(k) = \mathrm{FFT}\left(\begin{bmatrix}\mathbf{0}\\ \mathbf{e}(k)\end{bmatrix}\right)$$

for $l = 0$ to $P-1$

$$\mathbf{w}_{\mathcal{F},l}(k+1) = \mathbf{w}_{\mathcal{F},l}(k) + 2\boldsymbol{\mu}(k) \odot \mathbf{x}_{\mathcal{F},0}^*(k-pl) \odot \mathbf{e}_{\mathcal{F}}(k)$$

5. Tap-weight constraint:

for $l = 0$ to $P-1$

$$\mathbf{w}_{\mathcal{F},l}(k+1) = \mathrm{FFT}\left(\begin{bmatrix}\text{first } M \text{ elements of IFFT}(\mathbf{w}_{\mathcal{F},l}(k+1))\\ \mathbf{0}\end{bmatrix}\right)$$

Notes:
- M: partition length; L: block length; $M' = M + L$.
- $\mathbf{0}$ denotes column zero vectors with appropriate length to extend vectors to the length of M'.
- $\odot$ denotes elementwise multiplication of vectors.
- Step 5 is applicable only for the constrained PFBLMS algorithm.

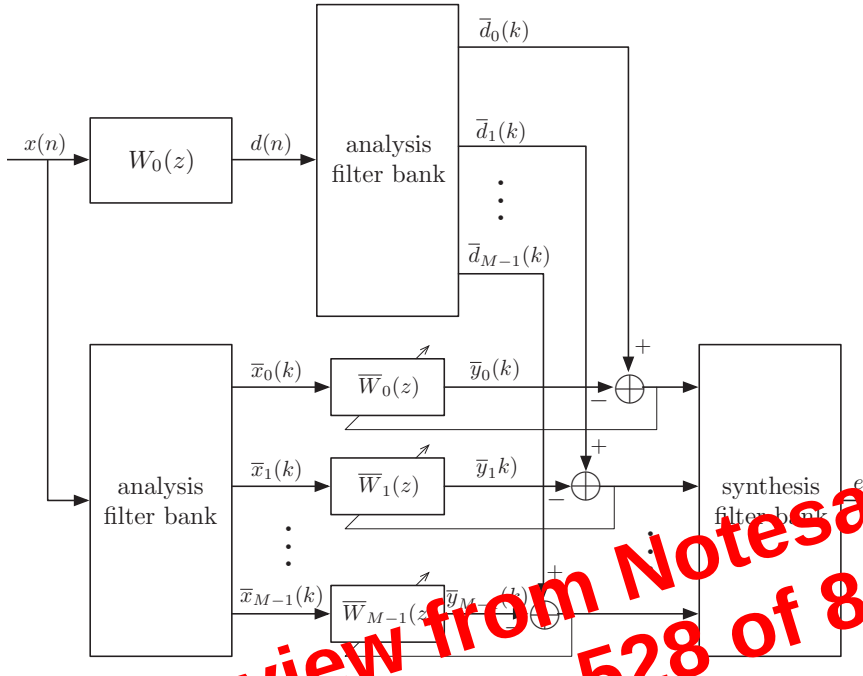

Figure 15.3 Schematic diagram of a subband synthesis-independent echo canceller.

Here, because of its simple structure, we limit our discussion to Algorithm 2 of Section 11.15. Also, to take care of the nonstationary nature of the speech signals, we develop a normalized version of the algorithm. Referring back to Chapter 11, Table 11.7, we recall that the tap-weight update equation

$$\mathbf{w}(n+1) = \mathbf{w}(n) + 2\mu e(n)\mathbf{u}_{\mathrm{a}}(n) \tag{15.29}$$

where as discussed in Section 11.15, $\mathbf{u}_{\mathrm{a}}(n)$ is an approximation to $\mathbf{R}^{-1}\mathbf{x}(n-M)$. Following the same line of derivation as those that led to Eq. (6.119), we begin with defining the *a posteriori* error

$$e^{+}(n) = d(n) - \mathbf{w}^{\mathrm{T}}(n+1)\mathbf{x}(n-M) \tag{15.30}$$

Substituting Eq. (15.29) in Eq. (15.30), replacing μ by $\mu(n)$, and rearranging, we obtain

$$e^{+}(n) = (1 - 2\mu(n)\mathbf{u}_{\mathrm{a}}^{\mathrm{T}}(n)\mathbf{x}(n-M))e(n) \tag{15.31}$$

Next, as in the case of the NLMS algorithm, minimizing $(e^{+}(n))^2$ with respect to $\mu(n)$ results in

$$\mu(n) = \frac{1}{2\mathbf{u}_{\mathrm{a}}^{\mathrm{T}}(n)\mathbf{x}(n-M)} \tag{15.32}$$

This forces $e^{+}(n)$ to 0. Substituting Eq. (15.32) in Eq. (15.29), we obtain

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \frac{1}{\mathbf{u}_{\mathrm{a}}^{\mathrm{T}}(n)\mathbf{x}(n-M)}e(n)\mathbf{u}_{\mathrm{a}}(n) \tag{15.33}$$

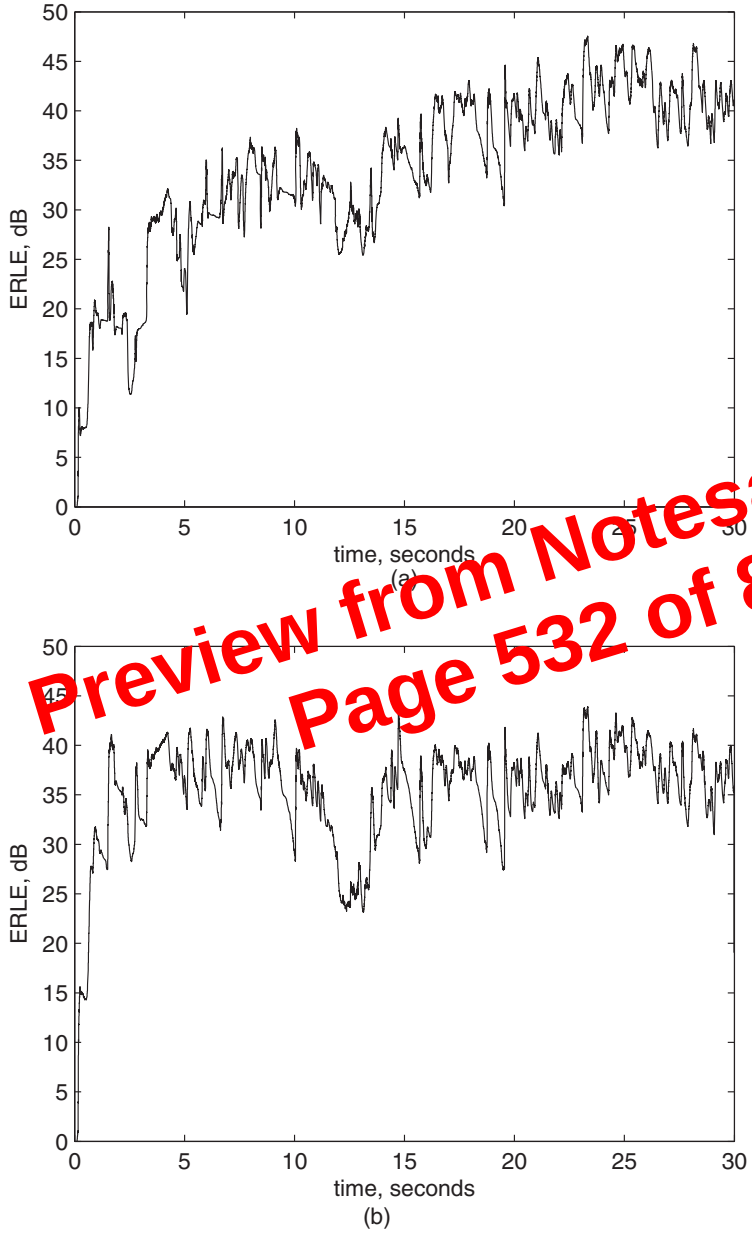

Figure 15.5 Echo return loss enhancement (ERLE) of the various algorithms: (a) NLMS, (b) APLMS, (c) FBLMS, (d) SBLMS, and (e) LMS-Newton.

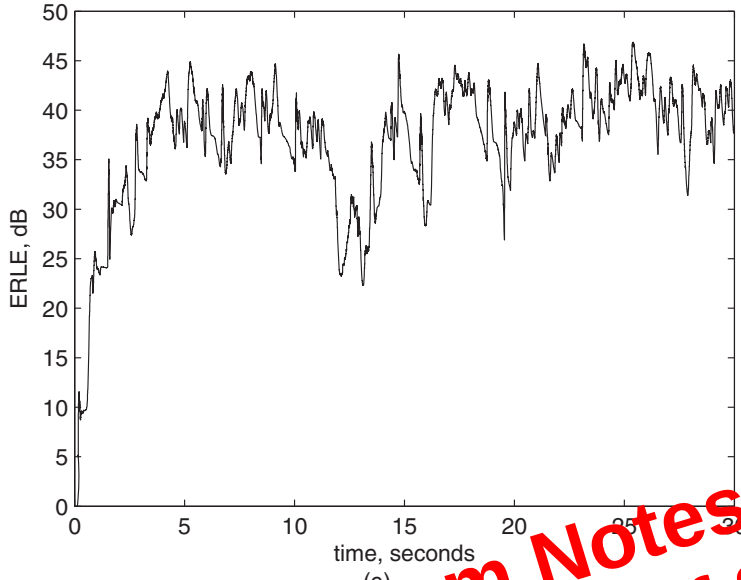

Figure 15.5 *(continued)*

Table 15.5 The parameters of the various algorithms that have been used for the results presented in Figure 15.5.

Algorithm	$\tilde{\mu}$	ψ	N	M	J	L	α_a	α_s	K_a	K_s	N_a	N_s	β	γ	ϵ	$\mu_{p,o}$
NLMS	0.5	0.1	1024	–	–	–	–	–	–	–	–	–	–	–	–	–
APLMS	0.5	0.1	1024	5	–	–	–	–	–	–	–	–	–	–	–	–
FBLMS	0.5	0.1	1024	64	–	64	–	–	–	–	–	–	–	–	–	–
SBLMS	0.5	0.1	1024	32	19	–	3/16	7/19	5	3	191	193	–	–	–	–
LMS-Newton	0.5	0.1	1024	8	–	–	–	–	–	–	–	–	0.98	0.9	0.05	0.001

accompanying website (www.wiley.com/go/adaptive_filters). For the results presented in Figure 15.5, the far-end signal is "`speech1.mat`." Here, we have kept the variable step-size part of the algorithms inactive; that is, we have used a fixed step-size parameter $\tilde{\mu}$. The parameters of the various algorithms that have been used for the results presented in Figure 15.5 are listed in Table 15.5. There is no near-end (double-talk) signal. However, a background Gaussian noise with a standard deviation of $\sigma_o = 0.001$ has been added at the microphone. This is about 45 dB below the near-end echoed signal that reaches the microphone. Also, as an alternative way of observing the performance of the various algorithms, the error (transmit) signal $e(n)$ produced by the examined algorithms, along with the microphone signal (i.e., before the echo cancellation), are presented in Figure 15.6.

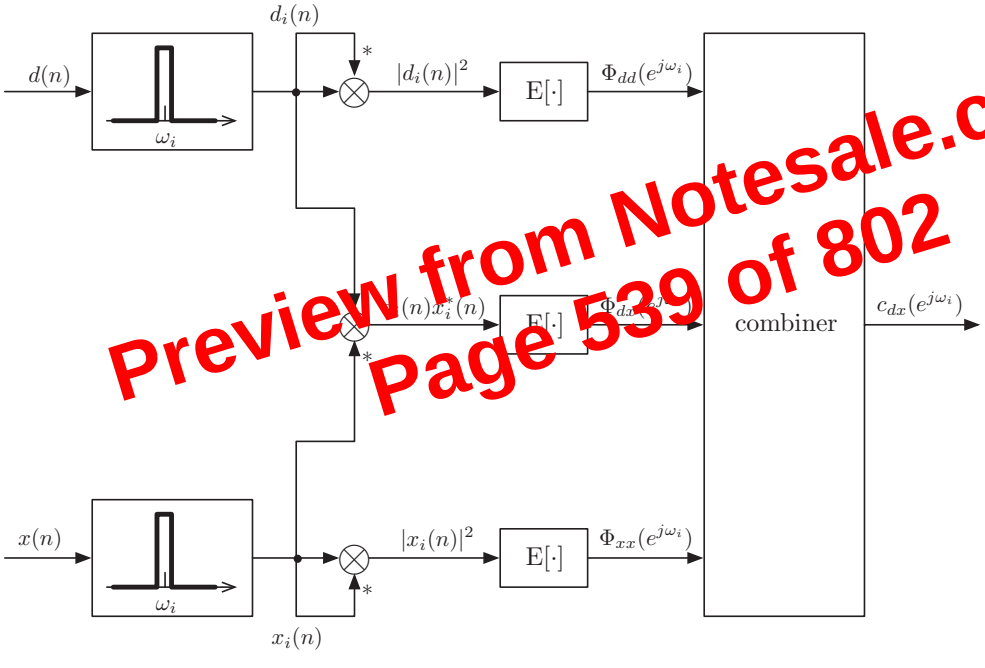

Figure 15.7 Procedure for calculation the coherence function $\hat{c}_{dx}(\mathrm{e}^{j\omega})$ at $\omega = \omega_i$.

nice and relevant to implementation property that half-band filters have is that almost half of their coefficients are 0. More specifically, $h_{\mathrm{HB},n} = 0$, for $n = \pm 2, \pm 4, \ldots$ This reduces the complexity of their implement by one-half. Once $H_{\mathrm{HB}}(e^{j\omega})$ is designed, it can be converted to the desired filter $H_{\mathrm{VSB}}(e^{j\omega})$ by modulating its coefficients with the sequence $\{j^n\} = \{\ldots, j, -1, -j, 1, j, -1, -j, \ldots\}$. Noting these points, given a half-band design $h_{\mathrm{HB},n}$, the coefficients of the desired filter $h_{\mathrm{VSB},n}$ are complex-valued, with the real and imaginary parts

$$h^{\mathrm{R}}_{\mathrm{VSB},n} = \begin{cases} h_{\mathrm{HB},n} & n = 0 \\ 0, & n \neq 0 \end{cases} \tag{15.56}$$

and

$$h^{\mathrm{I}}_{\mathrm{VSB},n} = \begin{cases} 0, & n = 0, \pm 2, \pm 4, \ldots \\ (-1)^{\frac{n-1}{2}} h_{\mathrm{HB},n}, & n = 1, 3, \ldots \end{cases} \tag{15.57}$$

respectively.

To help the reader to develop a better understanding of the impact of a spectral shift on the quality of speech signals, the MATLAB code "`specshift.m`," on the accompanying website, takes a speech signal, play it, then, introduce a shift in its spectrum and replay. The amount of spectral shift is a parameter. This will allow a curious reader to better understand what is the range of an acceptable spectral shift and why a maximum frequency shift 5 Hz has been imposed in the ITU-T Recommendation G.167.

15.5 Stereophonic Acoustic Echo Cancellation

The AE canceller setup that was presented in Figure 15.2 and was discussed in detail in the previous section is a monophonic system. Stereophonic AE cancellers have also been proposed and developed. Figure 15.17 presents the details of a stereophonic AE canceller setup. Compared to its monophonic counterpart, stereophonic teleconferencing provides special information that help a listener to distinguish the relative position of the speaker in the far-end teleconferencing room.

There are two microphones and two loudspeakers in each of the near-end and far-end teleconferencing rooms. Hence, considering the echo cancellation by the near-end room, there should be a pair of echo cancellers for removing echo signals from each of the microphones. A similar pair of echo cancellers should also be deployed by the far-end room. In Figure 15.17, for brevity and clarity of presentation, only one of the echo cancellers by the near-end room is shown. The relevant signals and signal paths/echoes are also shown. The speech signal, $s(n)$, from a person in the far-end room passes through the acoustic impulse responses $g_{1,n}$ and $g_{2,n}$ before reaching the pair of microphones in the far-end room. Assuming ideal transmission lines, one will find that

$$x_1(n) = g_{1,n} \star s(n) \tag{15.58}$$

and

$$x_2(n) = g_{2,n} \star s(n) \tag{15.59}$$

The signals $x_1(n)$ and $x_2(n)$, after being broadcast in the near-end room, will be picked up by the pair of microphones in the room. Here, the discussion is limited to the signal

Using Eq. (15.91), the misalignment of the leaky LMS algorithm when $\mathbf{w}(n)$ has approached $\hat{\mathbf{w}}_o$ is obtained as

$$\begin{aligned}\zeta &= (\mathbf{w}_o - \mathbf{R}'^{-1}\mathbf{R}\mathbf{w}_o)^T(\mathbf{w}_o - \mathbf{R}'^{-1}\mathbf{R}\mathbf{w}_o) \\ &= \mathbf{w}_o^T(\mathbf{I} - \mathbf{R}'^{-1}\mathbf{R})^2\mathbf{w}_o\end{aligned} \tag{15.92}$$

Recalling that according to the unitary singularity transformation, $\mathbf{R}$ and $\mathbf{R}'$ may be expanded as

$$\mathbf{R} = \mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^T \tag{15.93}$$

and

$$\mathbf{R}' = \mathbf{Q}\boldsymbol{\Lambda}'\mathbf{Q}^T \tag{15.94}$$

where $\boldsymbol{\Lambda}$ and $\boldsymbol{\Lambda}'$ are the diagonal matrices of the eigenvalues of $\mathbf{R}$ and $\mathbf{R}'$, respectively, and $\mathbf{Q}$ is the associated eigenvectors matrix. Also, note that $\mathbf{R}$ and $\mathbf{R}'$ share the same set of eigenvectors. Substituting Eqs. (15.93) and (15.94) in Eq. (15.92), recalling the identity $\mathbf{Q}^T\mathbf{Q} = \mathbf{I}$ and the definition

$$\mathbf{Q} = [\mathbf{q}_0 \ \mathbf{q}_1 \cdots \mathbf{q}_{2N-1}] \tag{15.95}$$

and rearranging the result, we obtain

$$\zeta = \left(\frac{1-\beta}{2\mu}\right)^2 \times \sum_{i=0}^{2N-1} \frac{\mathbf{q}_i^T\mathbf{w}_o^2}{\left(\lambda_i + \frac{1-\beta}{2\mu}\right)^2} \tag{15.96}$$

This result shows that, on one hand, we should choose β sufficiently smaller than 1 ($1-\beta$ large) to avoid slow modes of convergence. On the other hand, we should choose β sufficiently close to 1 ($1-\beta$ small) to reduce the misalignment.

Coherence Reduction Through Exclusive Maximum Tap-Selection

The exclusive maximum (XM) tap-selection method is an adaptation strategy that is applicable to the various adaptive algorithms, including NLMS, affine projection LMS, and RLS algorithms. The basic idea is to apply each adaptation step only to those taps with the larger amplitude tap inputs. Khong and Naylor (2005, 2007) have studied the application of the XM tap-selection method to stereophonic AE cancellers and noted that this method has some success in reducing the misalignment. The rationale here is that at each iteration of the adaptive algorithm, the set of tap inputs selected from the two channels are randomly unaligned, and this results in a lower coherency among the signals from the two channels that are used for adaptation.

Coherence Reduction Through Nonlinearity

It has been noted that introduction of some nonlinearity into a speech signal to a great degree is perceptually undetectable. On the other hand, addition of such nonlinearity reduces the coherence between a pair of signals in a stereophonic AE canceller, and thus resolves the problem of a large misalignment. Morgan, Hall, and Benesty (2001)

3. Compute the elements that are the delayed versions of the $(2M+1)$th and $(2M+2)$nd elements of $\mathbf{u}(n)$ from the previous iteration as

$$u_{1,j}(n) = u_{1,j-1}(n-1)\,;\;\; j = M+1, M+2, \ldots, L-M-1$$
$$u_{2,j}(n) = u_{2,j-1}(n-1)\,;\;\; j = M+1, M+2, \ldots, L-M-1$$

   Note $\mathbf{u}_j(n) = [u_{1,j}(n)\; u_{2,j}(n)]^{\mathrm{T}}$ is the jth 2×1 vector component of $\mathbf{u}(n)$.
4. Compute the first $2(M+1)$ elements of $\mathbf{u}(n)$ as

$$[u_{1,0}(n), u_{2,0}(n), u_{1,1}(n), u_{2,1}(n), \ldots, u_{1,M}(n), u_{2,M}(n)]^{\mathrm{T}} = \mathbf{L}_{\mathrm{tl}}^{\mathrm{T}}\mathbf{b}_{\mathrm{h}}(n)$$

   where

$$\mathbf{b}_{\mathrm{h}}(n) = \begin{bmatrix} \mathbf{R}_{b_0b_0}^{-1}(n)\mathbf{b}_0(n) \\ \mathbf{R}_{b_1b_1}^{-1}(n)\mathbf{b}_1(n) \\ \vdots \\ \mathbf{R}_{b_Mb_M}^{-1}(n)\mathbf{b}_M(n) \\ \mathbf{R}_{b_Mb_M}^{-1}(n)\mathbf{b}_M(n-1) \\ \vdots \\ \mathbf{R}_{b_Mb_M}^{-1}(n)\mathbf{b}_M(n-M) \end{bmatrix}$$

   and $\mathbf{L}_{\mathrm{tl}}$ denotes the top-left part of $\mathbf{L}$ having dimension $2(2M+1) \times 2(M+1)$.
5. Compute the last $2M$ elements of $\mathbf{u}(n)$ as

$$[u_{1,(L-M)}(n), u_{2,(L-M)}(n), \ldots, u_{1,L-1}(n), u_{2,L-1}(n)]^{\mathrm{T}} = \mathbf{L}_{\mathrm{br}}^{\mathrm{T}}\mathbf{b}_{\mathrm{t}}(n)$$

   where

$$\mathbf{b}_{\mathrm{t}}(n) = \begin{bmatrix} \mathbf{R}_{b_Mb_M}^{-1}(n-L+M+1)\mathbf{b}_M(n-L+2M) \\ \mathbf{R}_{b_Mb_M}^{-1}(n-L+M+1)\mathbf{b}_M(n-L+2M-1) \\ \vdots \\ \mathbf{R}_{b_Mb_M}^{-1}(n-L+M+1)\mathbf{b}_M(n-L+M+1) \end{bmatrix}$$

   and $\mathbf{L}_{\mathrm{br}}$ is the bottom-right part of $\mathbf{L}$ having dimension $2M \times 2M$, then the last $2M$ elements of $\mathbf{u}(n)$ are
6. Finally, compute the adaptive filter output $\hat{y}(n) = \mathbf{w}^{\mathrm{T}}(n)\mathbf{x}(n)$, the error signal $e(n) = d(n) - \hat{y}(n)$ and update the filter taps using Eq. (15.114).

To implement the lattice predictor using equations, we require $25M+5$ multiplications. The two-channel Levinson-Durbin algorithm requires $8M(M-1)$ multiplications. We further need $6M^2+26M+8$ multiplications to update $\mathbf{u}(n)$. Finally, $4N$ multiplications are required to adaptively update the transversal filter coefficients. Hence, in order to implement the fast LMS-Newton Algorithm 1, we require a total of $14M^2 + 43M + 13 + 4N$ multiplications. The number of the required additions is about the same.

in a two-channel lattice. Thus, some simplifications that are applicable to single-channel lattice equations are inapplicable to the two-channel case. Consequently, direct mimicking of the results of Chapter 11 is not possible here. This is why we present a fresh derivation of Algorithms 1 and 2.

Typically, M can take a value of 8 and the adaptive filter length N may be 1500, for a medium size office room. With these numbers, each update of $\mathbf{u}(n)$ would make up only 17% of the total computational complexity of the AE canceller.

Algorithm 2: The two-channel LMS-Newton Algorithm 1 is structurally complicated despite having reasonably low-computational complexity. Manipulating the data is not all that straightforward and hence it is more suitable for implementing using software. We now propose an alternate algorithm that is computationally less complex and can be easily implemented in hardware.

If we look at the $\mathbf{L}$ matrix given in Eq. (15.116), we observe that only the first $2(M+1)$ rows of this matrix are uniquely represented. The remaining rows are just the delayed versions of the $(2M+1)$th and $(2M+2)$nd rows given by

$$[-\mathbf{G}_{M,1} \quad -\mathbf{G}_{M,2} \quad \cdots \quad -\mathbf{G}_{M,M} \quad \mathbf{I} \quad 0 \quad \cdots \quad 0]$$

As discussed in Chapter 11, for the single-channel case, Algorithm 1 may be greatly simplified by extending the input and tap-weight vectors $\mathbf{x}(n)$ and $\mathbf{w}(n)$ to the followings vectors

$$\mathbf{x}_{\mathrm{E}}(n) = [x_1(n+M), x_2(n+M), \ldots, x_1(n+1), x_2(n+1), x_1(n), x_2(n), \ldots,$$
$$x_1(n-L+1), x_2(n-L+1), \ldots, x_1(n-L-M+1), x_2(n-L-M+1)]^{\mathrm{T}} \quad (15.120)$$

and

$$\mathbf{w}_{\mathrm{E}}(n) = [w_{1,-M}(n), w_{2,-M}(n), \ldots, w_{1,-1}(n), w_{2,-1}(n), w_{1,0}(n), w_{2,0}(n), \ldots,$$
$$w_{1,L-1}(n), w_{2,L-1}(n), \ldots, w_{1,L+M-1}(n), w_{2,L+M-1}(n)]^{\mathrm{T}} \quad (15.121)$$

respectively, and applying Eq. (15.114) to update the extended tap-weight vector $\mathbf{w}_{\mathrm{E}}(n)$. We also need to appropriately take care of the dimensions of $\mathbf{L}$ and $\mathbf{R}_{bb}$. Moreover, because we are interested only in the tap-weights corresponding to $w_{1,0}(n)$, $w_{2,0}(n)$, $w_{1,1}(n)$, $w_{2,1}(n)$, $\ldots$, $w_{1,L-1}(n)$, $w_{2,L-1}(n)$, the first $2M$ and the last $2M$ elements of the extended tap-weight vector can be permanently set to zero. This will also remove the computation of the first $2M$ and the last $2M$ elements of $\mathbf{L}^{\mathrm{T}}\mathbf{R}_{bb}^{-1}\mathbf{L}\mathbf{x}_{\mathrm{E}}(n)$. This leads to the following update equation:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu\mathbf{u}_{\mathrm{a}}(n)e(n) \quad (15.122)$$

where

$$\mathbf{u}_{\mathrm{a}}(n) = \mathbf{L}_2\mathbf{R}_{bb}^{-1}(n)\mathbf{L}_1\mathbf{x}_{\mathrm{E}}(n) \quad (15.123)$$

Also, $\mathbf{L}_1$ is a $2(L+M) \times 2(L+2M)$ matrix defined as

$$\mathbf{L}_1 = \begin{bmatrix} -\mathbf{G}_{M,1} & -\mathbf{G}_{M,2} & \cdots & \mathbf{I} & \mathbf{0} & \cdots & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -\mathbf{G}_{M,1} & \cdots & -\mathbf{G}_{M,M} & \mathbf{I} & \cdots & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \cdots & -\mathbf{G}_{M,1} & \cdots & -\mathbf{G}_{M,M} & \mathbf{I} \end{bmatrix} \quad (15.124)$$

The results presented in Figure 15A.2 have been obtained using the MATLAB program "`prolates.m`," available on the accompanying website. Interested readers are encouraged to run this program for different choices of the parameters to see their effects.

Polyphase Filter Banks

According to the discussion in Section 15.3.3, an implementation of the multitaper method requires realization of a multiple set of filter banks. An example of such filter banks was presented in Figure 15.14. We also recall that a filter bank is naturally implemented based on a prototype filter. Here, we consider a filter bank with a prototype filter

$$H_0(z) = H(z) = \sum_{n=0}^{M-1} h_n z^{-n} \tag{15A.9}$$

Here, we are interested in implementing the set of filters

$$H_m(z) = \sum_{n=0}^{M-1} h_n e^{j\frac{2\pi mn}{L}} z^{-n} \tag{15A.10}$$

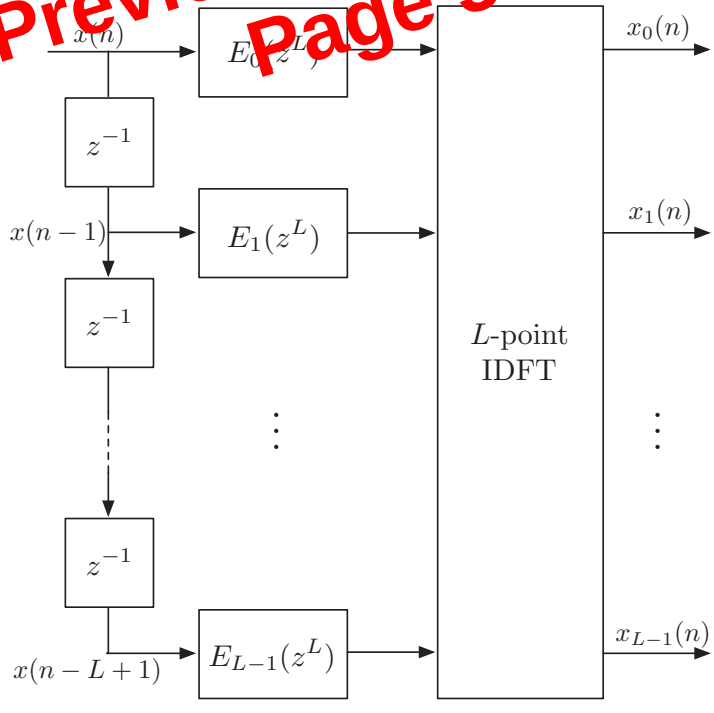

Figure 15A.3 Polyphase filter bank.

16

Active Noise Control

Acoustic noise occurs in different forms in variety of environments. Clearly, such noises are undesirable and methods of reducing them are always sought. Conventionally, the use of passive materials to absorb an acoustic noise or contain it within an enclosure has been considered. However, these methods generally found unsuccessful is suppressing the low frequency portions of acoustic signals. This is because for an absorber to be effective its thickness should be comparable or larger than the wavelength of the acoustic signal that it meant to be suppressed. Moreover, at low frequencies the acoustic wavelength grows very fast. For instance, while, in air, the wavelength of a 5 kHz acoustic signal is 6.8 cm, it increases to 3.4 m at 100 Hz.

The term *active noise control* (ANC) refers to the methods where acoustic (and, also, hydroacoustic) noise signals are canceled using a combination of electromechanical systems. ANC methods are built based on the fact that an acoustic signal/pressure can be nullified by introducing an acoustic pressure of opposite direction. To put this in a right prospect and also to be able to discuss the advantages as well as the limitations of ANC methods, in this chapter, we present two (important) examples of ANC systems.

The first case of interest is the problem of noise cancellation in air ducts, for example, air conditioning ducts and exhaust pipes in cars. This case is often exemplified by the noise cancellation setup that was presented in Figure 1.20 and for convenience of reference is repeated here in Figure 16.1.

The second case that we consider is when a primary acoustical source has generated a signal (noise) that reaches a point P in space with a pressure of $p(t)$. A microphone measures this pressure and through an ANC filter instructs a secondary source (a canceling loudspeaker) at a point near P to broadcast the same signal, but with an opposite sign. This concept is presented in Figure 16.2. We note that when the primary source is at some location far away from P, the cancellation will be limited to the points near P, that is, a silent zone around the error microphone is generated. This is because the acoustic pressure resulting from the secondary source vanishes in space relatively fast, while that of the primary source remains nearly intact at the points surrounding P. This may be understood, if one notes that the sound pressure resulting from a source in free space at a distance r is proportional to r^{-1}.

The setups presented in Figures 16.1 and 16.2 are different in a number of ways. While the setup in Figure 16.2 is that of an acoustic noise cancellation in a three-dimensional

Adaptive Filters: Theory and Applications, Second Edition. Behrouz Farhang-Boroujeny.

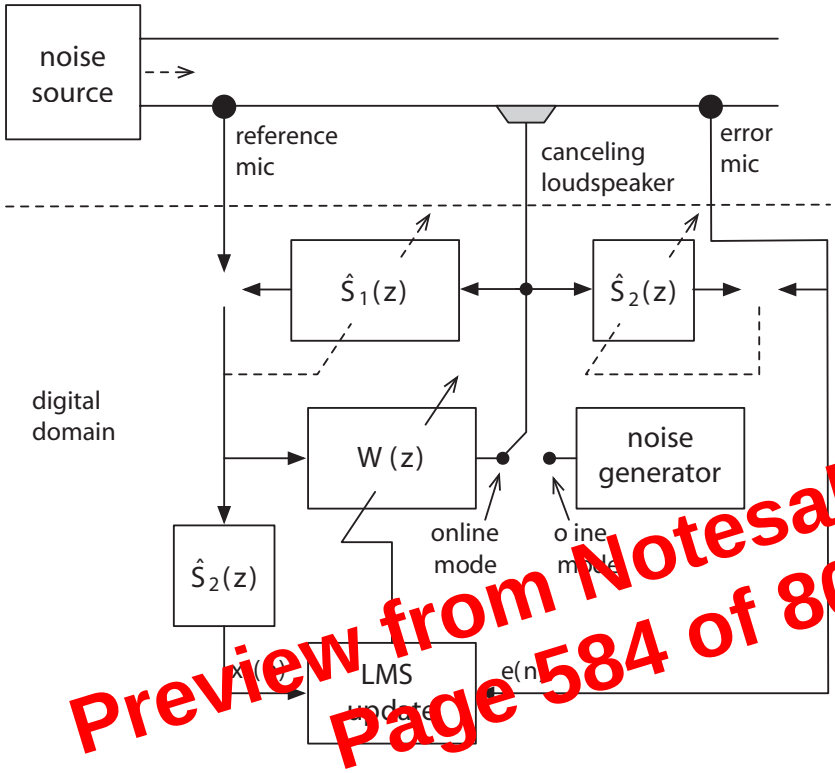

Figure 16.9 Complete ANC system for an air duct.

filter (the ANC filter) which shapes its spectrum to match that of the noise amplitude and phase (with an opposite sign) at the canceling loudspeaker. The error microphone picks the residual noise and instructs the adaptation of the ANC filter.

Comparing Figure 16.10 with its broadband counterpart in Figure 16.1, one may observe the following fundamental differences. In Figure 16.1, the broadband noise is picked up by the reference microphone. In Figure 16.10, on the other hand, the period/frequency of the periodic noise is obtained through a nonacoustic device. A reference signal is then generated accordingly. In both cases (Figures 16.1 and 16.10), the ANC filter shapes the spectrum of the reference signal such that a correct anti-noise is generated at the canceling loudspeaker. However, the feedback from the canceling loudspeaker to the reference microphone, which can be a major source of instability, has no counterpart in the case of narrowband ANC systems. Hence, the narrowband ANC systems are less prone to instability problems and thus are easier to design and implement.

16.2.1 Waveform Synthesis Method

The waveform synthesis method constructs a sign-inverted version of the periodic noise, viz., the anti-noise, at the canceling loudspeaker through the following mechanism. The ANC filter is chosen to be an FIR filter with a length equal to one period of the

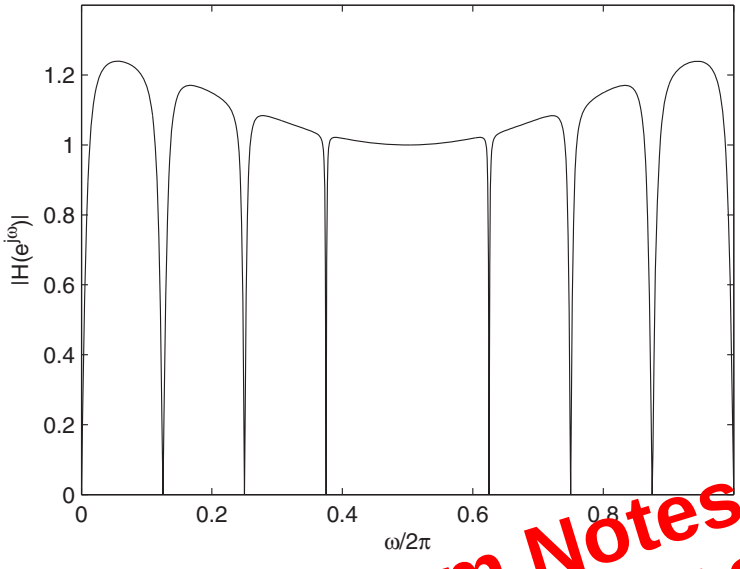

Figure 16.17 An example of magnitude response of the multinotch filter $W(z)$ for $N = 8$ and $\mu = 0.2$, and in the presence of a secondary path $S_2(z) = 0.5 + (0.5)z^{-1}$

counterpart of Figure 16.14, when a secondary path $S_2(z) = 0.5 + 0.5z^{-1}$ has been added. To explain the magnitude response plot of Figure 16.17, we note that, here,

$$|S_2(e^{j\omega})|^2 = \cos^2\omega \tag{16.42}$$

Moreover, we note that $\cos^2\omega$ is equal to 1 at $\omega = 0$ and decreases as ω varies between 0 and π. It reaches a minimum of zero at $\omega = \pi$, and increases toward 1, as ω varies between π and 2π. The variation of $S_2(e^{j\omega})$ has the following impact on the magnitude response of $H(z)$. Near zero, where $|S_2(e^{j\omega})|^2 \approx 1$, there is very little difference between the plot in Figure 16.17 and its counterpart in Figure 16.14. On the other hand, as ω approaches π, thus, $|S_2(e^{j\omega})|^2$ decreases, the bandwidths of the notches decrease. At $\omega = \pi$, where $|S_2(e^{j\omega})|^2 = 0$, the notch disappears from the response. This observation shows that when the secondary path $S_2(z)$ has a null at any of the harmonics of the noise, the ANC system discussed in this section is incapable of removing that harmonic.

Synchronization

The underlying assumption made in the above derivations was that the rate of the input samples to the ANC filter $W(z)$ was synchronized with the fundamental frequency of the noise such that each period of the noise was exactly equal to N samples. Such synchronization can be achieved in practice by a proper design of the tachometer. The tachometer should be designed so that it generates N pulses per each full rotation of the noise generating rotor, for example, by placing N equally spaced sensors (magnets) around the shaft of the rotor.

its sampled version $x(n) = \sin(\Omega n T_s) = \sin(\omega n)$. One may also note that the absolute Ω and the normalized ω frequencies are related according to the equation

$$\omega = \frac{\Omega}{f_s} \tag{17.1}$$

17.1 Continuous Time Channel Model

In the study of adaptive equalizers, the channel is often modeled in its equivalent baseband and discrete time form, even though the actual transmission happens over a passband radio frequency (RF) and in continuous time form. In this section, we present a development of such a channel model.

Figure 17.1 presents the block diagram of a digital quadrature-amplitude modulation (QAM) communication system. The transmit data symbols, $s(n)$, are streamed to a sequence of impulses at the interval of T. We refer to T as *symbol-spacing*. This is presented by the continuous time signal

$$s(t) = \sum_n s(n)\delta(t - nT) \tag{17.2}$$

where $s(n)$ are data symbols from a QAM constellation.

In Figure 17.1, the sequence of the symbol modulated impulses, represented by the continuous time signal $s(t)$, is passed through a transmit pulse-shaping filter with the impulse response $p_T(t)$. This results in a band-limited baseband signal, which subsequently modulates a carrier with the frequency Ω_c. Multiplication by $e^{j\Omega_c t}$ shifts the baseband signal spectrum to the carrier band, and the block $\Re[\cdot]$ takes the real part of the result, thus produces a duplicate image spectrum around $-\Omega_c$ as well. This leads to

$$X_{\text{QAM}}(\Omega) = S(\Omega - \Omega_c)P_T(\Omega - \Omega_c) + S^*(-\Omega - \Omega_c)P_T^*(-\Omega - \Omega_c) \tag{17.3}$$

where $X_{\text{QAM}}(\Omega)$ is the Fourier transform of $x_{\text{QAM}}(t)$. Using Eq. (17.3), we obtain

$$X'_{\text{QAM}}(\Omega) = \big(S(\Omega - \Omega_c)P_T(\Omega - \Omega_c) + S^*(-\Omega - \Omega_c)P_T^*(-\Omega - \Omega_c)\big)\,C(\Omega) \tag{17.4}$$

where $X'_{\text{QAM}}(\Omega)$ and $C(\Omega)$ are the Fourier transforms of $x'_{\text{QAM}}(t)$ and $c(t)$, respectively. At the receiver side, the multiplication of $x'_{\text{QAM}}(t)$ by $e^{-j\Omega_c t}$ shifts the spectrum to the

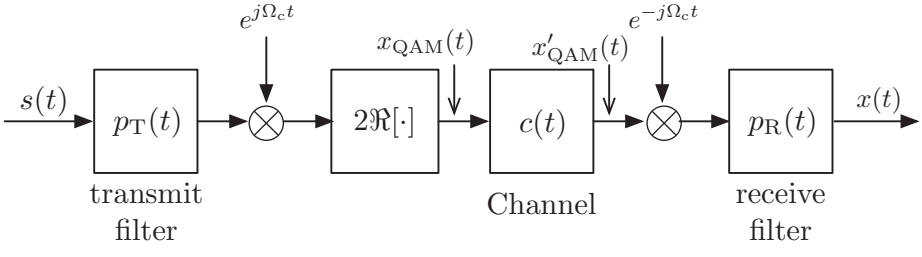

Figure 17.1 Block diagram of the channel model in a digital QAM communication system.

$$x(n) = \sum_m s(m)h(n - mL) + \nu(n) \tag{17.15}$$

where $\nu(n)$ is the channel noise.

In a digital communication system, the samples $x(n)$ are passed through an adaptive equalizer whose coefficients are adjusted to undo the distortion caused by the channel and thus deliver (almost) ISI free estimates of $s(n)$ at its output. Considering the channel equation (17.15), and adding the equalizer after the channel model, the discrete-time system model of a digital communication system is obtained as the one presented in Figure 17.5. The interpolator block adds $L - 1$ zeros after each symbol $s(n)$, changing the sample rate from $1/T$ to L/T. Passing the upsampled sequence through $H(z)$ and adding the channel noise, $\nu(n)$, is to implement Eq. (17.15). The equalizer $W(z)$ is an FIR filter with the tap weights w_0, w_1, ..., w_{N-1}, and the estimates $\hat{s}(n - \Delta)$ of the transmitted data symbols, where Δ is the delay caused by the combination of the channel and equalizer, are obtained by taking the decimated samples at the output of $W(z)$.

17.2.1 Symbol-Spaced Equalizer

If one chooses to use an equalizer whose tap spacing is equal to the spacing T of the transmitted symbols, Figure 17.5 reduces to Figure 17.6. This corresponds to the case where the interpolation and decimation factor L is equal to 1. It may be noted that the system setup in Figure 17.6 is similar to the equalizer setup that was presented in Chapter 6 (Section 6.4.2). Because of obvious reasons, in this setup, $W(z)$ is referred to as a *symbol-spaced equalizer*.

It is also worth noting that because the transmit signal has a transmission bandwidth that is larger than $1/T$ (e.g., when $p_T(t)$ is square-root raised-cosine pulse shape with a

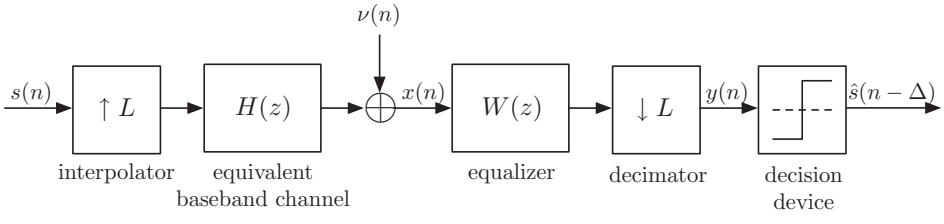

Figure 17.5 The discrete-time system model of a digital communication system.

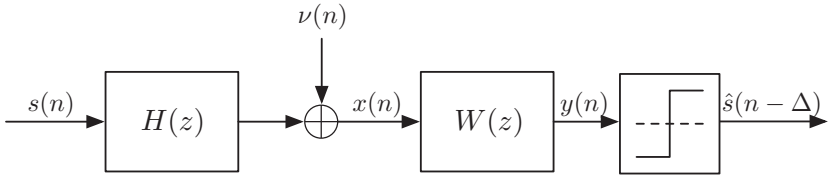

Figure 17.6 A digital communication system with a symbol-spaced equalizer.

roll-off factor α, the transmission bandwidth is $(1+\alpha)/T$), the sampled signal $x(n)$ (the equalizer input) is subject to aliasing. This, as discussed later (Section 17.4), may result in a poor performance if the input signal to the equalizer is sampled at a timing phase that results in a significant spectral attenuation within the aliased band.

17.2.2 Fractionally Spaced Equalizer

As it was noted earlier, the symbol-spaced equalizers are subject to aliasing and an improper choice of the timing phase of the samples to the equalizer may lead to significant performance degradation. This problem is avoided by adopting an equalizer whose input is sampled at a rate above the Nyquist. Such equalizers are referred to as *fractionally spaced*. As, in practice, $0 < \alpha \leq 1$, an equalizer tap-spacing $T/2$ is always sufficient to avoid aliasing. However, when $\alpha < 1$, one may set the tap-spacing equal to some value between $T/2$ and T. For example, when $\alpha = 0.5$, an equalizer tap-spacing $2T/3$ is sufficient to avoid aliasing. Also, in this case, one may prefer the spacing $2T/3$ over $T/2$, because for a fixed time span of the equalizer, the choice of $2T/3$ results in 4/3 $(= (2/3)/(1/2))$ times less number of tap weights.

Implementation of a fractionally spaced equalizer with the tap-spacing $T/2$ is straightforward and follows the block diagram of Figure 17.5, with $L = 2$. Figure 17.7 presents the detail of implementation of this equalizer. The equalizer taps are at the spacing $T/2$. Moreover, as the equalizer output is decimated two-fold, the equalizer output needs to be calculated only for even values of n.

When the tap-spacing is a less trivial fraction of T, for example, it is equal to KT/L, for a pair of integers K and L, the equalizer detail is only slightly more involved than that in Figure 17.7. The detail of a fractionally spaced equalizer with the tap-spacing $2T/3$ is presented in Figure 17.8. This should serve as an example that can be easily extended to other cases as well. Here, the samples $x(n)$ are spaced at $T/3$, and the tap-spacing $2T/3$ is achieved by replacing each single delay block z^{-1} by a doubly delay block z^{-2}. Also, samples of output are calculated for values of n that are integer multiple of 3.

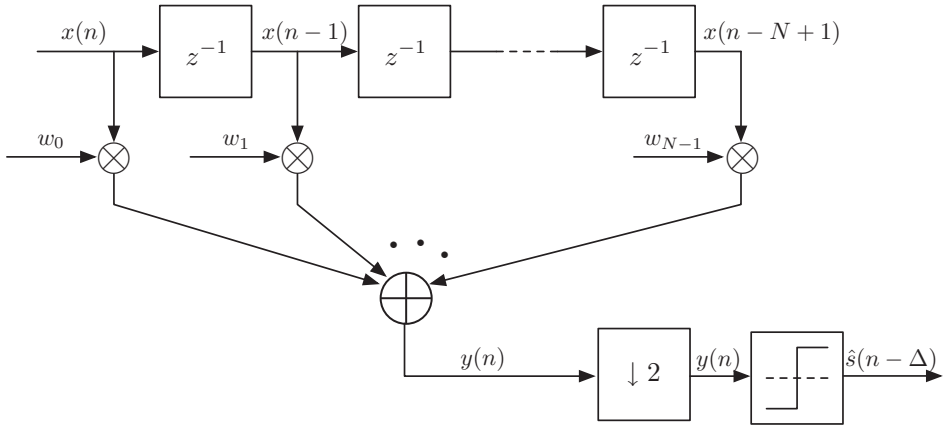

Figure 17.7 Details of a fractionally spaced equalizer with the tap-spacing $T/2$.

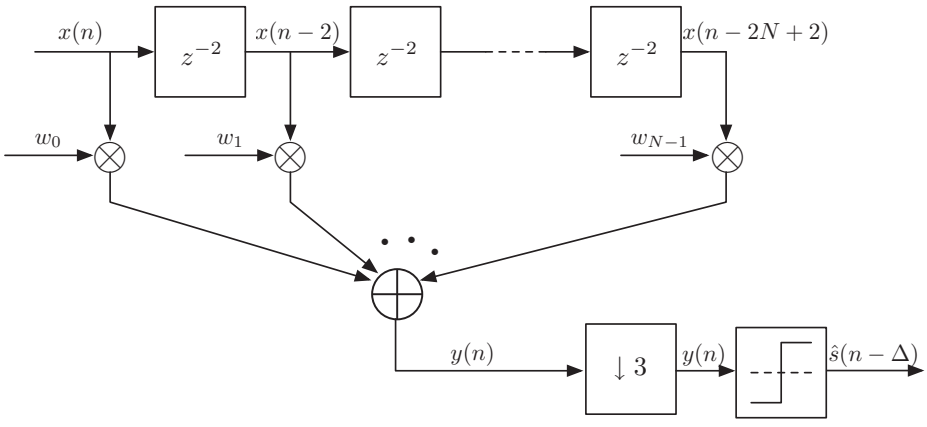

Figure 17.8 Details of a fractionally spaced equalizer with the tap-spacing $2T/3$.

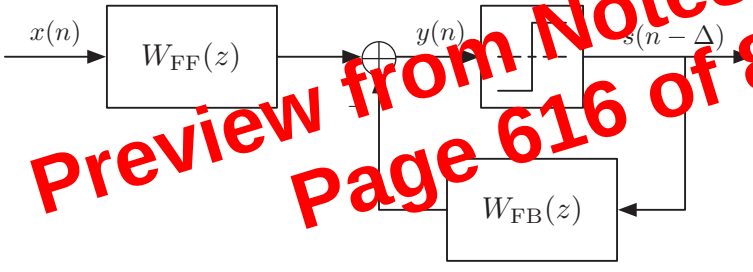

Figure 17.9 Block diagram of a decision feedback equalizer.

17.2.3 Decision Feedback Equalizer

The symbol-spaced and the fractionally spaced equalizer structures that were presented earlier belong to the class of linear equalizers/filters. The decision feedback (DF) equalizers, on the other hand, belong to the class of nonlinear filters. Figure 17.9 presents a block diagram of a DF equalizer. It consists of a feedforward filter $W_{\mathrm{FF}}(z)$ and a feedback filter $W_{\mathrm{FB}}(z)$. The feedforward filter $W_{\mathrm{FF}}(z)$ is a linear (transversal) filter and can be a symbol-spaced or fractionally spaced one. It equalizes the channel so that its response up to the decision device has no precursor ISI, but may contain some postcursor ISI. The postcursor ISI is canceled by passing the output of the decision device through the feedback filter with the transfer function

$$W_{\mathrm{FB}}(z) = \sum_{i=1}^{M} w_{\mathrm{FB},i} z^{-i} \tag{17.16}$$

where $w_{\mathrm{FB},i}$ are the feedback tap weights and M is the number of feedback taps.

When the channel contains some deep fade(s) in its amplitude response, a linear equalizer has to compensate for the fade(s). This, along with amplifying the desired (data)

find the power spectral density of the channel output as

$$\Phi_{xx}(e^{j\omega}, \tau) = |H(e^{j\omega}, \tau)|^2 \tag{17.33}$$

Also, recalling Eq. (2.60),

$$\begin{aligned}\rho(\tau) &= \frac{1}{2\pi}\int_{-\pi}^{\pi} \Phi_{xx}(e^{j\omega}, \tau)d\omega \\ &= \frac{1}{2\pi T^2}\int_{-\pi}^{\pi} |H(e^{j\omega}, \tau)|^2 d\omega \end{aligned} \tag{17.34}$$

where $\rho(\tau) = E[|x(n)|^2]$ denotes the signal power at the equalizer input for a timing phase τ. Recall that this is the timing recovery cost function that was defined earlier.

Next, considering Eq. (17.34), one may find that for the present case, where the channel is ideal, $\rho(\tau)$ finds its maximum value at $\tau = 0$ and it reduces as τ varies from 0 to $T/2$, reaching its minimum at $\tau = T/2$. This can be easily understood if one notices from Figure 17.11 that the area under $|H(e^{j\omega}, \tau)|^2$ is at its maximum when $\tau = 0$ and reduces as τ varies from 0 to $T/2$. Moreover, it may be noted that when $\tau = 0$, $h(n, \tau)$ reduces to an impulse in the discrete time. Hence, the input to the equalizer is free of ISI, and there is no need for equalization. When $\tau \neq 0$, the dent in the magnitude of $|H(e^{j\omega}, \tau)|$ has to be compensated for by an equalizer with a gain larger than one around $\omega = \pm\pi$. This, clearly, results in some noise enhancement, and thus, results in a poorer performance of the receiver. We may hence conclude that when the channel is ideal, the optimum timing phase is the one that avoids any attenuation in the aliased response $H(e^{j\omega}, \tau)$ of the channel. One may imagine that this conclusion is true even in the cases where the channel is nonideal, viz., a choice of timing phase that results in a significant attenuation of $H(e^{j\omega}, \tau)$, over the aliased portion of the spectrum, may lead to a significant noise enhancement, thus a poor performance of the receiver. However, unfortunately, as the numerical results presented in Section 17.4.2 show, the maximization of the cost function $\rho(\tau)$ does not necessarily avoid nulls in the aliased spectra and hence cannot always avoid possible poor performance of the symbol-spaced equalizers.

17.3.3 Improving the Cost Function

Figure 17.13 presents plots of the cost function $\rho(\tau)$ for an ideal channel where $H(\Omega) = P(\Omega)$ and $P(\Omega)$ is the Fourier transform of a raised-cosine pulse shape. The plots are given for three values of the roll-off factor $\alpha = 0.25$, 0.5, and 1. An important point to note here is that the variation of $\rho(\tau)$ with τ reduces as α decreases. Also, in an adaptive setting, a stochastic gradient (similar to the one in LMS algorithm) is used to search for the timing phase that maximizes $\rho(\tau)$. In addition, we note that the variance of a stochastic gradient is approximately proportional to the magnitude of the underlying signal power. Relating these points, one may argue that the stochastic gradients used for timing recovery become less reliable as α decreases.

```
%%% Receiver filtering  %%%
pR=pT; x=conv(xbbR,pR);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Early-late gate timing recovery %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
beta=0;              % Should be changed for the modified algorithm
mu0=0.005;           % Step-size parameter
dtau=12;             % (Δ tau) times L
mu=mu0*(L/4)/dtau;   % Adjusted step-size
kk=1;xp=0;xm=0;
start=5*L+1;         % To drop the transient of x(t) at the beginning
tau=0.3*ones(1,floor((length(x)-start)/L)); % Initialize the timing offset
            % The timing offset tau is adjusted as the algorithm proceeds.
for k=start:L:length(tau)*L
    tauT=round(tau(kk)*L);
    xp=sqrt(1-beta^2)*x(k+tauT+dtau)-beta*xp;
    xm=sqrt(1-beta^2)*x(k+tauT-dtau)-beta*xm;
    tau(kk+1)=tau(kk)+mu*(abs(xp)^2-abs(xm)^2);
    kk=kk+1;
end
figure, axes('position',[0.1 0.25 0.7 0.5]), plot(tau,'k')
xlabel('Iteration Number, n'), ylabel('\tau(n)')
```


Figure 17.16 presents a set of plots of variation of τ as the early-late gate timing recovery operates according to the above MATLAB script. The results are for choices of $\delta\tau = T/L$, $10T/L$, $20T/L$, and $25T/L$ (respectively, dtau = 1, 10, 20, and 25, in the MATLAB script). From these results, and further experiments that may be performed,

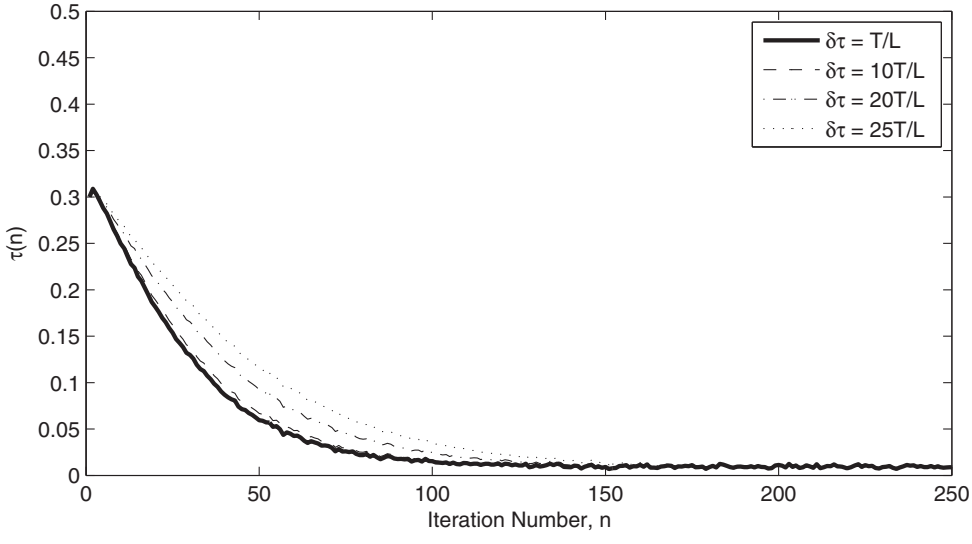

Figure 17.16 Plots of the timing phase update of the early-late gate timing recovery algorithm. The parameters used are $\beta = 0$, $\mu_0 = 0.01$, and four choices of $\delta\tau$. Each plot is an ensemble average of 100 independent runs.

stability problem, especially for channels that are poorly conditioned, that is, the cases where the eigenvalue spread of the underlying correlation matrix is large.
- The intermediate algorithms, such as the affine projection algorithm (of Section 6.7) and the LMS-Newton algorithm (of Section 11.15) may thus be good compromised choices for adaptation of channel equalizers.
- For the fractionally spaced equalizers, one should take note of the following point. The input signal to the equalizer is oversampled above its Nyquist rate. This means that the power spectral density of the input signal to the equalizer over higher portion of its frequency band reaches zero. This, in turn, translates to the fact that the underlying correlation matrix is poorly conditioned. Hence, the problems mentioned in the first two items above may be more pertinent in the case of the fractionally spaced equalizers.

17.6 Cyclic Equalization

Training symbols that are known to the receiver are often used at the beginning of a communication session to synchronize the receiver carrier and symbol clock to the incoming signal and to adjust the equalizer coefficients to a point near their optimal values. It turns out that if the training sequence is selected to be periodic and have a period equal to the length of the equalizer, the equalizer tap weights can be obtained almost instantly. Moreover, when such training sequences are used, simple mechanism can be developed for fast acquisition of the carrier frequency and symbol clock of the received signal. Furthermore, any phase offset in the carrier will be taken care of by the equalizer. In addition, the use of the periodic training sequences allows adoption of a simple mechanism for selection of the time delay Δ and, thus, alignment of the data symbol sequences between transmitter and receiver.

17.6.1 Symbol-Spaced Cyclic Equalizer

Let the periodic sequence $\ldots, s(N-1), s(0), s(1), s(2), \ldots, s(N-1), s(0), \ldots$ be transmitted through a channel with the symbol-spaced complex baseband equivalent response $h(n)$. Ignoring the channel noise, this periodic input to the channel results in a periodic output $x(n)$ with the same period. Now consider an equalizer setup with the input $x(n)$ and desired output $s(n)$. As, here, $s(n)$ and $x(n)$ are periodic, one may pick a period of samples of $x(n)$, with an arbitrary starting point, and put them in a tapped-delay line/shift register with its output connected back to its input. A similar shift register is also used to keep one cycle of $s(n)$. An equalizer whose tap weights are adjusted to match its output $y(n)$ with $s(n)$ is then constructed as shown in Figure 17.25.

The adaptation algorithm in Figure 17.25 can be any of the known adaptive algorithms, including the LMS, NLMS, APLMS or RLS. Also, once a cycle of $x(n)$ is received, the adaptation process can begin and run for sufficient number of iterations for the equalizer to converge. To put this in a mathematical framework, we define the column vectors

$$\mathbf{x}_0 = [x(n)\ x(n-1)\cdots x(n-N+1)]^{\mathrm{T}}$$

and

$$\mathbf{w} = [w_0\ w_1\cdots w_{N-1}]^{\mathrm{H}}$$

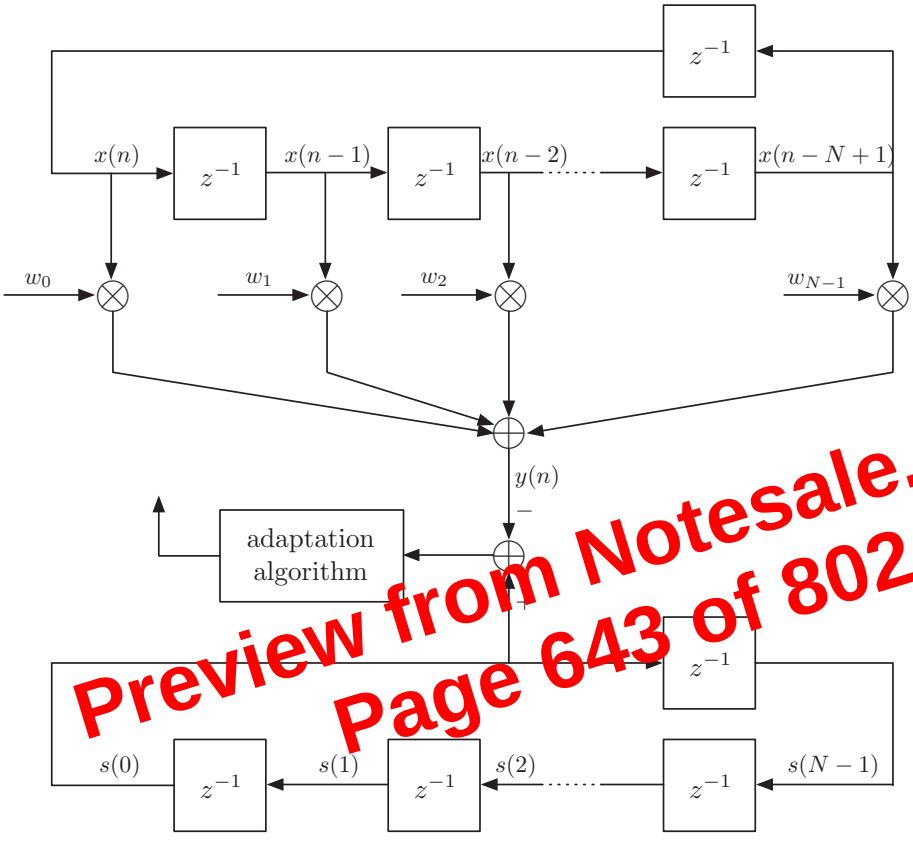

Figure 17.25 The adaptation setup for a symbol-spaced cyclic equalizer.

where "T" and "H" denote transpose and Hermitian (i.e., conjugate transpose), respectively. Also, if we define $\mathbf{x}_i$ as the circularly shifted version of $\mathbf{x}_0$ after i shifts and use the LMS algorithm for tap-weight adaptation, the repetition of the following loop will converge to a tap-weight vector that closely approximates the optimum vector $\mathbf{w}_o$. Here, the tap-weight vector $\mathbf{w}$ is initialized to an arbitrary value $\mathbf{w}(0)$. In practice, the common choice for $\mathbf{w}(0)$ is the zero vector.

for $i = 0, 1, 2, \ldots$
$\quad e(i) = s(i \mod N) - \mathbf{w}^{\mathrm{H}}(i)\mathbf{x}_i$
$\quad \mathbf{w}(i+1) = \mathbf{w}(i) + 2\mu e^*(i)\mathbf{x}_i$
end

In the above loop, "$i \mod N$" reads i modulo N, which means the integer remainder of i divided by N. It thus generates the ordered indices of the periodic sequence $s(0)$, $s(1)$, …, $s(N-1)$, $s(0)$, $s(1)$, …

the optimum tap weights. On the other hand, an indirect method that uses an estimate of the channel response and noise variance to calculate the equalizer tap weights results in much improved results.

17.6.3 Alignment of $s(n)$ and $x(n)$

So far, we have assumed that the receiver is able to identify the beginning of each cycle of the received signal that matches with the transmitted pilots. Actually, this assumption may never be true in practice. In a practical receiver, one can only obtain a coarse estimate of the boundaries of the beginning and end of the cyclic preamble.

When $s(n)$ and $x(n)$ are not time-aligned, the estimated equalizer tap weights or the channel impulse response will be replaced by their time-shifted replicas. Also, as the signals are periodic with period of N samples, these shifts will be cyclic, that is, the samples rotate within a vector of length N ($2N$ in the case of the half symbol-spaced equalizer). To remove the ambiguity caused by such rotation, the samples associated with the equalizer tap weights or the channel impulse response are rotated so that the larger tap weights/samples be positioned around the middle of the estimated vector. This method, in most cases, works pretty well, because both equalizer tap weights and the samples of the channel impulse response are sequences that grow from zero to a maximum (roughly at the middle) and decay to zero afterward.

17.6.4 Carrier and Timing Phase Acquisition and Tracking

Our discussion in this section so far was based on the assumption that the received cyclic signal was free of any frequency offset and in the case of symbol-spaced equalizer a proper timing phase was selected *a priori*. However, in practice, it is the task of the receiver to find a proper timing phase and to compensate for any residual carrier frequency offset in the demodulated received baseband signal.

Carrier Acquisition and Tracking

As in the case of the noise variance estimation, here also we assume that the cyclic preamble consists of a few cycles of the pilot sequence $s(n)$. In that case, in the presence of a carrier frequency offset $\Delta\omega_c$, the demodulated signal samples $x(n)$ are replaced by

$$x'(n) = e^{j\Delta\omega_c n} x(n) + \nu(n). \tag{17.106}$$

Note that we have ignored the rotation of the noise samples $\nu(n)$ that may be caused by the carrier frequency offset, as such rotation does not affect the statistics of the result. Recalling that during the cyclic preamble (assuming that the samples are taken at symbol rate, $1/T$) $x(n) = x(n+N)$ and assuming that the noise samples $\nu(n)$ are small enough, one may argue that

$$\sum_{n=n_0}^{K+n_0-1} x'(n+N)x'^*(n) \approx e^{j\Delta\omega_c N} \sum_{n=n_0}^{K+n_0-1} |x(n)|^2 \tag{17.107}$$

where

$$\mathbf{H} = \begin{bmatrix} h(L-1) & h(L-2) & \cdots & h(0) & 0 & \cdots & 0 \\ 0 & h(L-1) & h(L-2) & \cdots & h(0) & \cdots & 0 \\ & & & \ddots & & & \\ 0 & \cdots & 0 & h(L-1) & h(L-2) & \cdots & h(0) \end{bmatrix} \tag{17.136}$$

$\mathbf{s}(n)$ is the vector of the transmitted symbols compatible with $\mathbf{H}$, and the vector $\mathbf{v}(n)$ contains the channel noise samples. Assuming that the channel noise has a zero mean, from Eq. (17.135), one finds that

$$E[\mathbf{x}(n)] = \mathbf{H}E[\mathbf{s}(n)]. \tag{17.137}$$

We also note that the elements of $E[\mathbf{s}(n)]$ can be calculated using similar equations to Eq. (17.134).

Next, we use $y_o(n)$ to calculate a new value of the LLR according to

$$\lambda_1(s(n-\Delta)) = \ln \frac{P(s(n-\Delta)=-1)}{P(s(n-\Delta)=+1)} \tag{17.138}$$

where $P(s(n-\Delta)=+1)$ and $P(s(n-\Delta)=-1)$ are related according to the equation

$$P(s(n-\Delta)=+1) - P(s(n-\Delta)=-1) = y_o(n) \tag{17.139}$$

and also we note that

$$P(s(n-\Delta)=+1) + P(s(n-\Delta)=-1) = 1. \tag{17.140}$$

Solving Eqs. (17.139) and (17.140) for $P(s(n-\Delta)=+1)$ and $P(s(n-\Delta)=-1)$ and substituting the results in Eq. (17.138), we obtain

$$\lambda_1(s(n-\Delta)) = \ln \frac{1-y_o(n)}{1+y_o(n)}. \tag{17.141}$$

Finally,

$$\lambda_1^e(s(n-\Delta)) = \lambda_1(s(n-\Delta)) - \lambda_2^e(s(n-\Delta)) \tag{17.142}$$

is calculated and passed to the soft decoder for the next iteration.

17.9.2 Statistical Soft Equalizer

The soft MMSE equalizer obtains a linear estimate (an average) of data symbols according to Eq. (17.129) and use that to calculate the corresponding LLR values according to Eqs. (17.141) and (17.142).

The accurate estimate of the LLR value λ_1 of the data bit $b(n)$ is given by

$$\lambda_1(b(n)) = \ln \frac{P(b(n)=0|\mathbf{x},\mathbf{h},\sigma_v^2,\boldsymbol{\lambda}_2^e)}{P(b(n)=1|\mathbf{x},\mathbf{h},\sigma_v^2,\boldsymbol{\lambda}_2^e)} \tag{17.143}$$

$$p(\mathbf{x}|\mathbf{b}) = \prod_k p(x(k)|\mathbf{s}(k-L+1:k))$$
$$\propto \prod_k \exp\left(-\frac{1}{\sigma_\nu^2}\left|x(k)-\sum_{i=0}^{L-1}h(i)s(k-i)\right|^2\right)$$
$$= \exp\left(-\frac{1}{\sigma_\nu^2}\sum_k\left|x(k)-\sum_{i=0}^{L-1}h(i)s(k-i)\right|^2\right). \quad (17.149)$$

On the other hand,

$$P(\mathbf{b}|\boldsymbol{\lambda}_2^{\mathrm{e}}) = \prod_k P(b(k)|\lambda_2^{\mathrm{e}}(b(k))). \quad (17.150)$$

Moreover, as in Eq. (17.132), here,

$$P(b(k)=0|\lambda_2^{\mathrm{e}}(b(k))) = \frac{\mathrm{e}^{\lambda_2^{\mathrm{e}}(b(k))}}{1+\mathrm{e}^{\lambda_2^{\mathrm{e}}(b(k))}} = \frac{\mathrm{e}^{\lambda_2^{\mathrm{e}}(b(k))/2}}{\mathrm{e}^{-\lambda_2^{\mathrm{e}}(b(k))/2}+\mathrm{e}^{\lambda_2^{\mathrm{e}}(b(k))/2}} \quad (17.151)$$

and

$$P(b(k)=1|\lambda_2^{\mathrm{e}}(b(k))) = \frac{1}{1+\mathrm{e}^{\lambda_2^{\mathrm{e}}(b(k))}} = \frac{\mathrm{e}^{-\lambda_2^{\mathrm{e}}(b(k))/2}}{\mathrm{e}^{-\lambda_2^{\mathrm{e}}(b(k))/2}+\mathrm{e}^{\lambda_2^{\mathrm{e}}(b(k))/2}}. \quad (17.152)$$

Combining these results, we obtain

$$P(b(k)|\lambda_2^{\mathrm{e}}(b(k))) = \frac{\mathrm{e}^{(-1)^{b(k)}\lambda_2^{\mathrm{e}}(b(k))/2}}{\mathrm{e}^{-\lambda_2^{\mathrm{e}}(b(k))/2}+\mathrm{e}^{\lambda_2^{\mathrm{e}}(b(k))/2}}. \quad (17.153)$$

Next, substituting Eqs. (17.149) and (17.150) in Eq. (17.148) and using Eq. (17.153), we obtain

$$\lambda_1(b(n)) = \ln\frac{\sum_{\mathbf{b}:b(n)=0}\exp\left(-\frac{1}{\sigma_\nu^2}\sum_k\left|x(k)-\sum_{i=0}^{L-1}h(i)s(k-i)\right|^2\right)}{\sum_{\mathbf{b}:b(n)=1}\exp\left(-\frac{1}{\sigma_\nu^2}\sum_k\left|x(k)-\sum_{i=0}^{L-1}h(i)s(k-i)\right|^2\right)}$$
$$+\ln\frac{\prod_k\frac{\mathrm{e}^{(-1)^{b(k)}\lambda_2^{\mathrm{e}}(b(k))/2}}{\mathrm{e}^{-\lambda_2^{\mathrm{e}}(b(k))/2}+\mathrm{e}^{\lambda_2^{\mathrm{e}}(b(k))/2}}}{\prod_k\frac{\mathrm{e}^{(-1)^{b(k)}\lambda_2^{\mathrm{e}}(b(k))/2}}{\mathrm{e}^{-\lambda_2^{\mathrm{e}}(b(k))/2}+\mathrm{e}^{\lambda_2^{\mathrm{e}}(b(k))/2}}}$$
$$= \ln\frac{\sum_{\mathbf{b}:b(n)=0}\exp\left(-\frac{1}{\sigma_\nu^2}\sum_k\left|x(k)-\sum_{i=0}^{L-1}h(i)s(k-i)\right|^2+\sum_k(-1)^{b(k)}\frac{\lambda_2^{\mathrm{e}}(b(k))}{2}\right)}{\sum_{\mathbf{b}:b(n)=1}\exp\left(-\frac{1}{\sigma_\nu^2}\sum_k\left|x(k)-\sum_{i=0}^{L-1}h(i)s(k-i)\right|^2+\sum_k(-1)^{b(k)}\frac{\lambda_2^{\mathrm{e}}(b(k))}{2}\right)}. \quad (17.154)$$

Noting that $b(n)$ maps to $s(n)$, we have

$$\begin{aligned}\gamma_{\mathrm{a}} &= P(b(n)=a|\bar{\mathbf{b}}_n, \mathbf{x}, \lambda_2^e(b(n))) \\ &\propto p(\mathbf{x}|\mathbf{b}_n^a)P(b(n)=a) \\ &= p(\mathbf{x}|\mathbf{s}_n^a)P(b(n)=a) \\ &= \prod_{i=0}^{N_{\mathrm{b}}+L-2} p(x(i)|\mathbf{s}_m^a(i-L+1:i))P(b(n)=a) \\ &= \left\{\prod_{i=0}^{n-1} p(x(i)|\mathbf{s}_n^a(i-L+1:i)) \prod_{i=n+L}^{N_{\mathrm{b}}+L-2} p(x(i)|\mathbf{s}_n^a(i-L+1:i))\right\} \\ &\quad \times \prod_{i=n}^{n+L-1} p(x(i)|\mathbf{s}_n^a(i-L+1:i))P(b(n)=a) \end{aligned} \tag{17.155}$$

where $\mathbf{s}_n^a$ is obtained from a direct mapping of $\mathbf{b}_n^a$. Moreover, because $i \leq n-1$ and $i \geq n+L$, the vector $\mathbf{s}_n^a(i-L+1:i)$ is independent of a. Eq. (17.155) reduces to

$$\gamma_{\mathrm{a}} \propto \prod_{i=n}^{n+L-1} p(x(i)|\mathbf{s}_n^a(i-L+1:i))P(b(n)=a)$$

$$= C \cdot \exp\left\{\sum_{i=n}^{n+L-1} -\frac{1}{2\sigma_v^2}\left|x(i) - \sum_{l=0}^{L} h(l)s^a(i-l)\right|^2 + (-1)^a \frac{\lambda_2^{\mathrm{e}}(b(n))}{2}\right\} \tag{17.156}$$

where C is a scaling constant to ensure that $\gamma_0 + \gamma_1 = 1$.

Step 2: Computation of LLR Values

Assume that the Gibbs sampler has produced the important sample set $\mathcal{I}$. Each element in $\mathcal{I}$ is a bit vector of length N_{b}. As bit $b(n)$ is mapped to symbol $s(n)$, the received signal samples that are affected by $b(n)$ are $\mathbf{x}(n:n+L-1)$. As $\mathbf{x}(n:n+L-1)$ depends only on bits $\{b(l),\ n_1 = n-L+1 \leq l \leq n+L-1 = n_2\}$, we find that when computing the output LLR for $b(n)$, it is sufficient to truncate each sequence in $\mathcal{I}$ to take into account only bits $\{b(l),\ n_1 \leq l \leq n_2\}$. We denote the set that contains the truncated sequences by $\mathcal{I}_{n_1:n_2}$. For each $0 \leq n \leq N_{\mathrm{b}}-1$, we construct a larger set $\mathcal{I}^{\,n}_{n_1:n_2}$, which includes all sequences in $\mathcal{I}_{n_1:n_2}$, together with new sequences that are obtained by flipping the nth bit of each sequence in $\mathcal{I}_{n_1:n_2}$. Repetitious sequences are removed from $\mathcal{I}^{\,n}_{n_1:n_2}$. Furthermore, we let $\mathcal{I}^{\,n,0}_{n_1:n_2}$ and $\mathcal{I}^{\,n,1}_{n_1:n_2}$ denote subsequences in $\mathcal{I}^{\,n}_{n_1:n_2}$ whose nth bit equals 0 and 1, respectively. The LLR value for bit $b(n)$ is then computed as

$$\lambda_1(b(n)) = \ln \frac{\sum_{\mathbf{b}_{n_1:n_2} \in \mathcal{I}^{\,n,0}_{n_1:n_2}} p(\mathbf{x}(n:n+L-1)|\mathbf{b}_{n_1:n_2}) \prod_{l=n_1}^{n_2} P(b(l))}{\sum_{\mathbf{b}_{n_1:n_2} \in \mathcal{I}^{\,n,1}_{n_1:n_2}} p(\mathbf{x}(n:n+L-1)|\mathbf{b}_{n_1:n_2}) \prod_{l=n_1}^{n_2} P(b(l))} \tag{17.157}$$

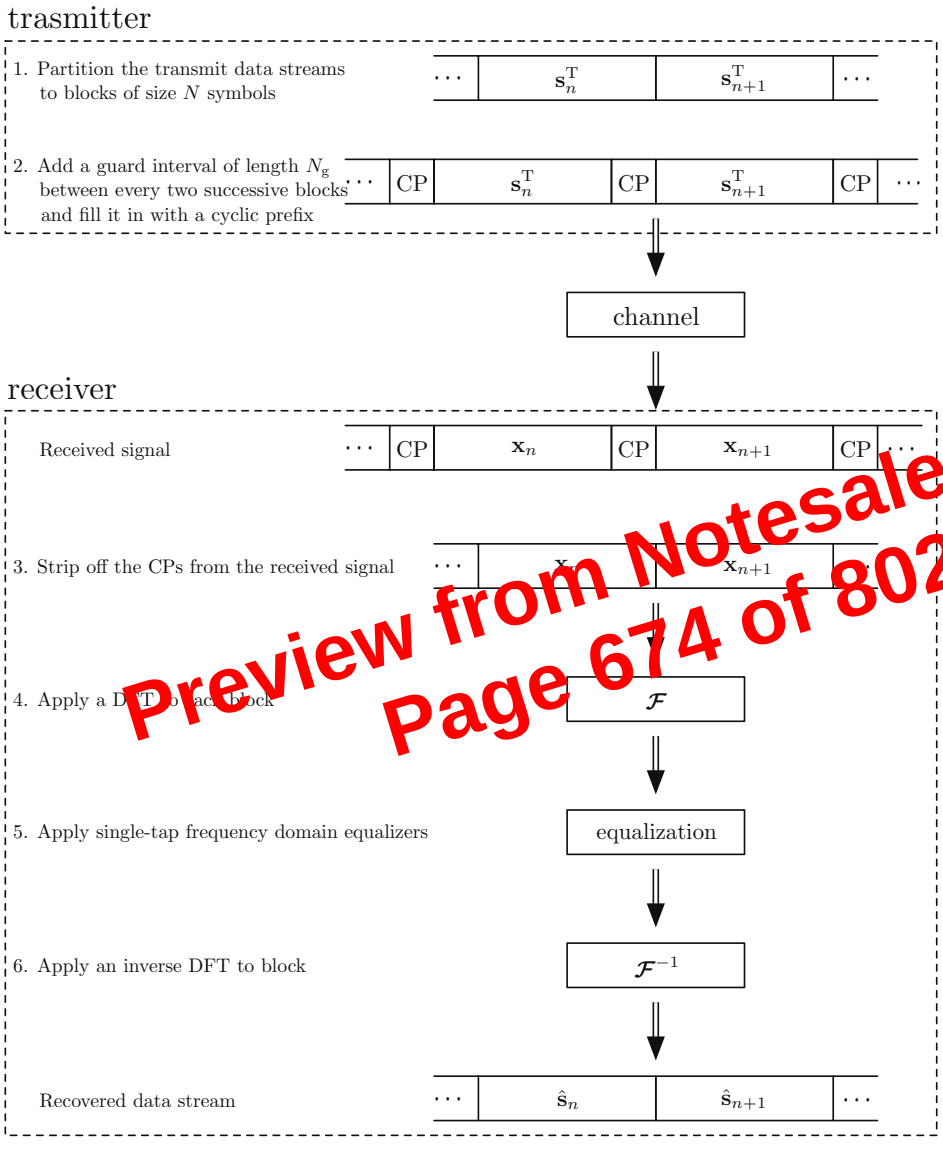

Figure 17.34 Summary of the frequency domain equalization.

to those developed in Section 17.4, the equalizer coefficients are calculated based on the estimated channel.

2. *Cyclostationary (Second-Order) Statistics-Based Methods*: These methods take note of the fact that in a digital communication system, the received signal is cyclostationary and its statistics repeat after every symbol interval. This property allows one to obtain an estimate of the impulse response of the equivalent baseband channel from the

Upon convergence of the equalizer tap weights, assuming that they have converged to the desired values, $y_o(n) \approx s(n)$, hence, one may argue that the choice of

$$\gamma = \frac{E[|s(n)|^{2l}]}{E[|s(n)|^{l}]} \tag{17.194}$$

is reasonable.

17.12.3 Blind Adaptation Algorithm

Starting with the cost function ξ and following the LMS algorithm philosophy to replace ξ by its coarse estimate $\hat{\xi} = (|y(n)|^l - r^l)^2$ in a steepest descent recursive equation, we obtain

$$\mathbf{w}(n+1) = \mathbf{w}(n) - l\mu y^*(n)|y(n)|^{l-2}(|y(n)|^l - \gamma)\mathbf{x}(n) \tag{17.195}$$

where μ is a step-size parameter. For the choice of $l = 1$, Eq. (17.195) reduces to

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \mu\frac{y^*(n)}{|y(n)|}(|y(n)| - \gamma)\mathbf{x}(n) \tag{17.196}$$

with $\gamma = E[|s(n)|^2]/E[|s(n)|]$. For the choice of $l = 2$, on the other hand, we obtain

$$\mathbf{w}(n+1) = \mathbf{w}(n) - 2\mu y^*(n)(|y(n)|^2 - \gamma)\mathbf{x}(n) \tag{17.197}$$

with $\gamma = E[|s(n)|^3]/E[|s(n)|^2]$.

One may note that the choice of $l = 2$ has a simpler form than the choice of $l = 1$. This is because the computation of $|y(n)| = \sqrt{|y(n)|^2} = \sqrt{y_R^2(n) + y_I^2(n)}$ is more involved than the computation of $|y(n)|^2 = y_R^2(n) + y_I^2(n)$ and besides, there is a divide by $|y(n)|$ in Eq. (17.196), which is absent in Eq. (17.197). Moreover, in practice $l = 2$ works better than the case of $l = 1$. The programs "`Blind_equalizer_l1.m`" and "`Blind_equalizer_l2.m`" for experimenting with the blind equalizers with choices of $l = 1$ and $l = 2$, respectively, are available on the accompanying website of this book.

Problems

P17.1 A naive pulse shape that satisfies the Nyquist condition is $p(t) = p_T(t) \star p_R(t) = \Lambda(t/T)$, where

$$\Lambda\left(\frac{t}{T}\right) = \begin{cases} 1 + t/T, & -T < t \leq 0 \\ 1 - t/T, & 0 < t < T \\ 0, & \text{otherwise.} \end{cases}$$

For this choice of $p(t)$ and an ideal channel $c(t) = \delta(t)$, consider the received signal

$$x(t) = \sum_{n=-\infty}^{\infty} s(n)p(t - nT).$$

P17.8 Repeat Problem P17.6 when the LMS-Newton algorithm is used for the adaptation of the equalizer tap weights.

P17.9 Repeat Problem P17.6 when the affine projection LMS algorithm is used for the adaptation of the equalizer tap weights.

P17.10 Present a detailed derivation of Eq. (17.111).

P17.11 Consider the system setup of Figure 17.28. Let the channel noise be absent and the impulse response of the channel be

$$\mathbf{h} = [0.1 \ -0.2 \ 0.3 \ 0.7 \ 1 \ 0.8 \ -0.5 \ -0.2 \ 0.1]^{\mathrm{T}}$$

(i) Find the equalizer vector $\mathbf{w}$ that truncates the channel response to a length of $L = 3$. Let $\Delta = 5$. To confirm your solution compare the sequences $h(n) \star w_n$ and d_n.
(ii) Develop an LMS algorithm for joint adaptation of $\mathbf{w}$ and Δ, and confirm that it converges to the solution you obtained in (i).

P17.12 Present a derivation of the solution (17.113).

P17.13 Figure 17.26 presents a structure for identifying the channel impulse response at the spacing T. Suggest a method of modifying this structure for identifying the channel impulse response at the spacing T/L, for an arbitrary integer L.

P17.14 Starting with the MATLAB script "`CyclicEq_eval.m`" on the accompanying website, extend it to confirm the results presented in Table 17.2.

P17.15 Develop a MATLAB code to obtain an estimate of the equivalent baseband impulse response of a channel by sending a single symbol $s(n) = \delta(n)$ and taking the received signal samples at the required rate, say, at the rate of $1/T$. Compare the result with what you obtain through the use of the MATLAB script "`CyclicEq_eval.m`" and confirm that both give the same channel estimate when channel noise is absent.

P17.16 In Figure 17.31, let the received signals $x_k(n)$ be related to the transmit signal $x(n)$ according to the equations

$$x_k(n) = g_k x(n) + v_k(n), \quad \text{for} \quad k = 0, 1, \ldots, M-1$$

where $v_k(n)$ is a white noise with variance σ_k^2. We wish to design an unbiased estimator for $x(n)$ using the received signal samples $x_k(n)$, for $k = 0, 1, \ldots, M-1$. To this end, one may first multiply both sides of the above equation by $1/\sigma_k$ to obtain

$$x'_k(n) = g'_k x(n) + v'_k(n), \quad \text{for} \quad k = 0, 1, \ldots, M-1$$

18.1 Narrowband Sensor Arrays

18.1.1 Array Topology and Parameters

The simplest and the most considered array geometric topology is the so-called *linear array*. In a linear array, the array elements are positioned on a straight line and often the elements are equally spaced. Figure 18.1 presents one such topology with M elements. The elements are spaced at a distance of l m. Following the presentations in Chapters 3 and 6, here too we have chosen to present the array elements as omni-directional antennas. However, we note that this is only for convenience and the results presented in this chapter are applicable to the variety of applications mentioned at the beginning of this chapter. Clearly, other array geometric topologies are also possible. Two examples are presented in Figures 18.2 and 18.3. These topologies may be found useful in cases where the space for placement of the sensor elements is limited.

In Figure 18.1, it is assumed that a plane-wave signal is impinging the sensor elements from an angle of θ. In this section, we assume that $x(n) = \alpha(n)\cos(\omega_0 n + \phi)$ is a narrowband signal centered around frequency ω_0. The fact that $x(n)$ is narrowband implies that $\alpha(n)$ varies slowly with the time index n. Moreover, in Figure 18.1, $\underline{x}_0(n)$ through $\underline{x}_{M-1}(n)$ are phasor representations of the received signal at the array element outputs. As discussed in Section 6.10, such phasor signals are recovered at each sensor element output and will be available to the sensor array processor, that is, the algorithms that are discussed in this chapter. The parameters w_0 through w_{M-1} are a set of complex-valued coefficients that we refer to as the array gains. Accordingly, we define the array *gain vector*

$$\mathbf{w} = [w_0 \;\; w_1 \;\; \cdots \;\; w_{M-1}^*]^{\mathrm{H}} \tag{18.1}$$

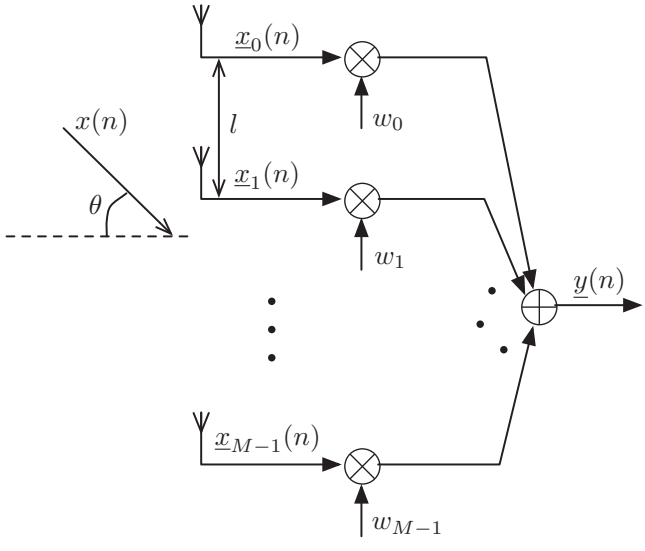

Figure 18.1 Linear geometric topology for sensor arrays.

beginning of this chapter, these idealized assumptions are usually not true. In the context of narrowband beamformers, the assumption that the steering vector of a plane-wave impinging the array is given by Eq. (18.7) will become inaccurate. The following example provides some insight into the impact of such wrong assumption.

Example 18.4

Consider a 10-element linear array that has been designed for element spacing of one half of the wavelength of impinging signals. However, owing to an implementation error, the spacing between the antennas turned out to be 5% shorter than the desired length. Study the generated optimum array pattern for the look direction of 45°, assuming the steering vector is calculated based on the assumption that the antennas spacing is that of the original design. Also, assume that in addition to the desired plane-wave signal, there are two interfering signals with powers of $P_1 = P_2 = 1$ impinging the array from angles 75° and -30°. The desired signal has the power of $P_0 = 4$. There is also a background white noise at each array element with variance $\sigma_v^2 = 0.01$.

Solution: The steering vector for the look direction, calculated based on the assumption that the element spacing is one half of the wavelength, is calculated as

$$\mathbf{s}_o(\theta_0) = \begin{bmatrix} 1 \\ e^{j\pi \sin 45^\circ} \\ \vdots \\ e^{j9\pi \sin 45^\circ} \end{bmatrix} \tag{18.60}$$

The steering vectors for the interfering signals as well as that for the desired signal, for the element spacing of 95% of one half of the wavelength, on the other, are

$$\mathbf{s}(\theta) = \begin{bmatrix} 1 \\ e^{j0.95\pi \sin\theta} \\ \vdots \\ e^{j0.95\times 9\pi \sin\theta} \end{bmatrix}, \quad \text{for} \quad \theta = 45^\circ, 75^\circ \text{ and } -30^\circ \tag{18.61}$$

Using these, we calculate the correlation matrix

$$\mathbf{R} = \sum_{k=0}^{2} P_k \mathbf{s}(\theta_k)\mathbf{s}^{\mathrm{H}}(\theta_k) + \sigma_v^2 \mathbf{I}$$

and use the result to find the optimum coefficients of the optimum beamformer according to Eq. (18.44) with $\mathbf{s}(\theta_0)$ substituted with $\mathbf{s}_o(\theta_0)$. Subsequently, the array gain is obtained using Eq. (18.10) with $\mathbf{w}$ replaced by $\mathbf{w}_o^c$ and for $\mathbf{s}(\theta)$ given by Eq. (18.61).

For the numerical values given in this example, Figure 18.20 presents the beam pattern of the implemented array. As seen, the design generates an undesirable null in the look direction $\theta_0 = 45^\circ$, meaning that the desired signal will be blocked by the array.

To explain this behavior of the array and develop further insight into the problem, we note that the steering vector $\mathbf{s}_o(\theta_0)$ given by Eq. (18.60) for the implemented array corresponds to the look direction θ_0', which is obtained by solving the equation

$$0.95\pi \sin\theta_0' = \pi \sin 45^\circ$$

The SMI method, if implemented according to Eq. (18.78), suffers from the same sensitivity problems as the original optimum beamformer. One common method of reducing this sensitivity, and thus to arrive at a robust design, is to replace $\mathbf{R}_{ss}$ by

$$\tilde{\mathbf{R}}_{ss} = \mathbf{T} \odot \mathbf{R}_{ss} \tag{18.79}$$

where $\mathbf{T}$ is a properly chosen Toeplitz matrix and $\odot$ denotes element-wise multiplication. Common choices of $\mathbf{T}$ are

$$[\mathbf{T}]_{m,n} = \mathrm{e}^{-\alpha|m-n|} \tag{18.80}$$

and

$$[\mathbf{T}]_{m,n} = \mathrm{e}^{-\alpha(m-n)^2} \tag{18.81}$$

where α is a positive parameter. $\mathbf{T}$ is called tapered matrix because of obvious reasons.

Application of tapered matrix to both $\mathbf{R}$ and $\mathbf{R}_{ss}$ have also been introduced by some researchers. This method, which is referred to as covariance matrix taper (CMT), replaces $\hat{\mathbf{R}}$ and $\mathbf{R}_{ss}$ in Eq. (18.78) by

$$\tilde{\mathbf{R}} = \mathbf{T} \odot \hat{\mathbf{R}} \quad \text{and} \quad \tilde{\mathbf{R}}_{ss} = \mathbf{T} \odot \mathbf{R}_{ss} \tag{18.82}$$

respectively. A detailed review of these methods and their extensions can be found in (Shahbazpanahi *et al.*, 2003).

Problems

P18.1 The MATLAB code used to generate the result of Figure 18.7 is available on the accompanying website. It is called `'DOABartlett.m'`.

(i) Examine this code and explain how it relates to the mathematical equations in the text.
(ii) By running `'DOABartlett.m'`, confirm the result presented in Figure 18.7.
(iii) By making necessary changes to `'DOABartlett.m'`, examine the results for the following parameters and discuss your observations.

(a) $\theta_0 = 10^\circ$, $\theta_1 = 30^\circ$, $\theta_2 = -30^\circ$, $P_0 = P_1 = P_2 = 1$, and $\sigma_v^2 = 0.01$.
(b) $\theta_0 = 10^\circ$, $\theta_1 = 30^\circ$, $\theta_2 = -30^\circ$, $P_0 = P_2 = 1$, $P_1 = 0.01$, and $\sigma_v^2 = 0.01$.

P18.2 The MATLAB code used to generate the result of Figure 18.8 is available on the accompanying website. It is called `'DOAMVDR.m'`.

(i) Examine this code and explain how it relates to the mathematical equations in the text.
(ii) By running `'DOAMVDR.m'`, confirm the result presented in Figure 18.8.
(iii) By making necessary changes to `'DOAMVDR.m'`, examine the results for the following parameters and discuss your observations.

(a) $\theta_0 = 20^\circ$, $\theta_1 = 25^\circ$, $\theta_2 = -30^\circ$, $P_0 = P_1 = P_2 = 1$, and $\sigma_v^2 = 1$.

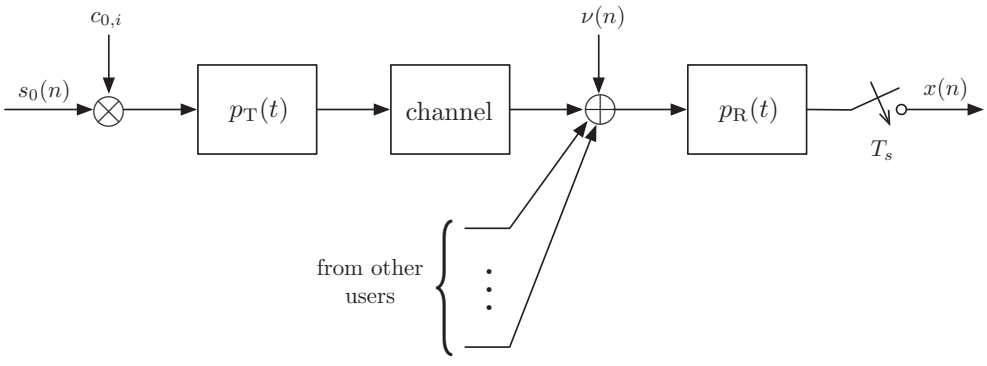

Figure 19.1 CDMA signal model.

is on various processing techniques that are applied to the signal sequence $x(n)$ for this purpose.

19.1.1 Chip-Spaced Users-Synchronous Model

Let us consider the simple case where the channel is absent and signals from all users are perfectly synchronized. Also, assume that the output of the receive filter $p_R(t)$ is perfectly sampled at the middle of each chip. Under this perfect condition, the vector of the signal samples over the interval of the nth information symbol $s_0(n)$ is obtained as

$$\mathbf{x}(n) = s_0(n)\mathbf{c}_0 + \sum_{k=1}^{K-1} \sqrt{P_k} s_k(n)\mathbf{c}_k + \mathbf{v}(n) \tag{19.1}$$

where P_k is signal power of kth user, it is assumed that $P_0 = 1$, K is the number of users,

$$\mathbf{c}_k = [c_{k,0}\ c_{k,1}\ \cdots\ c_{k,L-1}]^{\mathrm{T}} \tag{19.2}$$

$$\mathbf{x}(n) = [x_0(n)\ x_1(n)\ \cdots\ x_{L-1}(n)]^{\mathrm{T}} \tag{19.3}$$

and $\mathbf{v}(n)$ is the noise vector. We assume that the elements of $\mathbf{v}(n)$ are a set of identically independent zero-mean Gaussian random variables with variance σ_v^2. Hence, $\mathbf{v}(n)$ has the covariance matrix

$$E[\mathbf{v}(n)\mathbf{v}^{\mathrm{T}}(n)] = \sigma_v^2\mathbf{I} \tag{19.4}$$

where $\mathbf{I}$ is the identity matrix.

Under the condition that the spreading codes $\mathbf{c}_0, \mathbf{c}_1, \ldots$, and $\mathbf{c}_{K-1}$ are a set of orthogonal vectors, that is, $\mathbf{c}_k^{\mathrm{T}}\mathbf{c}_l = 0$, for $k \neq l$, an estimate of $s_0(n)$ can be trivially obtained as

$$\begin{aligned}\hat{s}_0(n) &= \frac{\mathbf{c}_0^{\mathrm{T}}\mathbf{x}(n)}{\|\mathbf{c}_0\|^2} \\ &= s_0(n) + \nu'(n)\end{aligned} \tag{19.5}$$

is orthogonal to the MAI subspace. Moreover, one may write $\mathbf{c}_0$ as $\mathbf{c}_0 = \mathbf{c}_{0,1} + \mathbf{c}_{0,2}$, where $\mathbf{c}_{0,2}$ is the projection of $\mathbf{c}_0$ into the MAI subspace.

We note that as $\mathbf{c}_{0,2}$ belongs to the MAI subspace and $\mathbf{w}$ is orthogonal to this subspace, the constraint (19.42) reduces to

$$\mathbf{w}^{\mathrm{H}}\mathbf{c}_{0,1} = 1 \tag{19.51}$$

Accordingly, one will find that, in this case, the tap-weight vector

$$\mathbf{w}_{\mathrm{o}} = \frac{\mathbf{c}_{0,1}}{\|\mathbf{c}_{0,1}\|^2} \tag{19.52}$$

satisfies the constraint (19.42) and removes MAI completely. In addition, one may realize that when $K < N$, this solution is not unique. Any addition to $\mathbf{w}_{\mathrm{o}}$ of Eq. (19.52) that is orthogonal to the spreading vectors $\mathbf{c}_0$ through $\mathbf{c}_{K-1}$ may also be seen as a solution that removes MAI completely and satisfies the constraint (19.42). The solution (19.52) would be unique in the sense that among all the solutions, it has the minimum norm. Hence, we refer to it as the *minimum norm solution*.

Next, we present an alternative method of calculating the minimum norm tap-weight vector $\mathbf{w}$ that removes MAI and satisfies the constraint (19.42). From the earlier discussion, we infer that the desired tap-weight vector belongs to the subspace spanned by the columns of the spreading code matrix $\mathbf{C}$. Hence, it may be written in the canonical form

$$\mathbf{w} = \mathbf{C}\mathbf{w}' \tag{19.53}$$

where $\mathbf{w}'$ is a length-K vector. Considering Eq. (19.53), Eq. (19.42) may be written as

$$\mathbf{w}'^{\mathrm{H}}\mathbf{c}'_0 = 1 \tag{19.54}$$

where $\mathbf{c}'_0 = \mathbf{C}^{\mathrm{H}}\mathbf{c}_0$. With these, one may use the following procedure to design an MOE detector.

Minimize the cost function

$$\xi = E[|\mathbf{w}^{\mathrm{H}}\mathbf{x}(n)|^2] = E[|\mathbf{w}'^{\mathrm{H}}\mathbf{C}^{\mathrm{H}}\mathbf{x}(n)|^2]$$

subject to the constraint (19.54).

Recalling Eq. (19.24), here, we will find that

$$\xi = \mathbf{w}'^{\mathrm{H}}\mathbf{S}\mathbf{w}' \tag{19.55}$$

where $\mathbf{S} = (\mathbf{C}^{\mathrm{H}}\mathbf{C})^2$. Note that $\mathbf{w}'$ has the length of K and $\mathbf{S}$ is a $K \times K$ matrix. Using the method of Lagrange multipliers, one will find that minimization of Eq. (19.55), subject to the constraint (19.54), has the following solution

$$\mathbf{w}'_{\mathrm{o}} = \frac{\mathbf{S}^{-1}\mathbf{c}'_0}{\mathbf{c}'^{\mathrm{H}}_0\mathbf{S}^{-1}\mathbf{c}'_0} \tag{19.56}$$

Also,

$$\xi_{\min} = \frac{1}{\mathbf{c}'^{\mathrm{H}}_0\mathbf{S}^{-1}\mathbf{c}'_0} \tag{19.57}$$

where $\mathcal{F}$ is the DFT matrix and thus $\mathcal{F}^{-1}$ is the IDFT matrix. Noting that

$$\mathcal{F}^{-1} = \frac{1}{N}\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & e^{j\frac{2\pi}{N}\times 1} & e^{j\frac{4\pi}{N}\times 1} & \cdots & e^{j\frac{2(N-1)\pi}{N}\times 1} \\ 1 & e^{j\frac{2\pi}{N}\times 2} & e^{j\frac{4\pi}{N}\times 2} & \cdots & e^{j\frac{2(N-1)\pi}{N}\times 2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & e^{j\frac{2\pi}{N}\times (N-1)} & e^{j\frac{4\pi}{N}\times (N-1)} & \cdots & e^{j\frac{2(N-1)\pi}{N}\times (N-1)} \end{bmatrix} \tag{20.2}$$

one finds that Eq. (20.1) can be expanded as

$$\mathbf{x}(n) = \sum_{k=0}^{N-1} \mathbf{x}_k(n) \tag{20.3}$$

where

$$\mathbf{x}_k(n) = \frac{1}{N} s_k(n)\mathbf{f}_k, \quad \text{for } k = 0, 1, 2, \ldots, N-1 \tag{20.4}$$

and

$$\mathbf{f}_k = \begin{bmatrix} 1 \\ e^{j\frac{2\pi k}{N}\times 1} \\ e^{j\frac{2\pi k}{N}\times 2} \\ \vdots \\ e^{j\frac{2\pi k}{N}\times (N-1)} \end{bmatrix} \tag{20.5}$$

This observation shows that the kth symbol $s_k(n)$ modulates a complex carrier at frequency $f_k = \frac{2\pi k}{N}$. The scale factor $\frac{1}{N}$ comes from the definition of the IDFT and may be removed without any major impact on the results. Also, we refer to the $\mathbf{f}_k$'s as *modulator vectors*.

For the purpose of transmission, the complex-valued sequence $x(n)$, obtained by concatenating the OFDM symbol vectors $\mathbf{x}(n)$, is modulated to an RF band. At the receiver, through a demodulation process, $x(n)$ or a distorted version of it is recovered; distortion, here, is caused by the channel and, thus, is linear. Assuming an ideal channel, an undistorted replica of $x(n)$ is recovered and, thus, the application of a DFT to each frame (i.e., each symbol vector) of $x(n)$ recovers the data symbols $s_0(n), s_1(n), \ldots, s_{N-1}(n)$. However, the sequence $x(n)$ often has a high rate, and the presence of a multipath channel, in general, results in a significant level of interference among the samples of $x(n)$, similar to ISI in single carrier communications (Chapter 17). Thus, some sort of equalization is needed to combat the channel distortion. In the case of single carrier communications, as discussed in Chapter 17, it is common to use an adaptive transversal equalizer with several taps. This has the disadvantages of high computational complexity and slow convergence. One of the key features that has made OFDM popular in broadband communication systems is that the very special structure of the OFDM symbols allows a very low complexity equalization method. Also, for reasons that are explained below, the OFDM equalizer does not suffer from any slow convergence problem.

The approach that OFDM designers have taken to combat channel effect is based on the following observation. Each subcarrier in OFDM is a constant amplitude sinusoidal tone

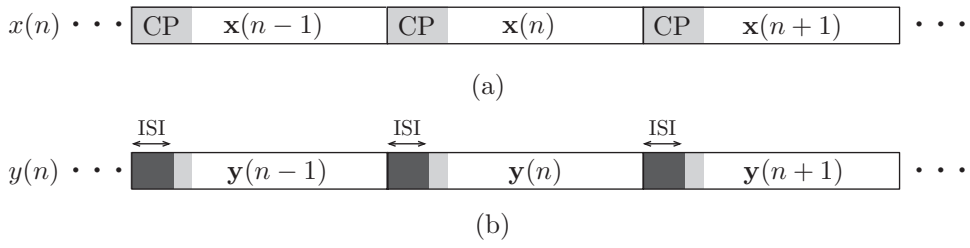

Figure 20.1 An OFDM signal with the added cyclic prefix samples: (a) transmitted signal $x(n)$ and (b) received signal $y(n)$. The presence of cyclic prefix samples avoid ISI among adjacent OFDM symbols.

that is modulated by the respective data symbol $s_k(n)$. Moreover, the frequencies of these tones are selected such that their summation at the transmitter can be generated through an IDFT and at the receiver they can be separated through a DFT. To perform the latter step, the DFT is applied to a length N sequence of the samples of the received signal. On the other hand, we note that each tone in the transmit signal undergoes a transient, which is equal to the duration of the channel impulse response, before reaching its steady state (i.e., becoming a modulated tone with the respective frequency) at the receiver. Hence, to allow separation of the data symbols at the receiver through a DFT, the tones at the transmitter should be extended from the length N to a length $N + N_{\text{cp}}$, where N_{cp} is a duration equal to or longer than the duration of the channel impulse response. Given the form of the tones (i.e., Eq. 20.5), this extension can be made by taking the last N_{cp} samples of $\mathbf{x}(n)$ and appending to its beginning. The added samples are thus called *cyclic prefix* or CP, in short. The subscript "CP" on N_{cp} is clearly selected to reflect this method of extending $\mathbf{x}(n)$.

Figure 20.1 presents an OFDM signal sequence at the transmitter and its counterpart at the receiver. At the transmitter, a CP is added to each OFDM symbol, $\mathbf{x}(n)$. The CP acts as a guard interval that absorbs the channel impulse response. That is, channel transient is absorbed within the CP, leaving the last part of each OFDM symbol a summation of the subcarrier tones with a similar form to Eq. (20.3). More specifically, if we call the last N samples of the nth OFDM symbol at the receiver $\mathbf{y}(n)$, it can be expanded as

$$\mathbf{y}(n) = \sum_{k=0}^{N-1} \mathbf{y}_k(n) \tag{20.6}$$

where

$$\mathbf{y}_k(n) = H(\omega_k) \times \frac{1}{N} s_k(n) \mathbf{f}_k, \quad \text{for } k = 0, 1, 2, \ldots, N-1 \tag{20.7}$$

$H(\omega)$ is the channel frequency response, and $\omega_k = 2\pi k/N$ is the frequency of the kth subcarrier. Note that this result follows as each tone after passing through channel and reaching its steady state is affected by a gain equal to the channel frequency response at the respective frequency.

Considering Eq. (20.6), the data symbols $s_k(n)$, for $k = 0, 1, \ldots, N-1$, can be recovered from $\mathbf{y}(n)$ by taking the DFT of $\mathbf{y}(n)$ and applying a set of single-tap

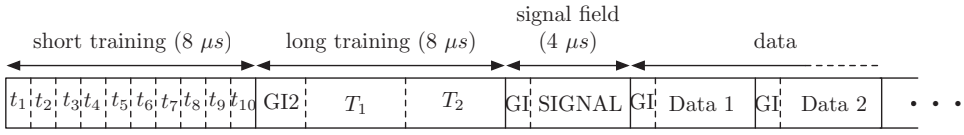

Figure 20.2 Packet structure of IEEE 802.11a.

Table 20.1 The symbol values for short training symbols in IEEE 802.11a.

k	$X_{s,k}$
−24	$\sqrt{\frac{13}{6}}(1+j)$
−20	$-\sqrt{\frac{13}{6}}(1+j)$
−16	$\sqrt{\frac{13}{6}}(1+j)$
−12	$-\sqrt{\frac{13}{6}}(1+j)$
−8	$\sqrt{\frac{13}{6}}(1+j)$
−4	$\sqrt{\frac{13}{6}}(1+j)$
4	$-\sqrt{\frac{13}{6}}(1+j)$
8	$-\sqrt{\frac{13}{6}}(1+j)$
12	$\sqrt{\frac{13}{6}}(1+j)$
16	$\sqrt{\frac{13}{6}}(1+j)$
20	$\sqrt{\frac{13}{6}}(1+j)$
24	$\sqrt{\frac{13}{6}}(1+j)$

values, $X_s(k)$, corresponding to these tones are listed in Table 20.1. The short symbols are thus generated according to the equation

$$x_{\text{short}}(t) = g_{\text{short}}(t) \sum_{k \in \mathcal{S}} X_s(k) e^{j2\pi k \Delta_c t} \tag{20.8}$$

where

$$g_{\text{short}}(t) = \begin{cases} 1, & 0 \le t \le 8\mu s \\ 0, & \text{otherwise} \end{cases} \tag{20.9}$$

and Δ_c is the spacing between the adjacent subcarriers. In 802.11a, if all 64 subcarriers were used, the total bandwidth would be 20 MHz. Hence, the subcarrier spacing is $\frac{20}{64} = 0.3125$ MHz. Substituting this value in Eq. (20.8), one finds that $x_{\text{short}}(t)$ over the interval $0 \le t \le 8\mu s$ is periodic and repeats 10 times. It is also worth noting that the factor $\sqrt{\frac{13}{6}}$ in

To proceed, we note that the phase offset factor $e^{j\phi_0}$ is a constant that can be absorbed in the single-tap equalizers tap weights. Hence, we remove the factor from the right-hand side of Eq. (20.31) for the rest of our discussions. With this modification, Eq. (20.31) reduces to

$$\mathbf{y}(n) = \mathbf{\Phi}(\Delta\omega_c)\sum_{k=0}^{N-1}\frac{H(\omega_k)}{N}s_k(n)\mathbf{f}_k + \mathbf{v}(n) \tag{20.33}$$

We note that $\mathbf{\Phi}(\Delta\omega_c)$ is an undesired factor that has been introduced as a result of the frequency offset. Moreover, premultiplying $\mathbf{y}(n)$ by $\mathbf{\Phi}^*(\Delta\omega_c) = \mathbf{\Phi}(-\Delta\omega_c)$ will result in the carrier offset free vector

$$\begin{aligned}\tilde{\mathbf{y}}(n) &= \mathbf{\Phi}(-\Delta\omega_c)\mathbf{y}(n)\\ &= \sum_{k=0}^{N-1}\frac{H(\omega_k)}{N}s_k(n)\mathbf{f}_k + \mathbf{\Phi}(-\Delta\omega_c)\mathbf{v}(n)\end{aligned} \tag{20.34}$$

To develop an adaptive algorithm for correcting the carrier frequency offset, we define the vector

$$\check{\mathbf{y}}(n) = \mathbf{\Phi}(\varphi)\mathbf{y}(n) \tag{20.35}$$

and search for the value of φ that minimizes the cost function

$$J(\varphi) = \sum_{k\in\mathcal{P}}\left[\left|\mathbf{f}_l^H\check{\mathbf{y}}(n) - H(\omega_k)s_l(n)\right|^2\right] \tag{20.36}$$

where $\mathcal{P}$ is the set of subcarrier numbers of the pilot symbols. An LMS-type algorithm can now be implemented by replacing $J(\varphi)$ with its coarse estimate

$$\hat{J}(\varphi) = \sum_{k\in\mathcal{P}}\left|\mathbf{f}_l^H\check{\mathbf{y}}(n) - H(\omega_k)s_l(n)\right|^2 \tag{20.37}$$

and running the gradient update equation

$$\varphi(n+1) = \varphi(n) - \mu\frac{\partial\hat{J}(\varphi)}{\partial\varphi} \tag{20.38}$$

Replacing Eq. (20.37) in Eq. (20.38) and noting that

$$\frac{\partial\mathbf{\Phi}(\varphi)}{\partial\varphi} = j\mathbf{\Lambda}\mathbf{\Phi}(\varphi) \tag{20.39}$$

where

$$\mathbf{\Lambda} = \begin{bmatrix}0&0&0&\cdots&0\\0&1&0&\cdots&0\\0&0&2&\cdots&0\\\vdots&\vdots&\vdots&\ddots&\vdots\\0&0&0&\cdots&(N-1)\end{bmatrix}$$

we obtain

$$\varphi(n+1) = \varphi(n) - 2\mu\sum_{k\in\mathcal{P}}\Re\left\{j\mathbf{f}_l^H\mathbf{\Lambda}\check{\mathbf{y}}(n)\left(\mathbf{f}_l^H\check{\mathbf{y}}(n) - H(\omega_k)s_l(n)\right)^*\right\} \tag{20.40}$$

The update equation (20.40) can be improved further by adding the unused subcarriers to the list of pilots. Here, the pilot symbols are equal to zero. As mentioned above,

the unused subcarriers are called virtual subcarriers and, accordingly, the related carrier recovery algorithms are referred to as virtual subcarriers-based carrier recovery methods. Note that even if there are no pilot symbols, the virtual/null subcarriers alone may be used for carrier tracking. In that case, Eq. (20.40) converts to

$$\varphi(n+1) = \varphi(n) - 2\mu \sum_{k \in \mathcal{V}} \Re\left\{ j \mathbf{f}_l^{\mathrm{H}} \mathbf{\Lambda} \check{\mathbf{y}}(n) \left(\mathbf{f}_l^{\mathrm{H}} \check{\mathbf{y}}(n) \right)^* \right\} \tag{20.41}$$

where $\mathcal{V}$ is the set of subcarrier numbers of the virtual subcarriers. Clearly, the update equations (20.40) and (20.41) may also be combined.

We note that an implementation of Eq. (20.40) requires the knowledge of the channel coefficients $H(\omega_k)$ that should be replaced by their estimates. On the other hand, if we limit ourselves to the virtual subcarriers, where Eq. (20.41) is applicable, the channel knowledge is not required. Yet, there are other methods that can be used to track the carrier frequency without even using the knowledge of the position of pilot or virtual subcarriers. These methods are referred to as *blind carrier-tracking algorithms*.

Blind Carrier-Tracking Algorithms

There are several blind carrier-tracking algorithms in the literature. In this section, we present one of them. The presented algorithm takes note of the fact that the addition of cyclic prefix to OFDM symbols adds a regular variation (ripple) to its spectrum that may be deployed to develop a simple carrier-tracking algorithm. The blind carrier-tracking method that is presented here is due to Talbot (2003), also see Talbot and Farhang-Boroujeny (2008).

OFDM spectrum

As discussed earlier, an OFDM signal is constructed by concatenating a set of cyclic prefixed symbols. Let us use $\mathbf{x}^{\mathrm{cpd}}(n)$ to denote the nth cyclic prefixed symbol of the OFDM signal $x(n)$. Following similar presentations to those of Eq. (20.3) through Eq. (20.5), here, we have

$$\mathbf{x}^{\mathrm{cpd}}(n) = \sum_{k=0}^{N-1} \mathbf{x}_k^{\mathrm{cpd}}(n) \tag{20.42}$$

where

$$\mathbf{x}_k^{\mathrm{cpd}}(n) = \frac{1}{N} s_k(n) \mathbf{f}_k^{\mathrm{cpd}}, \quad \text{for } k = 0, 1, 2, \ldots, N-1 \tag{20.43}$$

and

$$\mathbf{f}_k^{\mathrm{cpd}} = \begin{bmatrix} \mathrm{e}^{j\frac{2\pi k}{N} \times (N - N_{\mathrm{cp}})} \\ \vdots \\ \mathrm{e}^{j\frac{2\pi k}{N} \times (N-1)} \\ 1 \\ \mathrm{e}^{j\frac{2\pi k}{N} \times 1} \\ \mathrm{e}^{j\frac{2\pi k}{N} \times 2} \\ \vdots \\ \mathrm{e}^{j\frac{2\pi k}{N} \times (N-1)} \end{bmatrix} \tag{20.44}$$

We are interested in evaluating the power spectral density of $x(n)$. To this end, we first assume that $s_k(n)$'s are independent across the frequency axis (the index k), and note that this implies

$$\Phi_{xx}(\omega) = \sum_{k \in \mathcal{K}} \Phi_{x_k x_k}(\omega) \tag{20.45}$$

where $\mathcal{K}$ is the set of subcarrier indices of the active (data and pilot) subcarriers, $\Phi_{xx}(\omega)$, following the notations in Chapter 2, denotes the PSD of $x(n)$, and, similarly, $\Phi_{x_k x_k}(\omega)$ denotes the PSD of the sequence $x_k(n)$ obtained by concatenating the $\mathbf{x}_k^{\text{cpd}}(n)$'s.

Next, to evaluate $\Phi_{x_k x_k}(\omega)$, we note that

$$x_k(n) = \frac{1}{N} \sum_{l=-\infty}^{\infty} s_k(l) g(n - l(N + N_{\text{cp}})) \mathrm{e}^{j2\pi(n - l(N+N_{\text{cp}}) - N_{\text{cp}})k/N} \tag{20.46}$$

where $g(n)$ is referred to as *symbol shaping window function*. It is a rectangular window that confines the duration of each OFDM symbol to $N + N_{\text{cp}}$ samples. More specifically,

$$g(n) = \begin{cases} 1, & -N_{\text{cp}} \le n \le N - 1 \\ 0, & \text{otherwise} \end{cases} \tag{20.47}$$

Equation (20.46) implies that $x_k(n)$ is obtained by passing the symbol sequence $s_k(n)$ through the system shown in Figure 20.4.

Following Figure 20.4, the PSD of $x_k(n)$ is obtained as

$$\Phi_{x_k x_k}(\omega) = \frac{\sigma_s^2}{(N + N_{\text{cp}})N^2} \left| G(\mathrm{e}^{j(\omega - \frac{2\pi k}{N})}) \right|^2 \tag{20.48}$$

where σ_s^2 is the variance of $s_k(n)$ and $G(\omega)$ is the Fourier transform of $g(n)$. In Eq. (20.48), the frequency shift of $\frac{k}{N}$ in $G(\mathrm{e}^{j(\omega - \frac{2\pi k}{N})})$ is a consequence of the modulating factor $\mathrm{e}^{j\frac{2\pi k}{N}(n - N_{\text{cp}})}$ behind $g(n)$ in Figure 20.4, and the factor $1/N^2$ arises because of the factor $1/N$ on the front of $g(n)$. The additional factor $1/(N + N_{\text{cp}})$ is included because there is only one $s_k(n)$ for every $N + N_{\text{cp}}$ samples of $x_k(n)$, hence, its energy averages out over $N + N_{\text{cp}}$ samples.

Substituting Eq. (20.48) in Eq. (20.45), we obtain

$$\Phi_{xx}(\omega) = \frac{\sigma_s^2}{(N + N_{\text{cp}})N^2} \sum_{k \in \mathcal{K}} \left| G(\mathrm{e}^{j(\omega - \frac{2\pi k}{N})}) \right|^2 \tag{20.49}$$

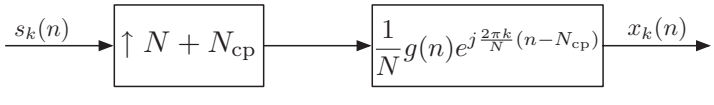

Figure 20.4 The process of generating $x_k(n)$ from $s_k(n)$.

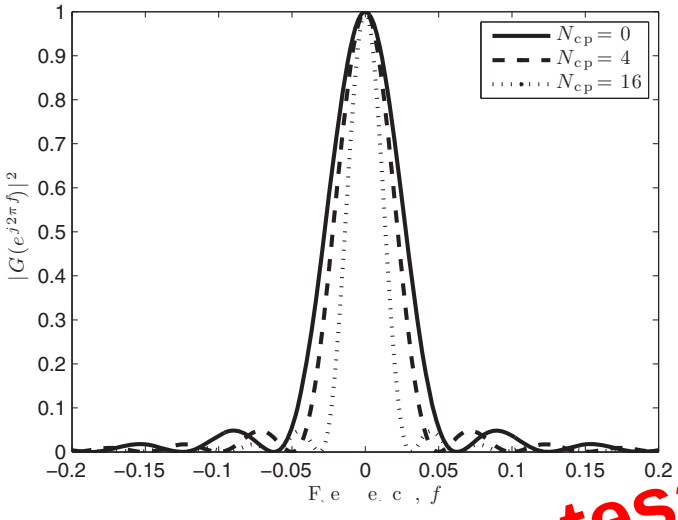

Figure 20.5 The impact of the cyclic prefix length on the variation of the width of main lobe of $G(\omega)$.

Also, it is straightforward to show that for the rectangular window (20.47) (see Problem P20.5, at the end of this chapter)

$$|G(\omega)|^2 = \sum_{n=-(N+N_{cp}-1)}^{N+N_{cp}-1} (N + N_{cp} - |n|)\cos(n\omega) \tag{20.50}$$

Figure 20.5 presents a set of plots of $|G(\omega)|^2$, normalized to the maximum amplitude of unity. The parameter N is set equal to 16 and N_{cp} (the length of CP) is given the values of 0, 4, and 16. We note that for a fixed N, as N_{cp} increases, the width of the main lobe of $|G(\omega)|^2$ decreases. On the other hand, the subcarrier spacing remains constant at $1/N$. Hence, Eq. (20.49) implies that, for a fixed N, increasing the length of CP induces ripples with larger amplitudes in the in-band region of OFDM PSD. This is shown in Figure 20.6 for the case where $N = 16$, there are 10 active subcarriers at frequencies $f = \pm\frac{1}{N}, \pm\frac{2}{N}, \ldots, \pm\frac{5}{N}$, and N_{cp} is given the values of 0, 4, and 16. The magnitude of different spectra are also normalized to the maximum of unity to allow a quantitative comparison.

Impact of Carrier Offset

The spectral effect of a carrier offset is, obviously, a shift of the signal spectrum. For example, if the OFDM signal corresponding to the PSD for the case of $N_{cp} = 4$ in Figure 20.6 experiences a carrier offset of $\Delta f_c = \Delta\omega_c/2\pi = 0.375/N$, then the normalized OFDM PSD will be the one shown in Figure 20.7. As the spectral shift is directly related to the size of carrier offset, Δf_c, a measurement of the spectral shift is a measurement of the carrier offset.

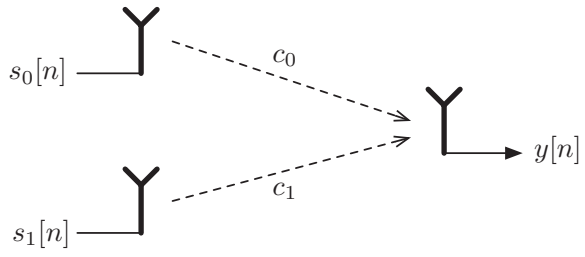

Figure 20.14 A communication channel with two transmit antennas and one receive antennas.

Each row of this matrix indicates the pair of symbols that are transmitted from the transmit antennas. The same pair of symbols are transmitted in the next time slot after conjugation and possibly a sign change. Note that in this arrangement two symbols are transmitted in two time slots and this, clearly, is equivalent of transmitting one symbol per time slot. We also note that the code matrix $\mathbf{S}$ satisfies

$$\mathbf{S}^{\mathrm{H}}\mathbf{S} = \left(|s_0|^2 + |s_1|^2\right)\mathbf{I} \tag{20.83}$$

where $\mathbf{I}$ is the identity matrix, hence the name OSTB code.

Assuming that the pair of data symbols $s(n)$ and $s(n+1)$ are transmitted according to the Alamouti code, the pair of the received signal samples received at the time slots n and $n+1$ are given by

$$\begin{bmatrix} y(n) \\ y(n+1) \end{bmatrix} = \begin{bmatrix} s(n) & s(n+1) \\ -s^*(n+1) & s^*(n) \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \end{bmatrix} + \begin{bmatrix} v(n) \\ v(n+1) \end{bmatrix} \tag{20.84}$$

To extract estimates of $s(n)$ and $s(n+1)$ from Eq. (20.84), we first note that Eq. (20.84) can be rearranged as

$$\begin{bmatrix} y(n) \\ y^*(n+1) \end{bmatrix} = \begin{bmatrix} c_0 & c_1 \\ -c_1^* & c_0^* \end{bmatrix} \begin{bmatrix} s(n) \\ s(n+1) \end{bmatrix} + \begin{bmatrix} v(n) \\ v^*(n+1) \end{bmatrix} \tag{20.85}$$

Defining

$$\mathbf{C} = \begin{bmatrix} c_0 & c_1 \\ -c_1^* & c_0^* \end{bmatrix} \tag{20.86}$$

and multiplying Eq. (20.84) through from left-hand side by $\dfrac{1}{|c_0|^2 + |c_1|^2}\mathbf{C}^{\mathrm{H}}$, we obtain

$$\begin{aligned} \begin{bmatrix} \tilde{s}(n) \\ \tilde{s}(n+1) \end{bmatrix} &= \frac{1}{|c_0|^2 + |c_1|^2}\mathbf{C}^{\mathrm{H}} \begin{bmatrix} y(n) \\ y^*(n+1) \end{bmatrix} \\ &= \begin{bmatrix} s(n) \\ s(n+1) \end{bmatrix} + \begin{bmatrix} \dfrac{c_0^* v(n) - c_1 v^*(n+1)}{|c_0|^2 + |c_1|^2} \\ \dfrac{c_1^* v(n) + c_0 v^*(n+1)}{|c_0|^2 + |c_1|^2} \end{bmatrix} \end{aligned} \tag{20.87}$$

Obviously, the performance of the ZF detector is determined by the variance of the elements of $\mathbf{v}'(n)$ that is quantified by the diagonal elements of its covariance matrix. Using Eqs. (20.91) and (20.68), the covariance matrix of $\mathbf{v}'(n)$ is obtained as

$$E[\mathbf{v}'(n)\mathbf{v}'^{\mathrm{H}}(n)] = \sigma_v^2\left(\mathbf{C}^{\mathrm{H}}\mathbf{C}\right)^{-1} \tag{20.92}$$

This shows that the size of elements of $E[\mathbf{v}'(n)\mathbf{v}'^{\mathrm{H}}(n)]$ are related to the size of elements of the matrix $\left(\mathbf{C}^{\mathrm{H}}\mathbf{C}\right)^{-1}$ which, in turn, are related to the condition number, respectively, the spread of eigenvalues of the matrix $\mathbf{C}^{\mathrm{H}}\mathbf{C}$. When one or more of eigenvalues of $\mathbf{C}^{\mathrm{H}}\mathbf{C}$ are small, the size of elements of $\left(\mathbf{C}^{\mathrm{H}}\mathbf{C}\right)^{-1}$ will be large and, hence, ZF detector performs poorly. More specifically, one may quantify the performance of the ZF detector by considering the SNR values at the detector outputs. These SNR values are defined as

$$\mathrm{SNR}_k = \frac{\sigma_s^2}{\sigma_{v_k'}^2} \tag{20.93}$$

where $\sigma_s^2 = E[|s_k(n)|^2]$ and $\sigma_{v_k'}^2 = E[|v_k'(n)|^2]$. Note that while we assume σ_s^2 is the same for the data symbols transmitted from different antennas, $\sigma_{v_k'}^2$ differs for various choices of k and also it varies with the channel gain $\mathbf{C}$.

Numerical Example: Part 1

Some of the properties of the ZF detector could be best understood through numerical examples. Let us consider the following two choices of $\mathbf{C}$.

$$\mathbf{C}_1 = \begin{bmatrix} 0.4209 + 0.0060i & -0.3859 + 0.1032i & 0.2284 - 0.0397i \\ -0.0327 + 0.1392i & 0.2426 - 0.0631i & 0.4007 - 0.1766i \\ -0.0280 + 0.1797i & -0.0037 - 0.4032i & -0.3683 - 0.0566i \end{bmatrix}$$

and

$$\mathbf{C}_2 = \begin{bmatrix} 0.0382 - 0.0424i & -0.0099 - 0.2285i & 0.1134 + 0.4595i \\ 0.4010 + 0.0661i & -0.0344 + 0.4154i & 0.3406 + 0.1319i \\ 0.2441 - 0.1635i & -0.1571 + 0.3103i & 0.0105 + 0.2018i \end{bmatrix}$$

The first choice is a poorly conditioned matrix; the eigenvalues of $\mathbf{C}_1^{\mathrm{H}}\mathbf{C}_1$ are 0.0081, 0.8843, and 1.1777. The second choice is a moderately conditioned matrix; the eigenvalues of $\mathbf{C}_2^{\mathrm{H}}\mathbf{C}_2$ are 0.0637, 0.2553, and 0.6810.

Letting $\sigma_v^2 = 0.01$, the SNR values at the detector outputs (calculated according to Eq. (20.93) for the two cases) are obtained as

$$\text{For } \mathbf{C}_1: \ \mathrm{SNR} = -5.46, -4.12, 1.18\mathrm{dB}$$

$$\text{For } \mathbf{C}_2: \ \mathrm{SNR} = 9.92, 11.37, 14.42\mathrm{dB}$$

Clearly, as expected, the poorly conditioned channel gain $\mathbf{C}_1$ leads to very low SNR values at the detector outputs.

20.3 MIMO–OFDM

Extension of OFDM to MIMO channel is straightforward. Figure 20.15 present a block diagram of a MIMO–OFDM system. For brevity, modulators/demodulators to/from RF are not included. Separate OFDM symbol/signal sequences are *synchronously* transmitted from different transmit antennas. At the receiver, a set of OFDM demodulators separate the received signals into their respective subcarriers. This includes the removal of CP from each OFDM symbol and applying an FFT. Recalling the discussion on OFDM in Section 20.1, one may note that OFDM partitions the channel into a set of flat-fading MIMO channels expressed by the set of equations

$$\mathbf{y}_k(n) = \mathbf{C}_k\mathbf{s}_k(n) + \mathbf{v}_k(n), \quad k \in \mathcal{K} \tag{20.112}$$

where $\mathcal{K}$ indicates the indices of the active subcarriers.

All the methods related to MIMO system that were discussed in Section 20.2 are obviously applicable to the set of channel models (20.112) as well. In particular, the space-diversity and/or space-multiplexing methods can be applied to each subcarrier channel separately. Moreover, the channel estimation methods that were discussed in Section 20.2.4 can be readily extended to each subcarrier channel. In addition, the pilot spreading and channel estimation methods that were developed for OFDM systems can extended to MIMO–OFDM. A few relevant literature that interested readers may refer to are Ghosh *et al*. (2011), Chiueh, Tsai, and Lai (2012), and Stüber *et al*. (2004).

Problems

P20.1 Study and present your observation on the effect of cyclic prefix extension of $\mathbf{x}(n)$ on the individual tones embedded in $\mathbf{x}(n)$. Specifically, confirm that the cyclic prefix extension does not introduced and discontinuity in the time index of the tones.

P20.2

(i) Starting with the MATLAB script "`OFDM.m`," available on the accompanying website, add the necessary line to form the autocorrelation coefficients $r_{yy}(n)$ and plot the result to confirm that you obtain a plot similar to the one in Figure 20.3.

(ii) Provide the necessary arguments to prove that the time indices n_1, n_2, n_3, and n_4 of Figure 20.3 are related accordingly as

$$n_1 + 192 = n_2 + 160 = n_3 + 32 = n_4$$

(iii) Check to see these relationships are applicable to the plot you obtained in (i).

(iv) Do the observations made above change in presence of a carrier frequency offset? Examine the developed code and also provide theoretical explanation.

P20.3 In the MATLAB script "`OFDM.m`," available on the accompanying website, let the carrier frequency offset be zero (Dfc = 0) and the noise variance be set equal to 10^{-8} (sigmav = 0.0001). Also, to begin assume that the channel is ideal ($c = 1$).

Clark, A.P. and Hau, S.F. (1984) Adaptive adjustment of receiver for distorted digital signals. *IEE Proceedings*, **131** part F, 526–536.

Clark, A.P. and Hariharan, S. (1989) Adaptive channel estimator for an HF radio link. *IEEE Transactions on Communications*, **COM-37** (9), 918–926.

Crochiere, R.E. and Rabiner, L.R. (1983) *Multirate Digital Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ.

Cupo, L.R. and Gitlin, R.D. (1989) Adaptive carrier recovery systems for digital data communications receivers. *IEEE Journal on Selected Areas in Communications*, **7**, 1328–1339.

David, R.D., Stearns, S.D., Elliott, G.R., and Etter, D.M. (1983) IIR algorithm for adaptive line enhancement. *Proceedings ICASSP'83*, pp. 17–20.

Davila, C.E. (1990) A stochastic Newton algorithm with data-adaptive step size. *IEEE Transactions on Acoustics, Speech, & Signal Processing*, **ASSP-38**, 1796–1798.

De Courville, M. and Duhamel, P. (1995) Adaptive filtering in subbands using a weighted criterion. *Proceedings IEEE ICASSP'95, May, Detroit, MI*, pp. 985–988.

de León, P.L. II and Etter, D.M. (1995) Experimental results with increased bandwidth analysis filters in oversampled, subband acoustic echo cancellers. *IEEE Signal Processing Letters*, **2** (1), 1–3.

Dembo, A. and Salz, J. (1990) On the least squares tap adjustment algorithm in adaptive digital echo cancellers. *IEEE Transactions on Communications*, **COM-38**, 622–628.

Demoment, G. and Reynaud, R. (1985) Fast minimum variance deconvolution. *IEEE Transactions on Acoustics, Speech, & Signal Processing*, **ASSP-33**, 1324–1326.

Douglas, S.C. (1997) Performance comparison of two implementations of the leaky LMS adaptive filter. *IEEE Transactions on Signal Processing*, **45** (8), 2125–2129.

Duttweiler, D.L. (1978) A twelve-channel digital echo canceler. *IEEE Transactions on Communications*, **COM-26** (5), 647–653.

Duttweiler, D.L. (1982) Adaptive filter performance with nonlinearities in the correlation multiplier. *IEEE Transactions on Acoustics Speech and Signal Processing*, **ASSP-30**, 578–586.

Egelmeers, G.P.M. and Sommen, P.C.W. (1996) A new method for efficient convolution in frequency domain by nonuniform partitioning for adaptive filtering. *IEEE Transactions on Signal Processing*, **44** (12), 3123–3129.

Eleftheriou, E. and Falconer, D.D. (1986) Tracking properties and steady-state performance of RLS adaptive filter algorithms. *IEEE Transactions on Acoustics, Speech, & Signal Processing*, **ASSP-34**, 1097–1109.

Eleftheriou, E. and Falconer, D.D. (1987) Adaptive equalization techniques for HF channels. *IEEE Journal on Selected Areas in Communications*, **SAC-5**, 238–247.

Elliott, S.J. and Nelson, P.A. (1993) Active noise control. *IEEE Signal Processing Letters*, **10** (4), 12–35.

Elliott, D.F. and Rao, K.R. (1982) *Fast Transforms: Algorithms, Analysis, Applications*, Academic Press, New York.

Elliott, S.J. and Rafaely, B. (1997) Rapid frequency-domain adaptation of causal FIR filters. *IEEE Signal Processing Letters*, **4** (12), 337–339.

Eneman, K. and Moonen, M. (1997) A Relation between subband and frequency-domain adaptive filtering. *IEEE International Conference on Digital Signal Processing*, **1**, 25–28.

Er, M. and Cantoni, A. (1983) Derivative constraints for broad-band element space antenna array processors. *IEEE Transactions on Acoustics, Speech, & Signal Processing*, **31** (6), 1378–1393.

Ersoy, O.K. (1997) *Fourier-Related Transforms, Fast Algorithms and Applications*, Prentice Hall PTR, Upper Saddle River, NJ, p. 07458.

Esterman, P. and Kaelin, A. (1994) Analysis of the transient behavior of unconstrained frequency domain adaptive filters. *IEEE International Symposium on Circuits and Systems*, **2**, 21–24.

Eweda, E. (1990a) Analysis and design of a signed regressor LMS algorithm for stationary and nonstationary adaptive filtering with correlated Gaussian data. *IEEE Transactions on Circuits and Systems*, **CAS-37**, 1367–1374.

Eweda, E. (1990b) Optimum step size of sign algorithm for nonstationary adaptive filtering. *IEEE Transactions on Acoustics, Speech, & Signal Processing*, **ASSP-38**, 1897–1901.

Eweda, E. (1994) Comparison of RLS, LMS, and sign algorithms for tracking randomly time-varying channels. *IEEE Transactions on Signal Processing*, **42** (11), 2937–2944.

Slock, D.T.M. and kailath, T. (1992) A modular multichannel multiexperiment fast transversal filter RLS algorithm. *Signal Processing*, **28**, 25–45.

Slock, D.T.M., Chisci, L., Lev-Ari, H., and Kailath, T. (1992) Modular and numerically stable fast transversal filters for multichannel and multiexperiment RLS. *IEEE Transactions on Signal Processing*, **ASSP-40** (4), 784–802.

Slock, D.T.M. and Kailath, T. (1993) in *Adaptive System Identification and Signal Processing Algorithms* (eds N. Kalouptsidis and S. Theodoridis), Prentice-Hall, UK, Chapter 5.

Schmidl, T.M. and Cox, D.C. (1997) Robust frequency and timing synchronization for OFDM. *IEEE Transactions on Communications*, **45** (12), 1613–1621.

Solo, V. (1992) The error variance of LMS with time-varying weights. *IEEE Transactions on Signal Processing*, **40** (4), 803–813.

Somayazulu, V.S., Mitra, S.K., and Shynk, J.J. (1989) Adaptive line enhancement using multirate techniques. *Proceedings of IEEE ICASSP'89, Conference, May, Glasgow, Scotland*, pp. 928–931.

Sommen, P. (1988) On the convergence properties of a partitioned block frequency domain adaptive filter (PBFDAF). *Proceedings EUSIPCO*, pp. 1401–1404.

Sommen, P.C.W. (1989) Partitioned frequency domain adaptive filters. *Proceedings Asilomar Conference on Signals and Systems, Pacific Grove, CA*, pp. 676–681.

Sommen, P.C.W., Van Gerwen, P.J., Kotmans, H., and Janssen, A.J.E.M. (1987) Convergence analysis of a frequency domain adaptive filter with exponential power averaging and generalized window function. *IEEE Transactions on Circuits and Systems*, **34** (7), 788–798.

Sommen, P.C.W. and de Wilde, E. (1992) Equal convergence conditions for normal and partitioned frequency domain adaptive filters *Proceedings ICASSP'92, Conference*, **Vol. IV**, pp. 69–72.

Sondhi, M.M. and Kellermann, W. (1992) Adaptive echo cancellation for speech signals in *Advances in Speech Signal Processing* (eds S. Furui and M.M. Sondhi), Marcel Dekker, New York, pp. 327–356, Chapter 11.

Sondhi, M.M., Morgan, D.R. and Hall, J.L. (1995). Stereophonic acoustic echo cancellation –an overview of the fundamental problem. *IEEE Signal Processing Letters*, **2** (8), 148–151.

Soo, J.S. and Pang, K.K. (1987) A new structure for block FIR adaptive digital filters. *IREECON Int. Dig. Papers, Sydney, Australia*, pp. 364–367.

Soo, J.S. and Pang, K.K. (1990) Multidelay block frequency domain adaptive filter. *IEEE Transactions on Acoustics, Speech, & Signal Processing*, **ASSP-38**, 373–376.

Soumekh, M. (1994) *Fourier Array Imaging*, Prentice-Hall, Englewood Cliffs, NJ.

South, C.R., Hoppitt, C.E., and Lewis, A.V. (1979) Adaptive filters to improve loudspeaker telephone. *Electronics Letters*, **15** (21), 673–674.

Stasinski, R. (1990) Adaptive Filters in Domains of Adaptive Transforms. *Proceedings of Singapore ICCS'90, Conference, 5–9 Nov Singapore*, pp. 18.2.1–18.2.5.

Stearns, S.D. (1981) Error surface of recursive adaptive filters. *IEEE Transactions on Circuits and Systems*, **CAS-28** (6), 603–606.

Stein, S. (1987) Fading channels issues in system engineering. *IEEE Journal on Selected Areas in Communications*, **SAC-5**, 68–89.

Stoica, P., Wang, Z., and Li, J. (2003) Robust Capon beamforming. *IEEE Signal Processing Letters*, **10** (6), 172–175.

Strang, G. (1980) *Linear Algebra and Its Applications*., 2nd edn, Academic Press, New York.

Stüber, G.L., Barry, J.R., McLaughlin, S.W., Li, Ye., Ingram, M.A., and Pratt, T.G. (2004) Broadband MIMO-OFDM wireless communications. *Proceedings of the IEEE*, **92** (2), 271–294.

Talbot, S.L. (2008) Carrier synchronization for mobile orthogonal frequency division multiplexing. PhD Thesis, Univ. of Utah.

Talbot, S.L., and Farhang-Boroujeny, B., (2008) Spectral method of blind carrier tracking for OFDM, *IEEE Transactions on Signal Processing*, **56** (7), Part 1, 2706–2717.

Tanrikulu, O., Baykal, B., Constantinides, A.G., and Chambers, J.A. (1997). Residual echo signal in critically sampled subband acoustic echo cancellers based on IIR and FIR filter banks *IEEE Transactions on Signal Processing*, **45** (4), 901–912.

Tarrab, M. and Feuer, A. (1988) Convergence and performance analysis of the normalized LMS algorithm with uncorrelated Gaussian data. *IEEE Transactions on Information Theory*, **IT-34**, 680–691.

Tong, L., Xu, G., and Kailath, T. (1994) Blind identification and equalization based on second-order statistics: a time domain approach. *IEEE Transactions on Information Theory*, **40** (2), 340–349.

Tourneret, J.-Y., Bershad, N.J., and Bermudez, J.C.M. (2009) Echo cancellation – the generalized likelihood ratio test for double-talk versus channel change. *IEEE Transactions on Signal Processing*, **57** (3), 916–926.

Tsatsanis, M.K. and Giannakis, G.B. (1997) Transmitter induced cyclostationarity for blind channel equalization. *IEEE Transactions on Signal Processing*, **45** (7), 1785–1794.

Tüchler, M., Singer, A.C., and Koetter, R. (2002) Minimum mean squared error equalization using a priori information. *IEEE Transactions on Signal Processing*, **50** (3), 673–683.

Tugnait, J.K. (1995) Blind equalization and estimation of digital communication FIR channels using cumulant matching. *IEEE Transactions on Communications*, **43** (2-4), 1240–1245.

Tummala, M. and Parker, S.R. (1987) A new efficient adaptive cascade lattice structure. *IEEE Transactions on Circuits and Systems*, **34** (7), 707–711.

Vaidyanathan, P.P. (1993) *Multirate Systems and Filter Banks*, Prentice-Hall, Englewood Cliffs, NJ.

Vaidyanathan, P.P. and Nguyen, T.Q. (1987) Eigenfilters: a new approach to least-squares FIR filter design and applications including Nyquist filters. *IEEE Transactions on Circuits and Systems*, **CAS-34**, 11–23.

Valin, J.-M. (2007) On adjusting the learning rate in frequency domain echo cancellation with double-talk. *IEEE Transactions Audio, and Language Processing*, **15** (3), 1030–1034.

van den Bos, A. (1994) Complex gradient and Hessian. *IEE Proceedings - Vision, Image, and Signal Processing*, **141** (1), 380–382.

van Waterschoot, T. and Moonen, M. (2011) Fifty years of acoustic feedback control: State of the art and future challenges. *Proceedings of the IEEE*, **99** (2), 288–327.

Verhaegen, M.H. (1989) Improved understanding of the loss-of-symmetry phenomenon in the conventional Kalman filter. *IEEE Transactions on Automatic Control*, **AC-34**, 331–333.

Verhaegen, M.H. (1989) Round-off error propagation in four generally applicable, recursive least-squares estimation schemes. *Automatica*, **5** (3), 437–444.

Verhoeckx, N.A.M. and Claasen, T.A.C.M. (1984) Some considerations on the design of adaptive digital filters equipped with the sign algorithm. *IEEE Transactions Acoustic, Speech & Signal Processing*, **ASSP-32**, 258–266.

Vikalo, H., Hassibi, B., and Kailath, T. (2004) Iterative decoding for MIMO channels via modified sphere decoding. *IEEE Transactions on Wireless Communications*, **3** (6), 2299–2311.

von Zitzewitz, A. (1990), Considerations on acoustic echo cancelling based on realtime experiments *Proceedings IEEE EUSIPCO'90 V European Signal Process. Conference, Sept, Barcelona, Spain*, pp. 1987–1990.

Wackersreither, G. (1985) On the design of filters for ideal QMF and polyphase filter banks. *AEU – International Journal of Electronics and Communications, Elsevier*, 123–130.

Wang, X. and Poor, H.V. (1999) Iterative (turbo) soft interference cancellation and decoding for coded CDMA. *IEEE Transactions on Communications*, **47** (7), 1046–1061.

Wang, Z. (1996) Adaptive filtering in subband. M.Eng. thesis. Department of Electrical Engrg, National Univ. of Singapore.

Ward, C.R., Hargrave, P.J., and McWhirter, J.G. (1986) A novel algorithm and architecture for adaptive digital beamforming. *IEEE Transactions on Antennas and Propagation*, **AP-34**, 338–346.

Wei, P.C., Zeidler, J.R., and Ku, W.H. (1997) Adaptive recovery of a chirped signal using the RLS algorithm. *Ieee Transactions on Signal Processing*, **45** (2), 363–376.

Weiss, A. and Mitra, D. (1979) Digital adaptive filters: Conditions for convergence, rates of convergence, effects of noise and errors arising from the implementation. *IEEE Transactions on Information Theory*, **IT-25**, 637–652.

Widrow, B. and Hoff, M.E. Jr. (1960) Adaptive switching circuits. *IRE WESCON Conv. Rec.*, pt. 4 96–104.

Widrow, B., Mantey, P.E., Griffiths, L.J., and Goode, B.B. (1967). Adaptive antenna systems *Proceedings of the IEEE*, **55** (12), 2143–2159.

Widrow, B., McCool, J., and Ball, M. (1975) The complex LMS algorithm. *Proceedings of the IEEE*, **63**, 719–720.

Widrow, B., Glover, J.R., Jr., McCool, J.M., Kaunitz, J., Williams, C.S., Hearn, R.H., Zeidler, J.R., Dong, E. Jr., and Goodlin, R.C. (1975). Adaptive noise cancelling: principles and applications. *Proceedings of the IEEE*, **63** (12), 1692–1716.