CHAPTER 9

Least-Mean-Square Algorithm

In this chapter we develop the theory of a widely used algorithm named the *least-mean-square (LMS) algorithm* by its originators, Widrow and Hoff (1960). The LMS algorithm is an important member of the family of *stochastic gradient algorithms*. The term "stochastic gradient" is intended to distinguish the LMS algorithm from the method of steepest descent that uses a deterministic gradient in a recursive computation of the Wiener filter for stochastic inputs. A significant feature of the LMS algorithm is its *simplicity*. Moreover, it does not require measurements of the pertinent correlation functions, nor does it require matrix inversion. Indeed, it is the simplicity of the LMS algorithm that has made it the *standard* against which other adaptive filtering algorithms are benchmarked.

The material presented in this chapter also includes a detailed analysis of the convergence behavior of the LMS algorithm, supported by computer experiments. We begin our study of the LMS algorithm by presenting an overview of its structure and operation.

9.1 OVERVIEW OF THE STRUCTURE AND OPERATION OF THE LEAST-MEAN-SQUARE ALGORITHM

The least-mean-square (LMS) algorithm is a linear adaptive filtering algorithm that consists of two basic processes:

1. A *filtering process*, which involves (a) computing the output of a transversal filter produced by a set of tap inputs, and (b) generating an estimation error by comparing this output to a desired response.

2. An *adaptive process*, which involves the automatic adjustment of the tap weights of the filter in accordance with the estimation error.

Thus, the combination of these two processes working together constitutes a *feedback loop* around the LMS algorithm, as illustrated in the block diagram of Fig. 9.1(a). First, we have a transversal filter, around which the LMS algorithm is built; this component is responsible for performing the filtering process. Second, we have a mechanism for performing the adaptive control process on the tap weights of the transversal filter, hence the designation "adaptive weight-control mechanism" in Fig. 9.1(a).

Details of the *transversal filter* component are presented in Fig. 9.1(b). The tap inputs $u(n), u(n-1), \ldots, u(n-M+1)$ form the elements of the M-by-1 *tap-input vector* $\mathbf{u}(n)$, where $M-1$ is the number of delay elements; these tap inputs span a multidimensional space denoted by $\mathcal{U}_n$. Correspondingly, the tap weights $\hat{w}_0(n), \hat{w}_1(n), \ldots, \hat{w}_{M-1}(n)$ form the elements of the M-by-1 tap-weight vector $\hat{\mathbf{w}}(n)$. The value computed for the tap-weight vector $\hat{\mathbf{w}}(n)$ using the LMS algorithm represents an estimate whose expected value approaches the Wiener solution $\mathbf{w}_o$ (for a wide-sense stationary environment) as the number of iterations n approaches infinity.

During the filtering process the *desired response* $d(n)$ is supplied for processing, alongside the tap-input vector $\mathbf{u}(n)$. Given this input, the transversal filter produces an output $\hat{d}(n|\mathcal{U}_n)$ used as an *estimate* of the desired response $d(n)$. Accordingly, we may define an *estimation error* $e(n)$ as the difference between the desired response and the actual filter output, as indicated in the output end of Fig. 9.1(b). The estimation error $e(n)$ and the tap-input vector $\mathbf{u}(n)$ are applied to the control mechanism, and the feedback loop around the tap weights is thereby closed.

Figure 9.1(c) presents details of the *adaptive weight-control mechanism*. Specifically, a scalar version of the inner product of the estimation error $e(n)$ and the tap input $u(n-k)$ is computed for $k = 0, 1, 2, \ldots, M-2, M-1$. The result so obtained defines the *correction* $\delta\hat{w}_k(n)$ applied to the tap weight $\hat{w}_k(n)$ at iteration $n+1$. The scaling factor used in this computation is denoted by μ in Fig. 9.1(c). It is called the *step-size parameter*.

Comparing the control mechanism of Fig. 9.1(c) for the LMS algorithm with that of Fig. 8.2 for the method of steepest descent, we see that the LMS algorithm uses the product $u(n-k)e^*(k)$ as an estimate of element k in the gradient vector $\nabla J(n)$ that characterizes the method of steepest descent. In other words, the expectation operator is missed out from all the paths in Fig. 9.1(c). Accordingly, the recursive computation of each tap weight in the LMS algorithm suffers from a *gradient noise*.

Throughout this chapter, it is assumed that the tap-input vector $\mathbf{u}(n)$ and the desired response $d(n)$ are drawn from a jointly wide-sense stationary environment. For such an environment, we know from Chapter 8 that the method of steepest descent computes a tap-weight vector $\mathbf{w}(n)$ that moves down the ensemble-averaged error-performance surface along a deterministic trajectory, which terminates on the Wiener solution, $\mathbf{w}_o$. The LMS algorithm, on the other hand, behaves differently because of the presence of gradient noise.

Rather than terminating on the Wiener solution, the tap-weight vector $\hat{\mathbf{w}}(n)$ [different from $\mathbf{w}(n)$] computed by the LMS algorithm executes a *random motion* around the minimum point of the error-performance surface.

Earlier we pointed out that the LMS algorithm involves feedback in its operation, which therefore raises the related issue of *stability*. In this context, a meaningful criterion is to require that

$$J(n) \rightarrow J(\infty) \qquad \text{as } n \rightarrow \infty$$

where $J(n)$ is the mean-squared error produced by the LMS algorithm at time n, and its final value $J(\infty)$ is a constant. An algorithm that satisfies this requirement is said to be *convergent in the mean square*. For the LMS algorithm to satisfy this criterion, the step-size parameter μ has to satisfy a certain condition related to the eigenstructure of the correlation matrix of the tap inputs.

The difference between the final value $J(\infty)$ and the minimum value $J_{\min}$ attained by the Wiener solution is called the *excess mean-squared error* $J_{ex}(\infty)$. This difference represents the price paid for using the adaptive (stochastic) mechanism to control the tap weights in the LMS algorithm in place of a deterministic approach as in the method of steepest descent. The ratio of $J_{ex}(\infty)$ to $J_{\min}$ is called the *misadjustment*, which is a measure of how far the steady-state solution computed by the LMS algorithm is away from the Wiener solution. It is important to realize, however, that the misadjustment $\mathcal{M}$ is under the designer's control. In particular, the feedback loop acting around the tap weights behaves like a *low-pass filter*, whose "average" time constant is *inversely* proportional to the step-size parameter μ. Hence, by assigning a small value to μ the adaptive process is made to progress slowly, and the effects of gradient noise on the tap weights are largely filtered out. This, in turn, has the effect of reducing the misadjustment.

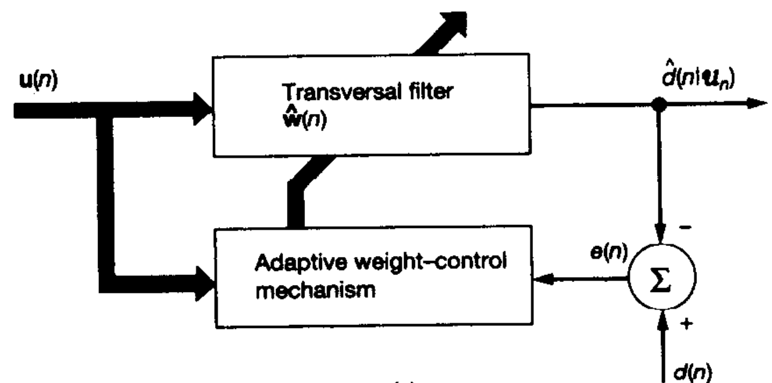
We may therefore justifiably say that the LMS algorithm is simple in implementation, yet capable of delivering high performance by adapting to its external environment. To do so, however, we have to pay particular attention to the choice of a suitable value for the step-size parameter μ.

A closely related issue is that of specifying a suitable value for the filter order $M-1$; for this specification, we may use the AIC or MDL criterion for the model-order selection that was discussed in Chapter 2.

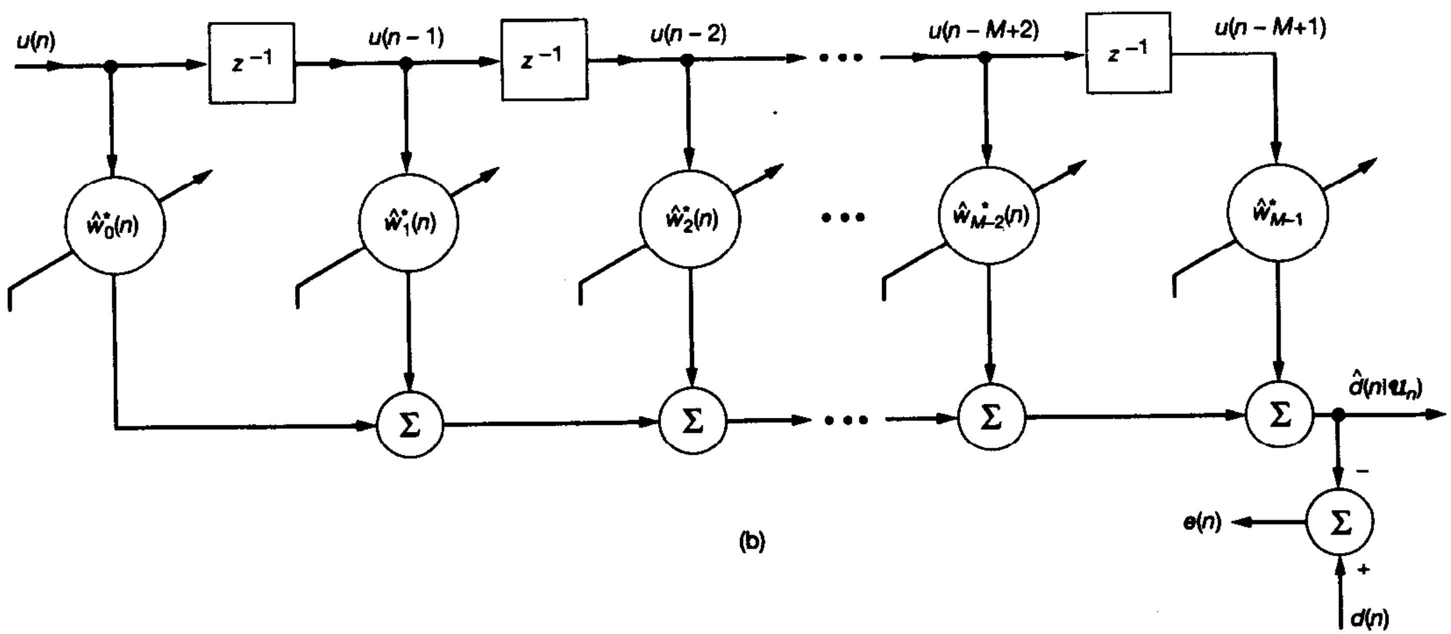
The factors influencing the choice of μ are covered in Sections 9.4 and 9.7. First and foremost, however, we wish to derive the LMS algorithm. This we do in the next section by building on our previous knowledge of the method of steepest descent.

9.2 LEAST-MEAN-SQUARE ADAPTATION ALGORITHM

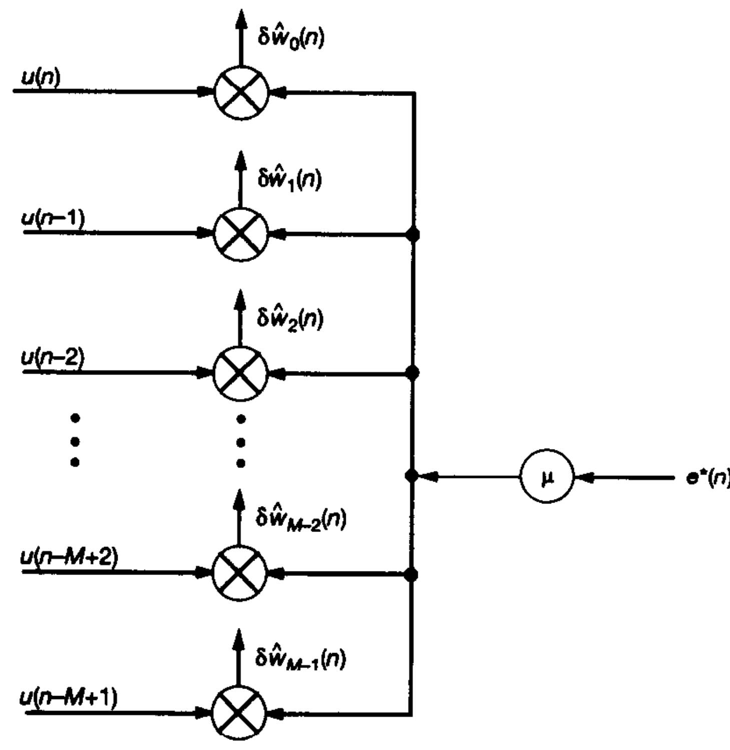
If it were possible to make exact measurements of the gradient vector $\nabla J(n)$ at each iteration n, and if the step-size parameter μ is suitably chosen, then the tap-weight vector computed by using the steepest-descent algorithm would indeed converge to the optimum


u(n)
Transversal filter
ŵ(n)
Adaptive weight-control
mechanism
d̂(n|𝒰n)
e(n)
Σ
−
+
d(n)

(a)


u(n)
z⁻¹
u(n − 1)
z⁻¹
u(n − 2)
u(n − M+2)
z⁻¹
u(n − M+1)
ŵ*0(n)
ŵ*1(n)
ŵ*2(n)
ŵ*M−2(n)
ŵ*M−1
Σ
Σ
Σ
Σ
d̂(n|𝒰n)
−
e(n)
Σ
+
d(n)

(b)



(c)

Figure 9.1 (a) Block diagram of adaptive transversal filter. (b) Detailed structure of the transversal filter component. (c) Detailed structure of the adaptive weight-control mechanism.

Wiener solution. In reality, however, exact measurements of the gradient vector are not possible since this would require prior knowledge of both the correlation matrix $\mathbf{R}$ of the tap inputs and the cross-correlation vector $\mathbf{p}$ between the tap inputs and the desired response [see Eq. (8.8)]. Consequently, the gradient vector must be *estimated* from the available data

To develop an estimate of the gradient vector $\nabla J(n)$, the most obvious strategy is to *substitute* estimates of the correlation matrix $\mathbf{R}$ and the cross-correlation vector $\mathbf{p}$ in the formula of Eq. (8.8), which is reproduced here for convenience:

$$\nabla J(n) = -2\mathbf{p} + 2\mathbf{R}\mathbf{w}(n) \tag{9.1}$$

The *simplest* choice of estimators for $\mathbf{R}$ and $\mathbf{p}$ is to use *instantaneous estimates* that are based on sample values of the tap-input vector and desired response, as defined by, respectively,

$$\hat{\mathbf{R}}(n) = \mathbf{u}(n)\mathbf{u}^H(n) \tag{9.2}$$

and

$$\hat{\mathbf{p}}(n) = \mathbf{u}(n)d^*(n) \tag{9.3}$$

Correspondingly, the instantaneous estimate of the gradient vector is

$$\hat{\nabla} J(n) = -2\mathbf{u}(n)d^*(n) + 2\mathbf{u}(n)\mathbf{u}^H(n)\hat{\mathbf{w}}(n) \tag{9.4}$$

Generally speaking, this estimate is *biased* because the tap-weight estimate vector $\hat{\mathbf{w}}(n)$ is a random vector that depends on the tap-input vector $\mathbf{u}(n)$. Note that the estimate $\hat{\nabla} J(n)$ may also be viewed as the gradient operator ∇ applied to the instantaneous squared error $|e(n)|^2$; see Problem 2 of Chapter 8.

Substituting the estimate of Eq. (9.4) for the gradient vector $\nabla J(n)$ in the steepest-descent algorithm described in Eq. (8.7), we get a new recursive relation for updating the tap-weight vector:

$$\hat{\mathbf{w}}(n + 1) = \hat{\mathbf{w}}(n) + \mu\mathbf{u}(n)[d^*(n) - \mathbf{u}^H(n)\hat{\mathbf{w}}(n)] \tag{9.5}$$

Here we have used a hat over the symbol for the tap-weight vector to distinguish it from the value obtained by using the steepest-descent algorithm. Equivalently, we may write the result in the form of three basic relations as follows:

1. *Filter output:*

$$y(n) = \hat{\mathbf{w}}^H(n)\mathbf{u}(n) \tag{9.6}$$

2. *Estimation error:*

$$e(n) = d(n) - y(n) \tag{9.7}$$

3. *Tap-weight adaptation:*

$$\hat{\mathbf{w}}(n + 1) = \hat{\mathbf{w}}(n) + \mu\mathbf{u}(n)e^*(n) \tag{9.8}$$

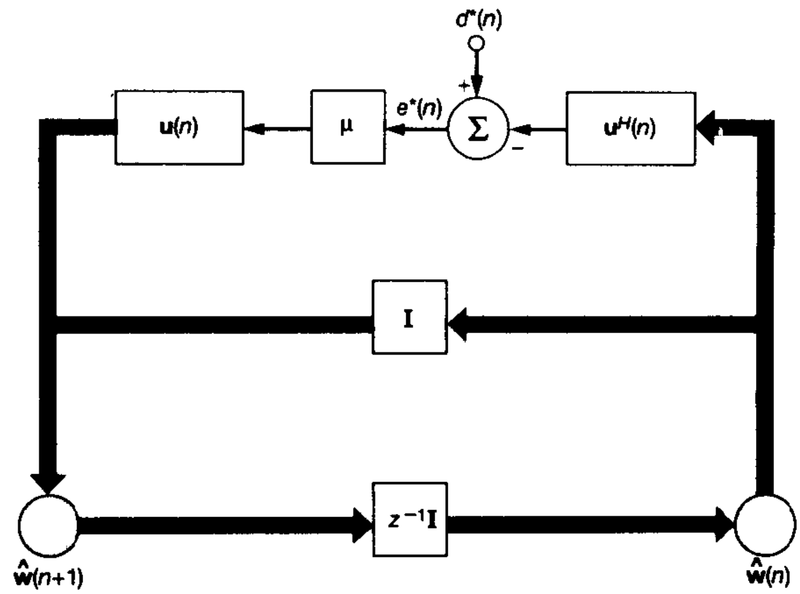


Figure 9.2 Signal-flow graph representation of the LMS algorithm.

Equations (9.6) and (9.7) define the estimation error $e(n)$, the computation of which is based on the *current estimate* of the tap-weight vector, $\hat{\mathbf{w}}(n)$. Note also that the second term, $\mu\mathbf{u}(n)e^*(n)$, on the right-hand side of Eq. (9.8) represents the *correction* that is applied to the current estimate of the tap-weight vector, $\hat{\mathbf{w}}(n)$. The iterative procedure is started with an initial guess $\hat{\mathbf{w}}(0)$.

The algorithm described by Eqs. (9.6) to (9.8) is the complex form of the adaptive *least-mean-square (LMS) algorithm.*[1] At each iteration or time update, it also requires knowledge of the most recent values: $\mathbf{u}(n)$, $d(n)$, and $\hat{\mathbf{w}}(n)$. The LMS algorithm is a member of the family of *stochastic gradient algorithms.* In particular, when the LMS algorithm operates on stochastic inputs, the allowed set of directions along which we "step" from one iteration cycle to the next is quite random and cannot therefore be thought of as being true gradient directions.

Figure 9.2 shows a signal-flow graph representation of the LMS algorithm in the form of a feedback model. This model bears a close resemblance to the feedback model of

[1]The complex form of the LMS algorithm, as originally proposed by Widrow et al. (1975b), differs slightly from that described in Eqs. (9.6) to (9.8). Widrow and other authors have based their derivation on the definition $\mathbf{R} = E[\mathbf{u}^*(n)\mathbf{u}^T(n)]$ for the correlation matrix. On the other hand, the LMS algorithm described by Eqs. (9.6) to (9.8) is based on the definition $\mathbf{R} = E[\mathbf{u}(n)\mathbf{u}^H(n)]$ for the correlation matrix. The adoption of the latter definition for the correlation matrix of complex-valued data is the natural extension of the definition for real-valued data.

Fig. 8.3 describing the steepest-descent algorithm. The signal-flow graph of Fig. 9.2 clearly illustrates the simplicity of the LMS algorithm. In particular, we see from this figure that the LMS algorithm requires only $2M + 1$ *complex multiplications and 2M complex additions per iteration,* where M is the number of tap weights used in the adaptive transversal filter. In other words, the computational complexity of the LMS algorithm is $O(M)$.

The instantaneous estimates of $\mathbf{R}$ and $\mathbf{p}$ given in Eqs. (9.2) and (9.3), respectively, have relatively large variances. At first sight, it may therefore seem that the LMS algorithm is incapable of good performance since the algorithm uses these instantaneous estimates. However, we must remember that the LMS algorithm is recursive in nature, with the result that the algorithm itself effectively averages these estimates, in some sense, during the course of adaptation.

9.3 EXAMPLES

Before proceeding further with a convergence analysis of the LMS algorithm, it is instructive to develop an appreciation for the versatility of this important algorithm. We do this by presenting six examples that relate to widely different applications of the LMS algorithm.

Example 1: Canonical Model of the Complex LMS Algorithm

The LMS algorithm described in Eqs. (9.6) to (9.8) is *complex* in the sense that the input and output data as well as the tap weights are all complex valued. To emphasize the complex nature of the algorithm, we use complex notation to express the data and tap weights as follows:

Tap-input vector:

$$\mathbf{u}(n) = \mathbf{u}_I(n) + j\mathbf{u}_Q(n) \tag{9.9}$$

Desired response:

$$d(n) = d_I(n) + jd_Q(n) \tag{9.10}$$

Tap-weight vector:

$$\hat{\mathbf{w}}(n) = \hat{\mathbf{w}}_I(n) + j\hat{\mathbf{w}}_Q(n) \tag{9.11}$$

Transversal filter output:

$$y(n) = y_I(n) + jy_Q(n) \tag{9.12}$$

Estimation error:

$$e(n) = e_I + je_Q(n) \tag{9.13}$$

The subscripts I and Q denote "in-phase" and "quadrature" components, respectively; that is, the real and imaginary parts, respectively. Using these definitions in Eqs. (9.6) to (9.8), expanding terms, and then equating real and imaginary parts, we get

$$y_I(n) = \hat{\mathbf{w}}_I^T(n)\mathbf{u}_I(n) - \hat{\mathbf{w}}_Q^T(n)\mathbf{u}_Q(n) \tag{9.14}$$

$$y_Q(n) = \hat{\mathbf{w}}_I^T(n)\mathbf{u}_Q(n) + \hat{\mathbf{w}}_Q^T(n)\mathbf{u}_I(n) \tag{9.15}$$

$$e_I(n) = d_I(n) - y_I(n) \tag{9.16}$$

$$e_Q(n) = d_Q(n) - y_Q(n) \tag{9.17}$$

$$\hat{\mathbf{w}}_I(n+1) = \hat{\mathbf{w}}_I(n) + \mu[e_I(n)\mathbf{u}_I(n) - e_Q(n)\mathbf{u}_Q(n)] \tag{9.18}$$

$$\hat{\mathbf{w}}_Q(n+1) = \hat{\mathbf{w}}_Q(n) + \mu[e_I(n)\mathbf{u}_Q(n) + e_Q(n)\mathbf{u}_I(n)] \tag{9.19}$$

Equations (9.14) to (9.17), defining the error and output signals, are represented by the cross-coupled signal-flow graph shown in Fig. 9.3(a). The update equations (9.18) and (9.19) are likewise represented by the cross-coupled signal-flow graph shown in Fig. 9.3(b). The combination of this pair of signal-flow graphs constitutes the *canonical model* of the complex LMS algorithm. This canonical model clearly illustrates that a complex LMS algorithm is equivalent to a set of four real LMS algorithms with *cross-coupling* between them. Its use may arise, for example, in the adaptive equalization of a communication channel for the transmission of binary data by means of a multiphase modulation scheme such as quadriphase-shift keying (QPSK).

Example 2: Adaptive Deconvolution for Processing of Time-Varying Seismic Data

In Section 7 of the introductory chapter, we described the idea of *predictive deconvolution* as a method for processing reflection seismograms. In this example we discuss the use of the LMS algorithm for the adaptive implementation of predictive deconvolution. It is assumed that the seismic data are stationary over the design gate used to generate the deconvolution operation.

Let the set of *real* data points in the seismogram to be processed be denoted by $u(n)$, $n = 1, 2, \ldots, N$, where N is the data length. The real LMS-based adaptive deconvolution method, which is appropriate for this example, proceeds as follows (Griffiths et al., 1977):

- An M-dimensional operator $\hat{\mathbf{w}}(n)$ is used to generate a predicted trace from the data; that is,

$$\hat{\mathbf{u}}(n+\Delta) = \hat{\mathbf{w}}^T(n)\mathbf{u}(n) \tag{9.20}$$

where

$$\hat{\mathbf{w}}(n) = [\hat{w}_0(n), \hat{w}_1(n), \ldots, \hat{w}_{M-1}(n)]^T$$

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

and $\Delta \geq 1$ is the *prediction depth*.

- The deconvolved trace $y(n)$ defining the difference between the input and predicted traces is evaluated:

$$y(n) = u(n) - \hat{u}(n)$$

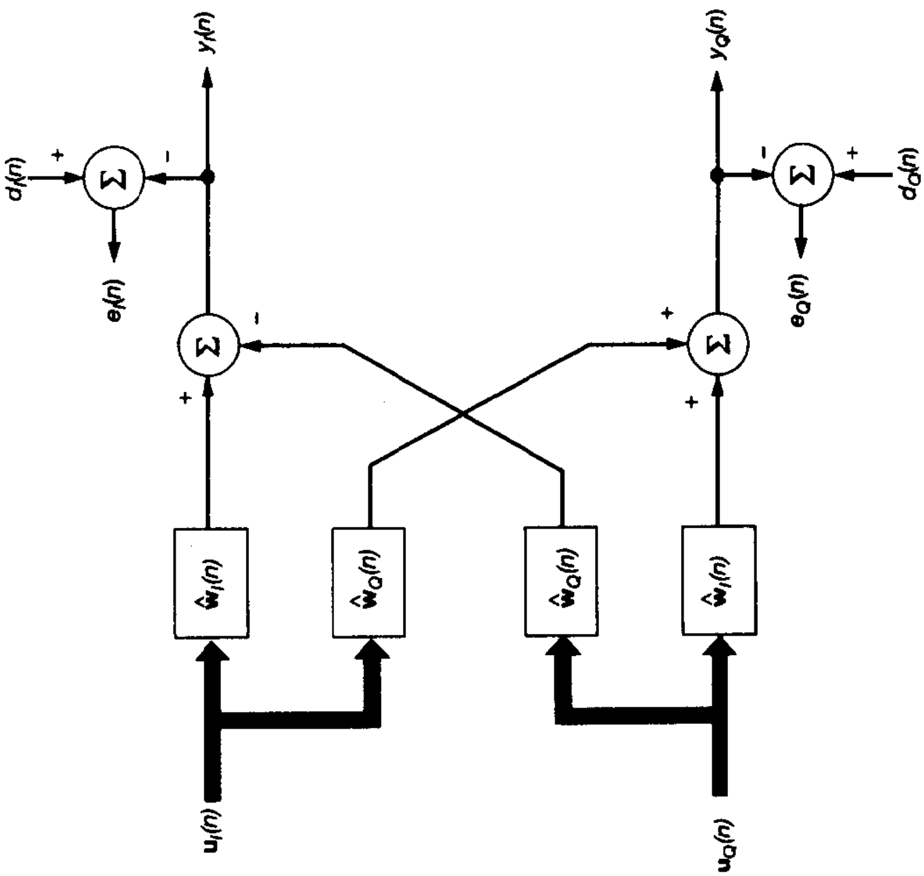
- The operator $\hat{\mathbf{w}}(n)$ is updated:

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \mu[u(n+\Delta) - \hat{u}(n+\Delta)]\mathbf{u}(n) \tag{9.21}$$

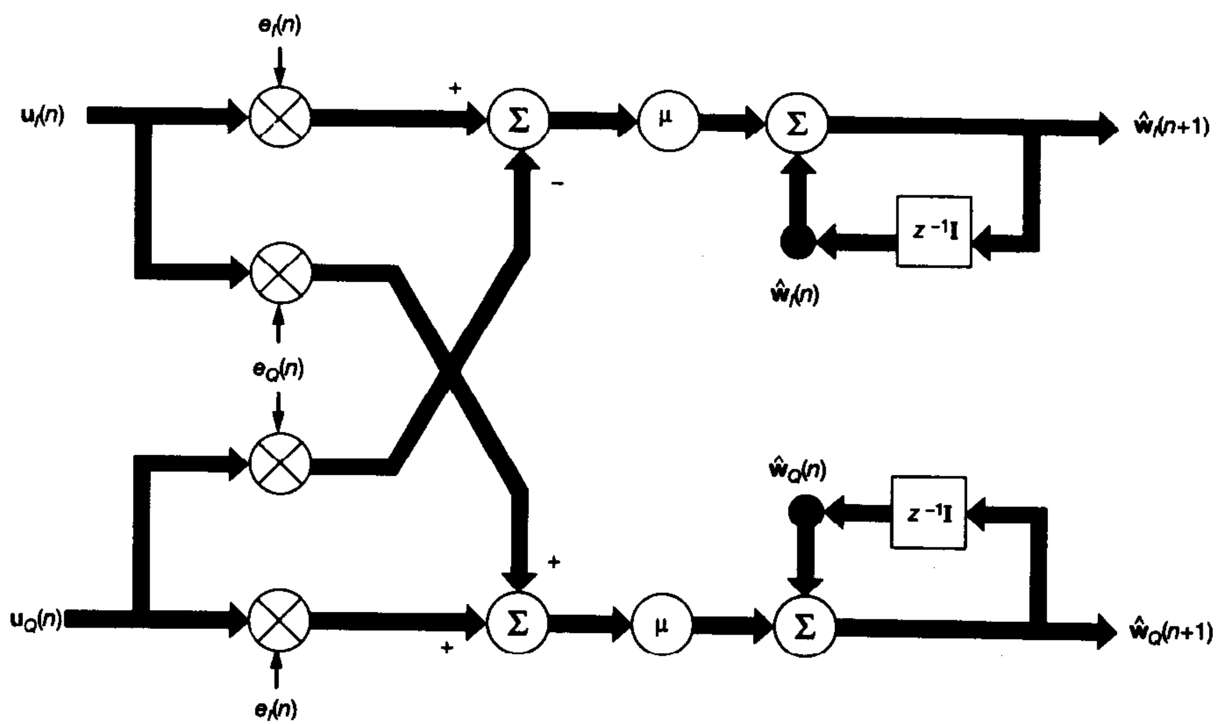
Equations (9.20) and (9.21) constitute the LMS-based adaptive seismic deconvolution algorithm. The adaptation is begun with an initial guess $\hat{\mathbf{w}}(0)$.

Example 3: Instantaneous Frequency Measurement

In this example we study the use of the LMS algorithm as the basis for estimating the frequency content of a *narrow-band signal* characterized by a rapidly varying power spectrum (Griffiths, 1975). In so doing we illustrate the linkage between three basic ideas: an autoregressive (AR) model for describing a stochastic process (studied in Chapter 2), a linear


$\mathbf{u}_I(n)$
$\mathbf{u}_Q(n)$
$\hat{\mathbf{w}}_I(n)$
$\hat{\mathbf{w}}_Q(n)$
$\hat{\mathbf{w}}_Q(n)$
$\hat{\mathbf{w}}_I(n)$
Σ
$y_I(n)$
$y_Q(n)$
$e_I(n)$
$e_Q(n)$
$d_I(n)$
$d_Q(n)$

(a)



(b)

Figure 9.3 Canonical signal-flow graph representation of the complex LMS algorithm. (a) Error and output signals. (b) Tap-weight update equation.

predictor for analyzing the process (studied in Chapter 6), and the LMS algorithm for estimating the AR parameters.

By a "narrow-band signal" we mean a signal whose bandwidth Ω is small compared to the midband angular frequency ω_c, as illustrated in Fig. 9.4. A *frequency-modulated (FM) signal* is an example of a narrow-band signal, provided that the carrier frequency is high enough. The *instantaneous frequency* (defined as the derivative of phase with respect to time) of an FM signal varies linearly with the modulating signal. Consider then a narrow-band process $u(n)$ generated by a *time varying* AR model of order M, as shown by the difference equation (assuming *real* data):

$$u(n) = -\sum_{k=1}^{M} a_k(n)u(n-k) + v(n) \tag{9.22}$$

where the $a_k(n)$ are the time-varying model parameters, and $v(n)$ is a zero-mean white-noise process of time-varying variance $\sigma_v^2(n)$. The *time-varying AR (power) spectrum* of the narrow-band process $u(n)$ is given by

$$S_{\text{AR}}(\omega; n) = \frac{\sigma_v^2(n)}{\left|1 + \sum_{k=1}^{M} a_k(n)e^{-j\omega k}\right|^2}, \qquad -\pi < \omega \leq \pi \tag{9.23}$$

Note that an AR process whose poles are near the unit circle in the z-plane has the characteristics of a narrow-band process.

To estimate the model parameters, we use an adaptive transversal filter employed as a linear predictor of order M. Let the tap weights of the predictor be denoted by $\hat{w}_k(n)$, $k = 1, 2, \ldots, M$. The tap weights are adapted continuously as the input signal $u(n)$ is received. In particular, we use the LMS algorithm for adapting the tap weights, as shown by

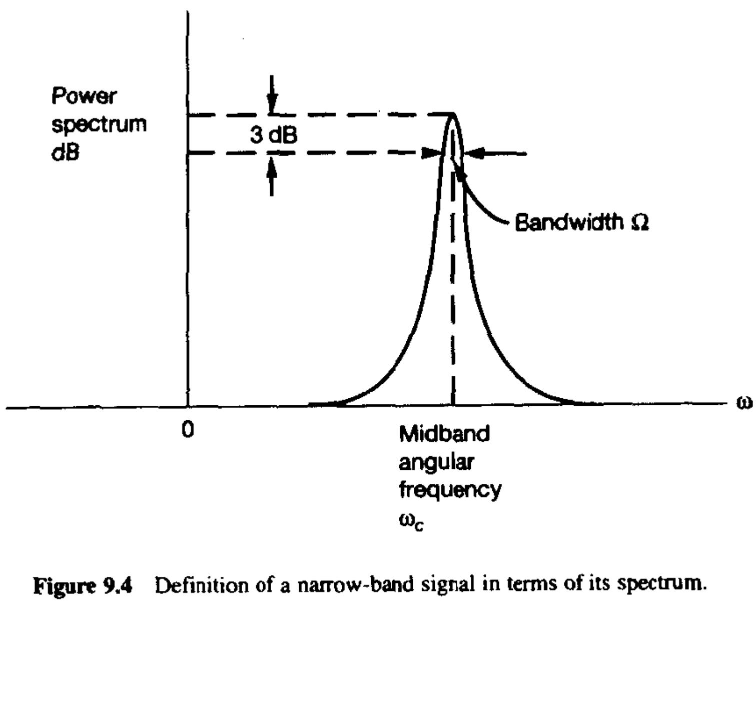


Figure 9.4 Definition of a narrow-band signal in terms of its spectrum.

$$\hat{w}_k(n+1) = \hat{w}_k(n) + \mu u(n-k) f_M(n), \qquad k = 1, 2, \ldots, M \tag{9.24}$$

where $f_M(n)$ is the *prediction error*:

$$f_M(n) = u(n) - \sum_{k=1}^{M} \hat{w}_k(n) u(n-k) \tag{9.25}$$

The tap weights of the adaptive predictor are related to the AR model parameters as follows:

$$-\hat{w}_k(n) = \text{estimate of } a_k(n) \text{ at time } n \qquad \text{for } k = 1, 2, \ldots, M$$

Moreover, the average power of the prediction error $f_M(n)$ provides an estimate of the noise variance $\sigma_v^2(n)$. Our interest is in locating the frequency of a narrow-band signal. Accordingly, in the sequel, we ignore the estimation of $\sigma_v^2(n)$. Specifically, we only use the tap weights of the adaptive predictor to define the *time-varying frequency function*:

$$F(\omega; n) = \frac{1}{\left|1 - \sum_{k=1}^{M} \hat{w}_k(n) e^{-j\omega k}\right|^2} \tag{9.26}$$

Given the relationship between $\hat{w}_k(n)$ and $a_k(n)$, we see that the essential difference be-tween the frequency function $F(\omega; n)$ in Eq. (9.26) and the AR power spectrum $S_{AR}(\omega; n)$ in Eq. (9.23) lies in their numerator scale factors. The numerator of $F(\omega; n)$ is a constant equal to 1, whereas that of $S_{AR}(\omega; n)$ is a time-varying constant equal to $\sigma_v^2(n)$. The advantages of the frequency function $F(\omega; n)$ over the AR spectrum $S_{AR}(\omega; n)$ are twofold. First, the 0/0 indeterminacy inherent in the narrow-band spectrum of Eq. (9.23) is replaced by a "computationally tractable" limit of 1/0 in Eq. (9.26). Second, the frequency function $F(\omega; n)$ is not affected by amplitude scale changes in the input signal $u(n)$, with the result that the peak value of $F(\omega; n)$ is related directly to the spectral width of the input signal.

We may use the function $F(\omega; n)$ to measure the instantaneous frequency of a frequency-modulated signal $u(n)$, provided that the following assumptions are justified (Griffiths, 1975):

- The adaptive predictor has been in operation sufficiently long, so as to ensure that any transients caused by the initialization of the tap weights have died out.
- The step-size parameter μ is chosen correctly for the adaptive predictor to track well; that is to say, the prediction error $f_M(n)$ is small for all n.
- The modulating signal is essentially constant over the sampling range of the adaptive predictor, which extends from time $(n - M)$ to time $(n - 1)$.

Given the validity of these assumptions, we find that the frequency function $F(\omega; n)$ has a peak at the instantaneous frequency of the input signal $u(n)$. Moreover, the LMS algorithm will track the time variation of the instantaneous frequency.

Example 4: Adaptive Noise Canceling Applied to a Sinusoidal Interference

The traditional method of suppressing a *sinusoidal interference* corrupting an information-bearing signal is to use a fixed *notch filter* tuned to the frequency of the interference. To design the filter, we naturally need to know the precise frequency of the interference. But what if the notch is required to be very sharp and the interfering sinusoid is known to *drift* slowly? Clearly, then, we have a problem that calls for an *adaptive* solution. One such solution is provided by the use of adaptive noise canceling.

Figure 9.5 shows the block diagram of a dual-input *adaptive noise canceler*. The primary input supplies an information-bearing signal and a sinusoidal interference that are uncorrelated with each other. The reference input supplies a correlated version of the sinusoidal interference. For the adaptive filter, we may use a transversal filter whose tap weights are adapted by means of the LMS algorithm. The filter uses the reference input to provide (at its output) an estimate of the sinusoidal interfering signal contained in the primary input. Thus, by subtracting the adaptive filter output from the primary input, the effect of the sinusoidal interference is diminished. In particular, an adaptive noise canceler using the LMS algorithm has two important characteristics (Widrow et al., 1976; Glover, 1977):

1. The canceler behaves as an adaptive notch filter whose null point is determined by the angular frequency ω_0 of the sinusoidal interference. Hence, it is tunable, and the tuning frequency moves with ω_0.
2. The notch in the frequency response of the canceler can be made very sharp at precisely the frequency of the sinusoidal interference by choosing a small enough value for the step-size parameter μ.

In the example considered here, the input data are assumed to be real valued:

- *Primary input:*

$$d(n) = s(n) + A_0 \cos(\omega_0 n + \phi_0) \tag{9.27}$$

where $s(n)$ is an information-bearing signal; A_0 is the amplitude of the sinusoidal interference, ω_0 is the *normalized* angular frequency, and ϕ_0 is the phase.

- *Reference input:*

$$u(n) = A \cos(\omega_0 n + \phi) \tag{9.28}$$

where the amplitude A and the phase ϕ are different from those in the primary input, but the angular frequency ω_0 is the same.

Using the real, expanded form of the LMS algorithm, the tap-weight update is described by the following equations:

$$y(n) = \sum_{i=0}^{M-1} \hat{w}_i(n)u(n-i) \tag{9.29}$$

$$e(n) = d(n) - y(n) \tag{9.30}$$

$$\hat{w}_i(n+1) = \hat{w}_i(n) + \mu u(n-i)e(n), \qquad i = 0, 1, \ldots, M-1 \tag{9.31}$$

where M is the total number of tap weights in the transversal filter, and the constant μ is the step-size parameter. Note that the sampling period in the input data and all other signals in the LMS algorithm is assumed to be unity for convenience of presentation; as mentioned previously this practice is indeed followed throughout the book.

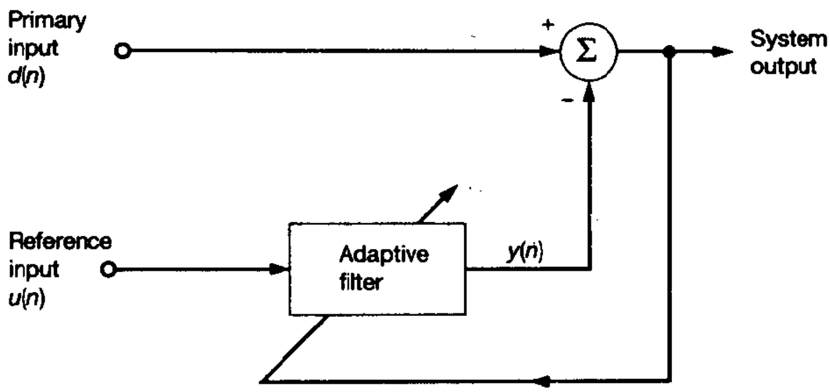


Figure 9.5 Block diagram of adaptive noise canceler.

With a sinusoidal excitation as the input of interest, we restructure the block diagram of the adaptive noise canceler as in Fig. 9.6(a). According to this new representation, we may lump the sinusoidal input $u(n)$, the transversal filter, and the weight-update equation of the LMS algorithm into a single (open-loop) system defined by a transfer function $G(z)$, as in the equivalent model of Fig. 9.6(b). The transfer function $G(z)$ is

$$G(z) = \frac{Y(z)}{E(z)}$$

where $Y(z)$ and $E(z)$ are the z-transforms of the reference input $u(n)$ and the estimation error $e(n)$, respectively. Given $E(z)$, our task is to find $Y(z)$, and therefore $G(z)$.

To do so, we use the detailed signal-flow-graph representation of the LMS algorithm depicted in Fig. 9.7 (Glover, 1977). In this diagram, we have singled out the ith tap weight for specific attention. The corresponding value of the tap input is

$$\begin{aligned} u(n-i) &= A\cos[\omega_0(n-i)+\phi] \\ &= \frac{A}{2}[e^{j(\omega_0 n+\phi_i)} + e^{-j(\omega_0 n+\phi_i)}] \end{aligned} \tag{9.32}$$

where

$$\phi_i = \phi - \omega_0 i$$

In Fig. 9.7, the input $u(n-i)$ is multiplied by the estimation error $e(n)$. Hence, taking the z-transform of the product $u(n-i)e(n)$ and using $z[\cdot]$ to denote this operation, we obtain

$$z[u(n-i)e(n)] = \frac{A}{2}e^{j\phi_i}E(ze^{-j\omega_0}) + \frac{A}{2}e^{-j\phi_i}E(ze^{j\omega_0}) \tag{9.33}$$

where $E(ze^{-j\omega_0})$ is the z-transform $E(z)$ rotated counterclockwise around the unit circle through the angle ω_0. Similarly, $E(ze^{j\omega_0})$ represents a clockwise rotation through ω_0.

Next, taking the z-transform of Eq. (9.31), we get

$$z\hat{W}_i(z) = \hat{W}_i(z) + \mu\, z[u(n-i)e(n)] \tag{9.34}$$

where $\hat{W}_i(z)$ is the z-transform of $\hat{w}_i(n)$. Solving Eq. (9.34) for $\hat{W}_i(z)$ and using the z-transform

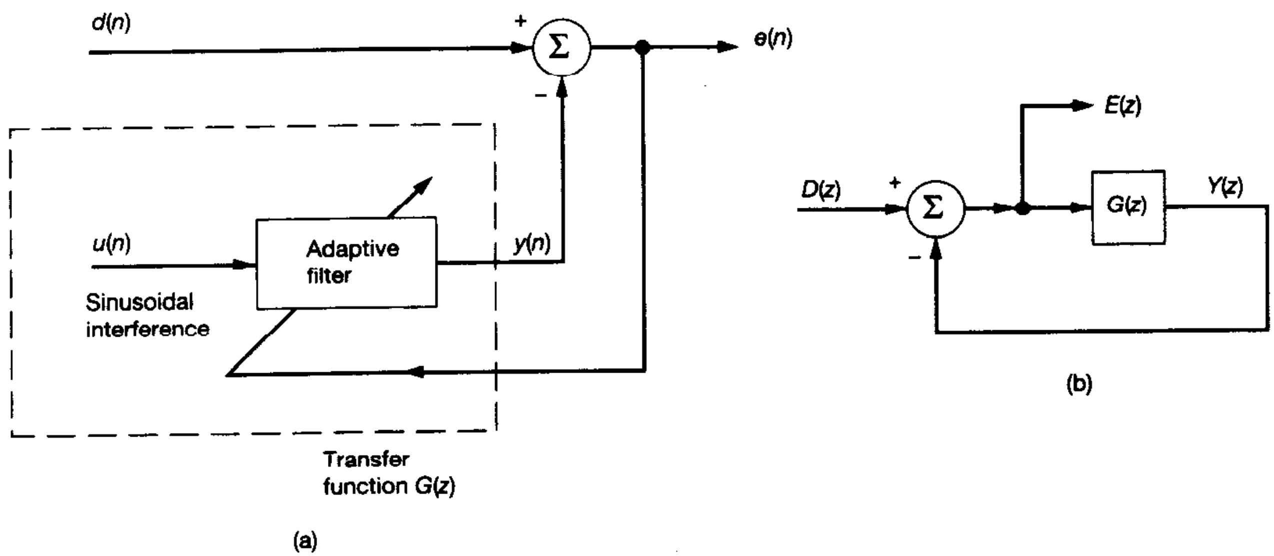


Figure 9.6 (a) New representation of adaptive noise canceler, (b) Equivalent model in the z-domain.

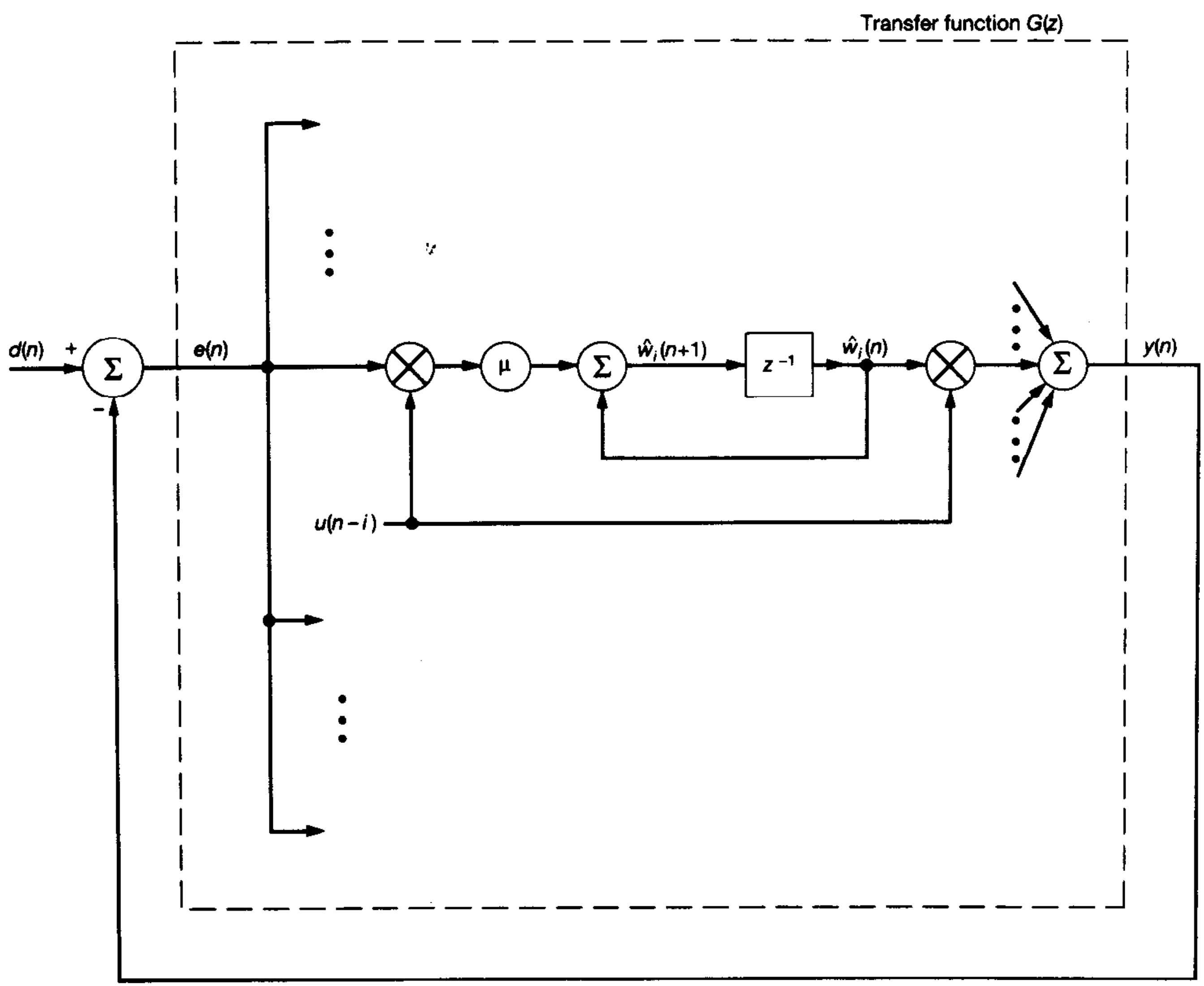


Figure 9.7 Signal-flow graph representation of adaptive noise canceler, singling out the ith tap weight for detailed attention.

given in Eq. (9.33), we get

$$\hat{W}_i(z) = \frac{\mu A}{2}\frac{1}{z-1}[e^{j\phi_i}E(ze^{-j\omega_0}) + e^{-j\phi_i}E(ze^{j\omega_0})] \tag{9.35}$$

We turn next to Eq. (9.29) that defines the adaptive filter output $y(n)$. Substituting Eq. (9.32) in (9.29), we get

$$y(n) = \frac{A}{2}\sum_{i=0}^{M-1}\hat{w}_i(n)[e^{j(\omega_0 n+\phi_i)} + e^{-j(\omega_0 n+\phi_i)}]$$

Hence, evaluating the z-transform of $y(n)$, we obtain

$$Y(z) = \frac{A}{2}\sum_{i=0}^{M-1}[e^{j\phi_i}\hat{W}_i(ze^{-j\omega_0}) + e^{-j\phi_i}\hat{W}_i(ze^{j\omega_0})] \tag{9.36}$$

Thus, using Eq. (9.35) in (9.36), we obtain an expression for $Y(z)$ that consists of the sum of two components (Glover, 1977):

1. A *time-invariant component* defined by

$$\frac{\mu M A^2}{4}\left(\frac{1}{ze^{-j\omega_0}-1} + \frac{1}{ze^{j\omega_0}-1}\right)$$

which is independent of the phase ϕ_i and, therefore, the time index i.

2. A *time-varying component* that is dependent on the phase ϕ_i, hence the variation with time i. This second component is scaled in amplitude by the factor

$$\beta(\omega_0, M) = \frac{\sin(M\omega_0)}{\sin\omega_0}$$

For a given angular frequency ω_0, we assume that the total number of tap weights M in the transversal filter is large enough to satisfy the following approximation:

$$\frac{\beta(\omega_0, M)}{M} = \frac{\sin(M\omega_0)}{M\sin\omega_0} \approx 0 \tag{9.37}$$

Accordingly, we may justifiably ignore the time-varying component of the z-transform $Y(z)$, and so approximate $Y(z)$ by retaining the time-invariant component only:

$$Y(z) \approx \frac{\mu M A^2}{4}E(z)\left(\frac{1}{ze^{-j\omega_0}-1} + \frac{1}{ze^{j\omega_0}-1}\right) \tag{9.38}$$

The open-loop transfer function $G(z)$ is therefore

$$\begin{aligned} G(z) &= \frac{Y(z)}{E(z)} \\ &\approx \frac{\mu M A^2}{4}\left(\frac{1}{ze^{-j\omega_0}-1} + \frac{1}{ze^{j\omega_0}-1}\right) \qquad (9.39) \\ &\approx \frac{\mu M A^2}{2}\left(\frac{z\cos\omega_0 - 1}{z^2 - 2z\cos\omega_0 + 1}\right) \end{aligned}$$

The transfer function $G(z)$ has two complex-conjugate poles on the unit circle at $z = e^{\pm j\omega_0}$ and a real zero at $z = 1/\cos\omega_0$, as illustrated in Fig. 9.8(a). In other words, the adaptive noise canceler has a null point determined by the angular frequency ω_0 of the sinusoidal interference, as stated previously (see characteristic 1). Indeed, according to Eq. (9.39), we may view $G(z)$ as a pair of *integrators* that have been rotated by $\pm\omega_0$. In actual fact, we see from Fig. 9.7 that it is the input that is first shifted in frequency by an amount $\pm\omega_0$ due to the first multiplication by the reference sinusoid $u(n)$, digitally integrated at zero frequency, and then shifted back again by the second multiplication. This overall operation is similar to a well-known technique in communications for obtaining a resonant filter that involves the combined use of two low-pass filters and heterodyning with sine and cosine at the resonant frequency (Wozencraft and Jacobs, 1965; Glover, 1977).

The model of Fig. 9.6 is recognized as a *closed-loop feedback system*, whose transfer function $H(z)$ is related to the open-loop transfer function $G(z)$ as follows:

$$H(z) = \frac{E(z)}{D(z)} = \frac{1}{1 + G(z)} \tag{9.40}$$

where $E(z)$ is the z-transform of the system output $e(n)$, and $D(z)$ is the z-transform of the system input $d(n)$. Accordingly, substituting Eq. (9.39) in (9.40), we get the approximate result

$$H(z) \approx \frac{z^2 - 2z\cos\omega_0 + 1}{z^2 - 2(1 - \mu MA^2/4)z\cos\omega_0 + (1 - \mu MA^2/2)} \tag{9.41}$$

Equation (9.41) is the transfer function of a *second-order digital notch filter* with a notch at the normalized angular frequency ω_0. The zeros of $H(z)$ are at the poles of $G(z)$; that is, they are located on the unit circle at $z = e^{\pm j\omega_0}$. For a small value of the step-size parameter μ (i.e., a slow adaptation rate), such that

$$\frac{\mu MA^2}{4} \ll 1$$

we find that the poles of $H(z)$ are approximately located at

$$z \approx \left(1 - \frac{\mu MA^2}{4}\right)e^{\pm j\omega_0} \tag{9.42}$$

In other words, the two poles of $H(z)$ lie inside the unit circle, a radial distance approximately equal to $\mu MA^2/4$ behind the zeros, as indicated in Fig. 9.8(b). The fact that the poles of $H(z)$ lie inside the unit circle means that the adaptive noise canceler is stable, as it should be for practical use in real time.

Figure 9.8(b) also includes the *half-power points* of $H(z)$. Since the zeros of $H(z)$ lie on the unit circle,the adaptive noise canceler has (in theory) a notch of infinite depth (in dB) at $\omega = \omega_0$. The *sharpness* of the notch is determined by the closeness of the poles of $H(z)$ to its zeros. The *3-dB bandwidth, B*, is determined by locating the two half-power points on the unit circle that are $\sqrt{2}$ times as far from the poles as they are from the zeros. Using this geometric approach, we find that the 3-dB bandwidth of the adaptive noise canceler is approximately

$$B \approx \frac{\mu MA^2}{2} \text{ radians} \tag{9.43}$$

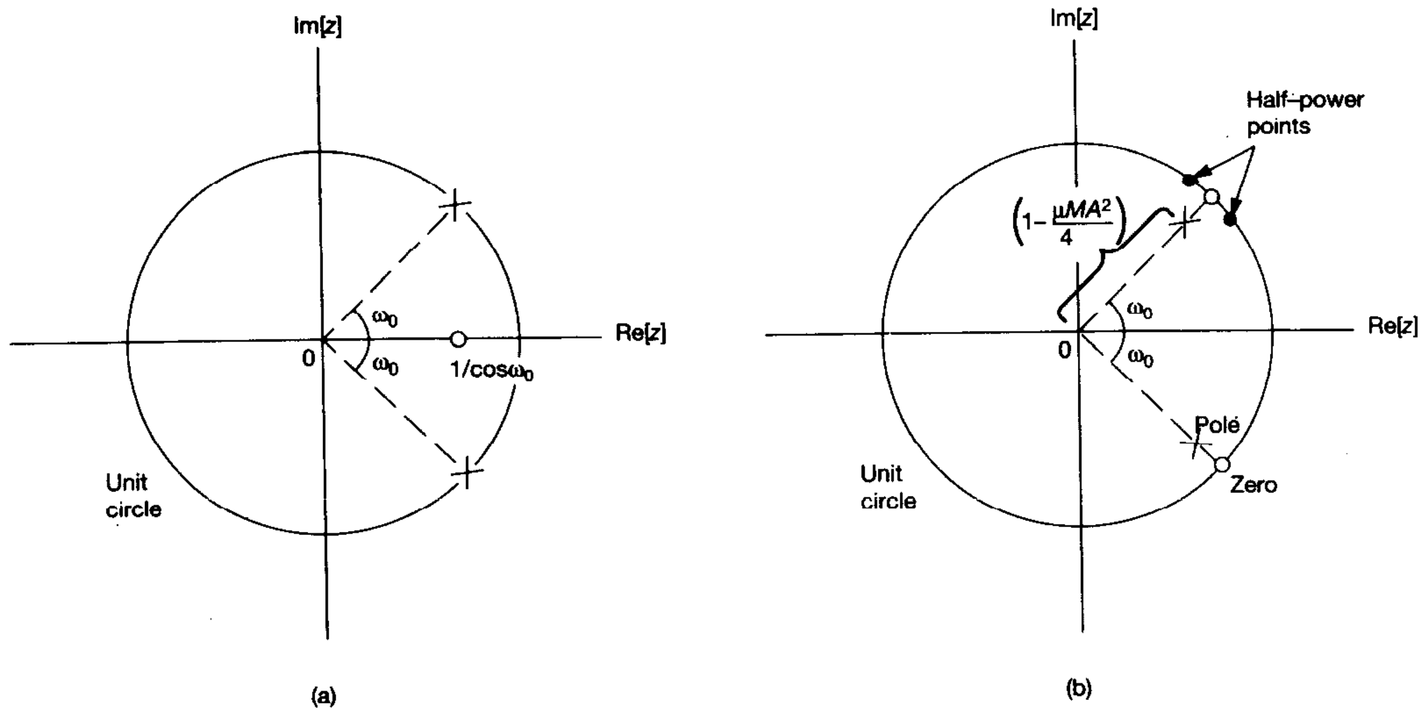


Figure 9.8 Approximate pole-zero patterns of (a) the open-loop transfer function $G(z)$. and (b) the closed-loop transfer function $H(z)$.

The smaller we therefore make μ, the smaller the bandwidth B is, and therefore the sharper the notch is. This confirms characteristic 2 of the adaptive noise canceler that was mentioned previously. Its analysis is thereby completed.

Example 5: Adaptive Line Enhancement

The *adaptive line enhancer (ALE)*, illustrated in Fig. 9.9, is a device that may be used to detect a periodic signal buried in a broad-band noise background.[2] This figure shows that the ALE is in fact a degenerate form of the adaptive noise canceler in that its reference signal, instead of being derived separately, consists of a delayed version of the primary (input) signal. The delay, denoted by Δ in Fig. 9.9, is called the *prediction depth* or *decorrelation delay* of the ALE, measured in units of the sampling period. The reference signal $u(n - \Delta)$ is processed by a transversal filter to produce an error signal $e(n)$, defined as the difference between the actual input $u(n)$ and the ALE's output $y(n) = u(n)$. The error signal $e(n)$ is, in turn, used to actuate the LMS algorithm for adjusting the M tap weights of the transversal filter.

Consider an input signal $u(n)$ that consists of a sinusoidal component $A\sin(\omega_0 n + \phi_0)$ buried in wide-band noise $\nu(n)$, as shown by

$$u(n) = A \sin(\omega_0 n + \phi_0) + \nu(n) \tag{9.44}$$

where ϕ_0 is an arbitrary phase shift, and the noise $\nu(n)$ is assumed to have zero mean and variance σ_ν^2. The ALE acts as a signal detector by virtue of two actions (Treichler, 1979):

- The prediction depth Δ is assigned a value large enough to *remove* the correlation between the noise $\nu(n)$ in the original input signal and the noise $\nu(n - \Delta)$ in the reference signal, while a simple phase shift equal to $\omega_0\Delta$ is introduced between the sinusoidal components in these two inputs.
- The tap weights of the transversal filter are adjusted by the LMS algorithm so as to minimize the mean-square value of the error signal and thereby compensate for the phase shift $\omega_0\Delta$.

The net result of these two actions is the production of an output signal $y(n)$ that consists of a scaled sinusoid in noise of zero mean. In particular, when ω_0 is several multiples of π/M away from zero or π, Rickard and Zeidler (1979) have shown that

$$y(n) = aA \sin(\omega_0 n + \phi) + \nu_{out}(n) \tag{9.45}$$

where ϕ denotes a phase shift, and $\nu_{out}(n)$ denotes the output noise. The scaling factor a is defined by

$$a = \frac{(M/2)\,\text{SNR}}{1 + (M/2)\,\text{SNR}} \tag{9.46}$$

[2]The ALE owes its origin to Widrow et al. (1975). For a statistical analysis of its performance for the detection of sinusoidal signals in wide-band noise, see Zeidler et al. (1978), Treichler (1979), and Rickard and Zeidler (1979). For a tutorial treatment of the ALE, see Zeidler (1990); the effects of signal bandwidth, input SNR, noise correlation, and noise nonstationarity are explicitly considered in Zeidler's paper.

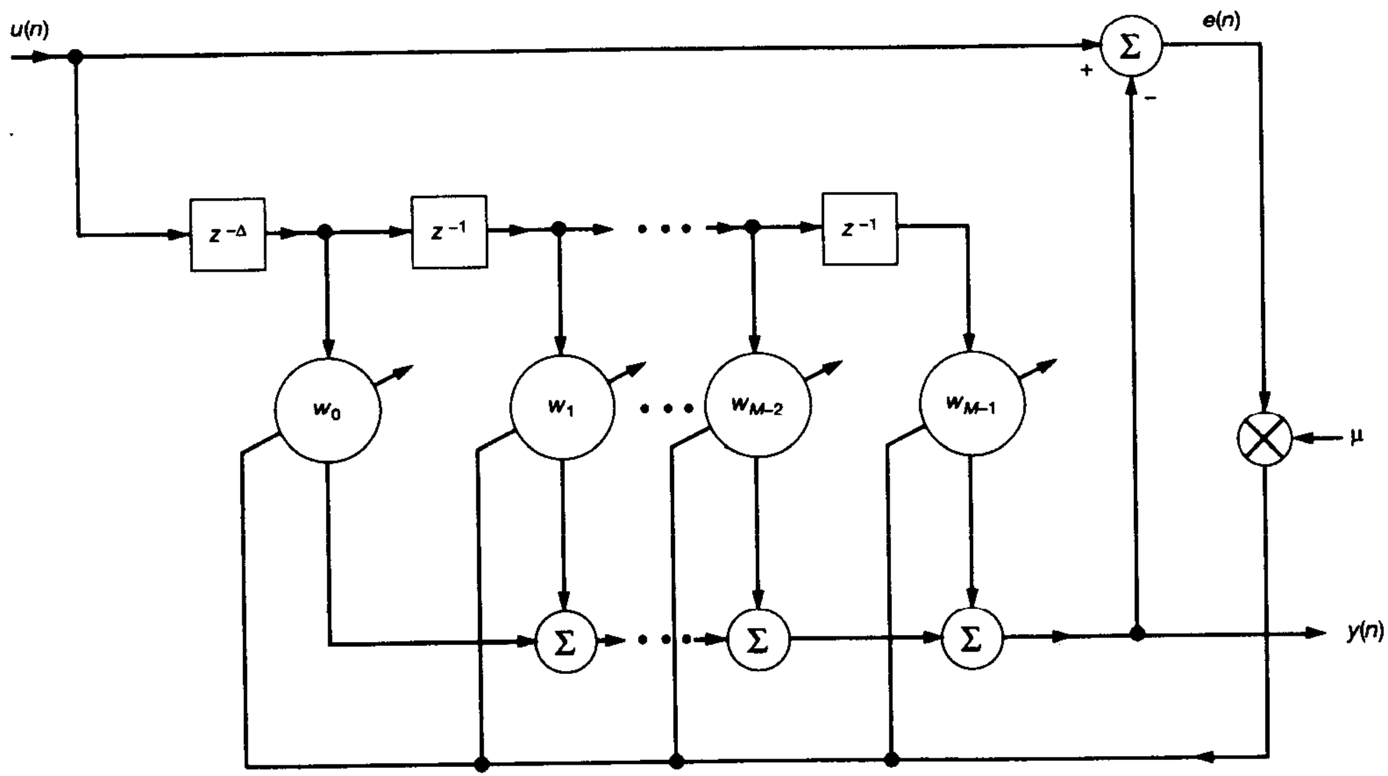


Figure 9.9 Adaptive line enhancer.

where M is the length of the transversal filter, and SNR denotes the *signal-to-noise ratio* at the input of the ALE:

$$\text{SNR} = \frac{A^2}{2\sigma_\nu^2} \tag{9.47}$$

According to Eq. (9.45), the ALE acts as a *self-tuning filter* whose frequency response exhibits a peak at the angular frequency ω_0 of the incoming sinusoid, hence the name "spectral line enhancer" or simply "line enhancer."

Rickard and Zeidler (1979) have also shown that the power spectral density of the ALE output $y(n)$ may be expressed as

$$S(\omega) = \frac{\pi A^2}{2}(a^2 + \mu\sigma_\nu^2 M)\,[\delta(\omega - \omega_0) + \delta(\omega + \omega_0)] + \mu\sigma_\nu^4 M$$

$$+ \frac{a^2\sigma^2_\nu}{M^2}\left[\frac{1 - \cos M(\omega - \omega_0)}{1 - \cos(\omega - \omega_0)} + \frac{1 + \cos M(\omega - \omega_0)}{1 + \cos(\omega - \omega_0)}\right], \quad -\pi < \omega \leq \pi \tag{9.48}$$

where $\delta(\cdot)$ denotes a Dirac delta function. To understand the composition of Eq. (9.48), we first recall from the overview of the LMS algorithm presented in Section 9.1 that, in a stationary environment, the mean of the weight vector $\hat{\mathbf{w}}(n)$ converges to the Wiener solution $\mathbf{w}_o(n)$. A formal analysis of this behavior is presented in the next section. For now it suffices to say that the steady-state model of the converged weight vector consists of the Wiener solution $\mathbf{w}_o$ acting in parallel with a slowly fluctuating, zero-mean random component $\boldsymbol{\epsilon}(n)$ due to gradient noise. The ALE may thus be modeled as shown in Fig. 9.10.

Recognizing that the ALE input itself consists of two components, a sinusoid of angular frequency ω_0 and a wide-band noise $\nu(n)$ of zero mean and variance σ_ν^2, we may distinguish four components in the power spectrum of Eq. (9.48) as described here (Zeidler, 1990):

- A sinusoidal component of angular frequency ω_0 and average power $\pi a^2 A^2/2$, which is the result of processing the input sinusoid by the Wiener filter represented by the weight vector $\mathbf{w}_o$
- A sinusoidal component of angular frequency ω_0 and average power $\pi\mu A^2\sigma_\nu^2 M/2$, which is due to the stochastic filter represented by the weight vector $\boldsymbol{\epsilon}(n)$ acting on the input sinusoid
- A wide-band noise component of variance $\mu\sigma_\nu^4 M$, which is due to the action of the stochastic filter on the noise $\nu(n)$
- A narrow-band filtered noise component centered on ω_0, which results from processing the noise $\nu(n)$ by the Wiener filter

These four components are depicted in Fig. 9.11, respectively. Thus, the power spectrum of the ALE output consists of a sinusoid at the center of a pedestal of narrow-band filtered noise, the combination of which is embedded in a wide-band noise background. Most importantly, when an adequate SNR exists at the ALE input, the ALE output is, on the average, approximately equal to the sinusoidal component present at the input, thereby providing a simple adaptive device for the detection of a sinusoidal in wide-band noise.

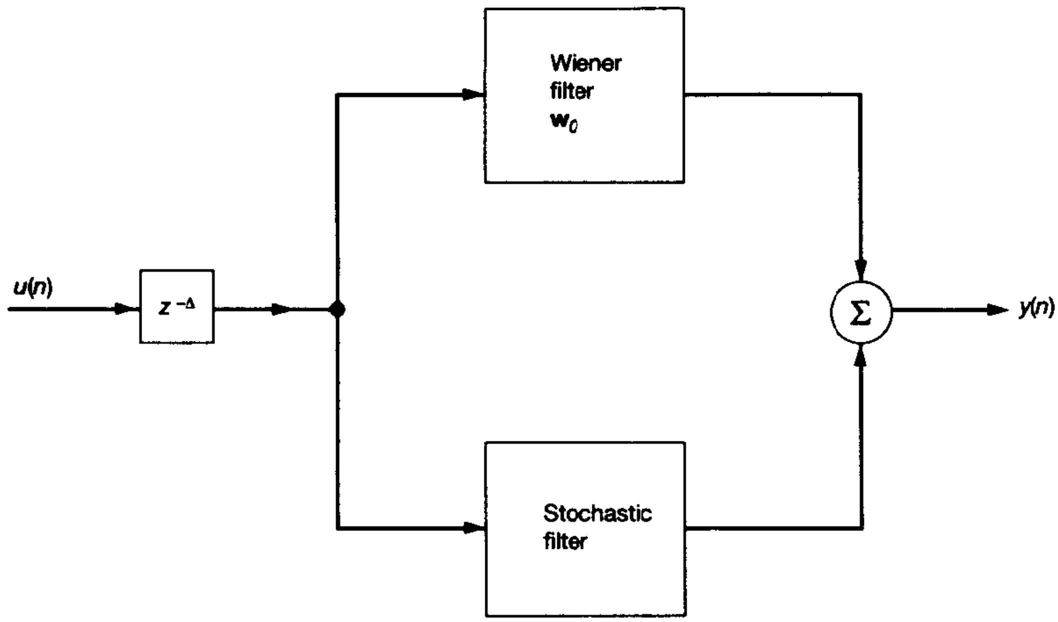


Figure 9.10 Model of the adaptive line enhancer.

Example 6: Adaptive Beamforming

In this final example, we consider a spatial application of the LMS algorithm, namely, that of adaptive beamforming. In particular, we revisit the *generalized sidelobe canceler* (GSC) that was studied under the umbrella of Wiener filter theory in Chapter 5.

Figure 9.12 shows a block diagram of the GSC, the operation of which hinges on the combination of two actions:

- The imposition of linear multiple constraints, designed to preserve an incident signal along a direction of interest
- The adjustment of some weights, in accordance with the LMS algorithm, so as to minimize the effects of interference and noise at the beamformer output

The multiple linear constraints are described by an M-by-L matrix $\mathbf{C}$, on the basis of which a signal blocking matrix $\mathbf{C}_a$ of size M-by-$(M - L)$ is defined by

$$\mathbf{C}_a^H\mathbf{C} = \mathbf{O} \tag{9.49}$$

In the GSC, the vector of weights assigned to the linear array of antenna elements is represented by

$$\mathbf{w}(n) = \mathbf{w}_q - \mathbf{C}_a\mathbf{w}_a(n) \tag{9.50}$$

where $\mathbf{w}_a(n)$ is the *adjustable-weight vector* and $\mathbf{w}_q$ is the *quiescent-weight vector*. The latter component is defined in terms of the constraint matrix $\mathbf{C}$ by

$$\mathbf{w}_q = \mathbf{C}(\mathbf{C}^H\mathbf{C})^{-1}\mathbf{g} \tag{9.51}$$

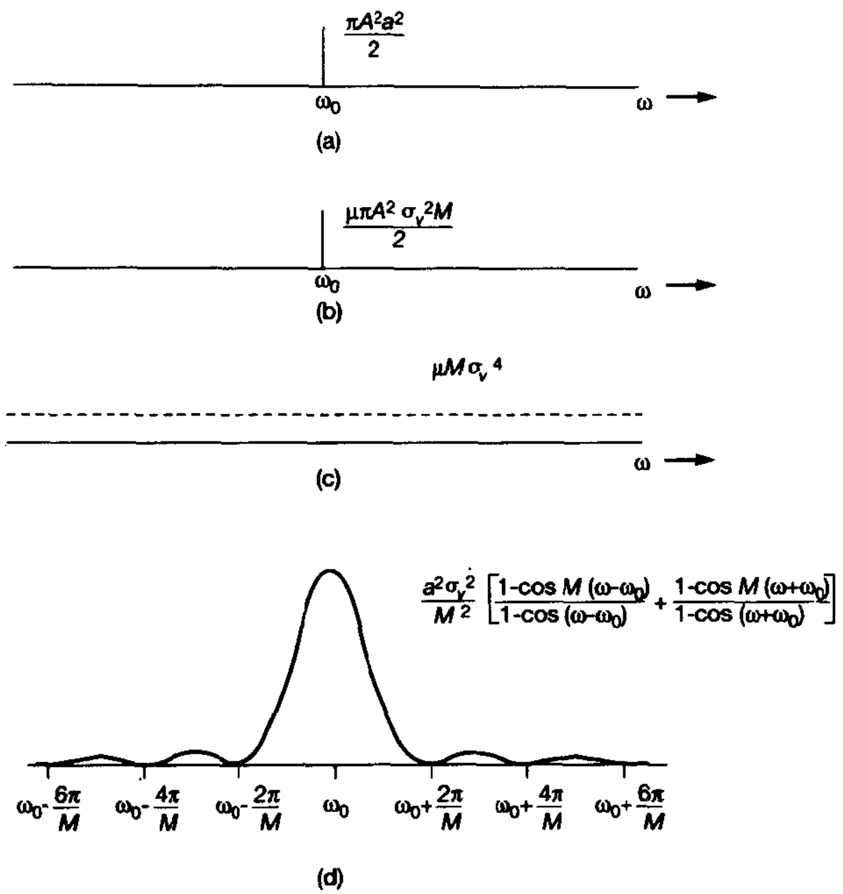


Figure 9.11 The four primary spectral components of the power spectral density at the ALE output. (a) Component due to Wiener filter acting on the input sinusoid. (b) Component due to stochastic filter acting on the input sinusoid. (c) Wide-band due to the stochastic filter acting on the noise $\nu(n)$. (d) Narrow-band noise due to the Wiener filter acting on $\nu(n)$.

where $\mathbf{g}$ is a prescribed gain vector.

The beamformer output is

$$\begin{aligned} e(n) &= \mathbf{w}^H(n)\mathbf{u}(n) \\ &= (\mathbf{w}_q - \mathbf{C}_a\mathbf{w}_a(n))^H\mathbf{u}(n) \\ &= \mathbf{w}_q^H\mathbf{u}(n) - \mathbf{w}_a^H(n)\mathbf{C}_a^H\mathbf{u}(n) \end{aligned} \tag{9.52}$$

In words, the quiescent weight vector $\mathbf{w}_q$ influences that part of the input vector $\mathbf{u}(n)$ that lies in the subspace spanned by the columns of constraint matrix $\mathbf{C}$, whereas the adjustable weight vector $\mathbf{w}_a(n)$ influences the remaining part of the input vector $\mathbf{u}(n)$ that lies in the complementary subspace spanned by the columns of signal blocking matrix $\mathbf{C}_a$. Note also the $e(n)$ in Eq. (9.52) is the same as $y(n)$ in Eq. (5.118).

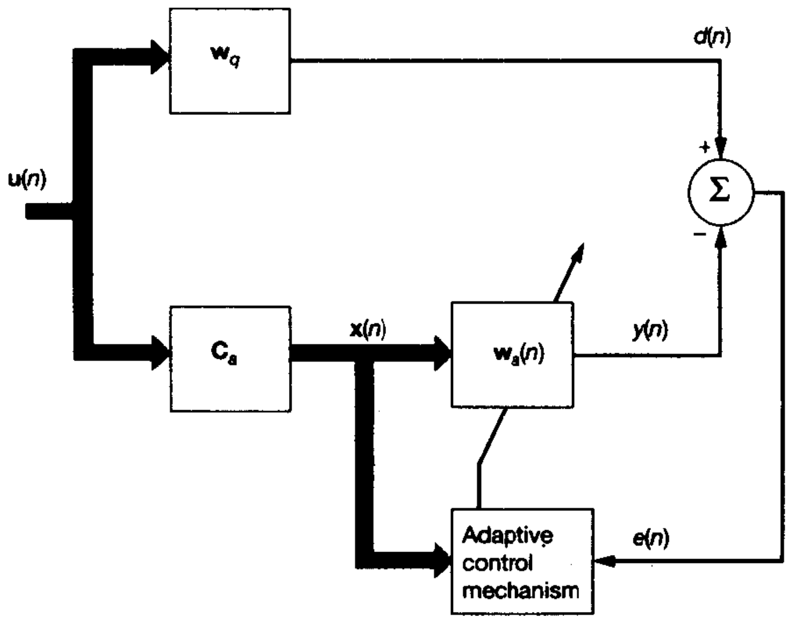


Figure 9.12 Block diagram of the generalized sidelobe canceler.

According to Eq. (9.52), the inner product $\mathbf{w}_q^H\mathbf{u}(n)$ plays the role of desired response:

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

By the same token, the matrix product $\mathbf{C}_a^H\mathbf{u}(u)$ plays the role of input vector for the adjustable weight vector $\mathbf{w}_a(n)$; to emphasize this point, we let

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

We are now ready to formulate the LMS algorithm for the adaptation of weight vector $\mathbf{w}_a(n)$ in the GSC. Specifically, we may write

$$\begin{aligned}\mathbf{w}_a(n+1) &= \mathbf{w}_a(n) + \mu\mathbf{x}(n)e^*(n)\\ &= \mathbf{w}_a(n) + \mu\mathbf{C}_a^H\mathbf{u}(n)\,(\mathbf{w}_q^H\mathbf{u}(n) - \mathbf{w}_a^H(n)\mathbf{C}_a^H\mathbf{u}(n))^* \qquad (9.53)\\ &= \mathbf{w}_a(n) + \mu\mathbf{C}_a^H\mathbf{u}(n)\mathbf{u}^H(n)\,(\mathbf{w}_q - \mathbf{C}_a\mathbf{w}_a(n))\end{aligned}$$

where μ is the step-size parameter, and all of the remaining quantities are displayed in the block diagram of Fig. 9.12.

9.4 STABILITY AND PERFORMANCE ANALYSIS OF THE LMS ALGORITHM

In this section we present a stability and performance analysis of the LMS algorithm that is largely based on the mean-squared value of the estimation error $e(n)$. For this analysis, we find it convenient to work with the weight-error vector rather than the tap-weight vector itself. To avoid confusion with the notation used in the study of the method of steepest descent in Chapter 8, we denote the *weight-error vector* in the LMS algorithm by

$$\boldsymbol{\epsilon}(n) = \hat{\mathbf{w}}(n) - \mathbf{w}_o \tag{9.54}$$

where, as before, $\mathbf{w}_o$ denotes the optimum Wiener solution for the tap-weight vector, and $\hat{\mathbf{w}}(n)$ is the estimate produced by the LMS algorithm at iteration n. Subtracting the optimum tap-weight vector $\mathbf{w}_o$ from both sides of Eq. (9.5), and using the definition of Eq. (9.54) to eliminate $\hat{\mathbf{w}}(n)$ from the correction term on the right-hand side of Eq. (9.5), we may rewrite the LMS algorithm in terms of the weight-error vector $\boldsymbol{\epsilon}(n)$ as follows:

$$\boldsymbol{\epsilon}(n + 1) = [\mathbf{I} - \mu\mathbf{u}(n)\mathbf{u}^H(n)]\boldsymbol{\epsilon}(n) + \mu\mathbf{u}(n)e_o^*(n) \tag{9.55}$$

where $\mathbf{I}$ is the identity matrix, and $e_o(n)$ is the *estimation error produced in the optimum Wiener solution*:

$$e_o(n) = d(n) - \mathbf{w}_o^H\mathbf{u}(n)$$

Direct-Averaging Method

Equation (9.55) is a *stochastic difference equation* in the weight-error vector $\boldsymbol{\epsilon}(n)$ with the following characteristic feature:

- A system matrix equal to $[\mathbf{I} - \mu\mathbf{u}(n)\mathbf{u}^H(n)]$, which is approximately equal to the identity matrix $\mathbf{I}$ for all n, provided that the step-size parameter μ is sufficiently small.

To study the convergence behavior of such a stochastic algorithm in an average sense, we may invoke the *direct-averaging method* described in Kushner (1984). According to this method, the solution of the stochastic difference equation (9.55), operating under the assumption of a *small* step-size parameter μ, is close to the solution of another stochastic difference equation whose system matrix is equal to the ensemble average:

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

where $\mathbf{R}$ is the correlation matrix of the tap-input vector $\mathbf{u}(n)$. More specifically, we may replace the stochastic difference equation (9.55) with another stochastic difference equation described by

$$\boldsymbol{\epsilon}(n + 1) = (\mathbf{I} - \mu\mathbf{R})\,\boldsymbol{\epsilon}(n) + \mu\mathbf{u}(n)e_o^*(n) \tag{9.56}$$

To be exact, the notation used in the new stochastic difference equation (9.56) should be different from that in the original stochastic difference equation (9.55). We have chosen *not* to do so here merely for convenience of presentation.

The direct averaging method is a reasonable heuristic, since it is based on the idea that, with small step sizes, the randomness of $\boldsymbol{\epsilon}(n)$ will tend to average out. The method applies rigorously, under appropriate conditions, to general stochastic approximation problems with small step sizes.

Independence Theory

In what follows, we will restrict ourselves to a statistical analysis of the LMS algorithm under the *independence assumption*, consisting of four points:

1. The tap-input vectors $\mathbf{u}(1), \mathbf{u}(2), \ldots \mathbf{u}(n)$ constitute a sequence of statistically independent vectors.
2. At time n, the tap-input vector $\mathbf{u}(n)$ is statistically independent of all previous samples of the desired response, namely, $d(1), d(2), \ldots, d(n-1)$.
3. At time n, the desired response $d(n)$ is dependent on the corresponding tap-input vector $\mathbf{u}(n)$, but statistically independent of all previous samples of the desired response.
4. The tap-input vector $\mathbf{u}(n)$ and the desired response $d(n)$ consist of *mutually Gaussian-distributed* random variables for all n.

The statistical analysis of the LMS algorithm so based is called the *independence theory*.[3]

From Eq. (9.5), we observe that the tap-weight vector $\hat{\mathbf{w}}(n+1)$ at time $n+1$ depends *only* on three inputs:

1. The previous sample vectors of the input process, $\mathbf{u}(n), \mathbf{u}(n-1), \ldots, \mathbf{u}(1)$.
2. The previous samples of the desired response, $d(n), d(n-1), \ldots, d(1)$.
3. The initial value of the tap-weight vector, $\hat{\mathbf{w}}(0)$.

Accordingly, in view of points 1 and 2 in the independence assumption, we find that the tap-weight vector $\hat{\mathbf{w}}(n+1)$, and therefore the weight-error vector $\boldsymbol{\epsilon}(n+1)$, is independent of both $\mathbf{u}(n+1)$ and $d(n+1)$. This is a very useful observation and one that will be used repeatedly in the sequel. The significance of the other two points in the independence assumption will become apparent as we proceed with the analysis.

[3]The independence theory of the LMS algorithm may be traced back to Widrow et al. (1976) and Mazo (1979).

Convergence analysis of the LMS algorithm remains an active area of research, which is motivated by a desire to relax the independence assumption. For a detailed discussion of a realistic model to prove convergence of the LMS algorithm, see Chapter 4 of the book by Macchi (1995); the model described therein assumes *no* independence between successive input vectors.

The independence assumption may be justified in certain applications such as adaptive beamforming, where it is possible for successive snapshots of data (i.e., input vectors) received by an array of antenna elements from the surrounding environment to be independent from each other. However, in adaptive filtering applications in communications (e.g., signal prediction, channel equalization, and echo cancelation), the sequence of input vectors that direct the "hunting" of the weight vector toward the optimum Wiener solution are in fact statistically dependent. This dependence arises because of the *shifting property* of the input data. Specifically, the tap-input vector at time n is

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

At time $n + 1$, it takes on the new value

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

Thus, with the arrival of the new sample $u(n + 1)$, the oldest sample $u(n - M + 1)$ is discarded from $\mathbf{u}(n)$, and the remaining samples $u(n), u(n - 1) \cdots, u(n - M + 2)$ are shifted back in time by one time unit to make room for the new sample. We see therefore that in a temporal setting the tap-input vectors, and correspondingly the gradient directions computed by the LMS algorithm, are indeed *statistically dependent.*

The independence theory *ignores* the statistical dependence among successive tap-input vectors at certain points in the development of the theory. For example, to evaluate the expectation of the term $\mathbf{u}(n)\mathbf{u}^H(n)\boldsymbol{\epsilon}(n)\boldsymbol{\epsilon}^H(n)$, it is assumed that the weight-error vector $\boldsymbol{\epsilon}(n)$ and the tap-input vector $\mathbf{u}(n)$ are statistically independent (even though, in reality, they may not be), and so this expectation is treated as the product of two expectations:

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

Ironically, when it comes to evaluating the expectation $E[\mathbf{u}(n)\mathbf{u}^H(n)]$, the correlation structure of the source producing $\mathbf{u}(n)$ is fully preserved, likewise for $E[\boldsymbol{\epsilon}(n)\boldsymbol{\epsilon}^H(n)]$. In so doing, the independence theory retains sufficient information about the structure of the adaptive process for the results of the theory to serve as reliable design guidelines.

Convergence Criteria

On the basis of Eq. (9.56), we could go on to establish the condition necessary for the *convergence of the mean*; that is,

$$E[\boldsymbol{\epsilon}(n)] \rightarrow \mathbf{0} \qquad \text{as } n \rightarrow \infty$$

or, equivalently,

$$E[\hat{\mathbf{w}}(n)] \rightarrow \mathbf{w}_o \qquad \text{as } n \rightarrow \infty \tag{9.57}$$

where $\mathbf{w}_o$ is the Wiener solution (see Problem 4 for details). Unfortunately, such a convergence criterion is of little practical value, since a sequence of zero-mean, but otherwise arbitrary, random variables converges in this sense. A stronger criterion is *convergence in the mean*, which is described by

$$E[\|\boldsymbol{\epsilon}(n)\|] \rightarrow 0 \qquad \text{as } n \rightarrow \infty$$

where $\|\boldsymbol{\epsilon}(n)\|$ is the Euclidean norm of the weight-error vector $\boldsymbol{\epsilon}(n)$. However, a proof of convergence in the mean is rather tedious, because $\|\boldsymbol{\epsilon}(n)\|$ is singular at the origin (Macchi, 1995).

To avoid the shortcomings of the two criteria, convergence of the mean and convergence in the mean, we may consider *convergence in the mean square*. Specifically, we say that the LMS algorithm is convergent in the mean square if

$$\mathfrak{D}(n) = E[\|\boldsymbol{\epsilon}(n)\|^2] \rightarrow \text{constant} \qquad \text{as } n \rightarrow \infty \tag{9.58}$$

where the scalar quantity $\mathfrak{D}(n)$ is called the *squared error deviation*. Another way of describing convergence of the LMS algorithm in the mean square is to require that

$$J(n) = E[|e(n)|^2] \rightarrow \text{constant} \qquad \text{as } n \rightarrow \infty \tag{9.59}$$

where $e(n)$ is the estimation error and $J(n)$ is the mean-squared error. Later in the section, we will show that

$$\lambda_{\min}\, \mathfrak{D}(n) \le J_{\text{ex}}(n) \le \lambda_{\max}\, \mathfrak{D}(n) \qquad \text{for all } n \tag{9.60}$$

where $\lambda_{\min}$ and $\lambda_{\max}$ are, respectively, the minimum and maximum eigenvalues of the correlation matrix $\mathbf{R}$; and $J_{\text{ex}}(n)$ is the excess mean-squared error, that is, the difference between $J(n)$ and the minimum mean-squared error $J_{\min}$ produced by the optimum Wiener filter. Hence, in light of this two-fold inequality, we may state that the decays of $J_{\text{ex}}(n)$ and $\mathfrak{D}(n)$ for increasing n are mathematically equivalent (Macchi, 1995). Therefore, it suffices for us to focus our attention on the convergence criterion described in (9.59).

With this convergence criterion for the LMS algorithm in mind, we propose to proceed as follows:

1. We use the stochastic difference equation (9.56) to derive a recursive equation for computing the correlation matrix of the weight-error vector $\boldsymbol{\epsilon}(n)$.
2. We then go on to derive an expression for the excess mean-squared error $J_{\text{ex}}(n)$.

These derivations are presented in the next two subsections, respectively.

Weight-Error Correlation Matrix

By definition, the correlation matrix of the weight-error vector $\boldsymbol{\epsilon}(n)$ is

$$\mathbf{K}(n) = E[\boldsymbol{\epsilon}(n)\boldsymbol{\epsilon}^H(n)] \tag{9.61}$$

Hence, applying this definition to the stochastic difference equation (9.56) and then invoking the independence assumption, we get

$$\mathbf{K}(n+1) = (\mathbf{I} - \mu\mathbf{R})\mathbf{K}(n)(\mathbf{I} - \mu\mathbf{R}) + \mu^2 J_{\min}\mathbf{R} \tag{9.62}$$

Equation (9.62) may be justified as follows:

- The first term, $(\mathbf{I} - \mu\mathbf{R})\mathbf{K}(n)(\mathbf{I} - \mu\mathbf{R})$, is the result of evaluating the expectation of the outer product of $(\mathbf{I} - \mu\mathbf{R})\boldsymbol{\epsilon}(n)$ with itself.
- The expectation of the cross-product term, $\mu e_o(n)(\mathbf{I} - \mu\mathbf{R})\boldsymbol{\epsilon}(n)\mathbf{u}^H(n)$, is zero by virtue of the implied independence of $\boldsymbol{\epsilon}(n)$ and $\mathbf{u}(n)$.
- The last term, $\mu^2 J_{\min}\mathbf{R}$, is obtained by applying the Gaussian factorization theorem to the product term $\mu^2 e_o^*(n)\mathbf{u}(n)\mathbf{u}^H(n)e_o(n)$; for details, see Problem 5.

The last term, $\mu^2 J_{\min}\mathbf{R}$, on the right-hand side of Eq. (9.62) prevents $\mathbf{K}(n) = \mathbf{O}$ from being a solution to this equation. Accordingly, the correlation matrix $\mathbf{K}(n)$ is prevented from going to zero by this small forcing term. In particular, the weight-error vector $\boldsymbol{\epsilon}(n)$ only approaches zero, but then executes small fluctuations about zero.[4] This formally confirms the point we made earlier under Example 5 that the convergent weight vector of the LMS algorithm may be modeled as shown in Fig. 9.10.

Since the correlation matrix $\mathbf{R}$ is positive definite, and μ is small, it follows that the first term of the expression on the right-hand side of Eq. (9.62) is also positive definite, provided that $\mathbf{K}(n)$ is positive definite. Clearly, the last term of this expression is always positive definite. It follows therefore that $\mathbf{K}(n+1)$ is positive definite, provided that $\mathbf{K}(n)$ is. The proof by induction is completed by noting that $\mathbf{K}(0)$ is positive definite, where

$$\begin{aligned}\mathbf{K}(0) &= \boldsymbol{\epsilon}(0)\boldsymbol{\epsilon}^H(0)\\ &= [\hat{\mathbf{w}}(0) - \mathbf{w}_o][\hat{\mathbf{w}}^H(0) - \mathbf{w}_o^H]\end{aligned}$$

In summary, Eq. (9.62) represents a recursive relationship for updating the weight-error correlation matrix $\mathbf{K}(n)$, starting with $n = 0$, for which we have $\mathbf{K}(0)$. Furthermore, after each iteration it does yield a positive-definite answer for the updated value of the weight-error correlation matrix.

Excess Mean-Squared Error

The matrix difference equation(9.62) provides a useful tool for determining the transient behavior of the mean-square error of the LMS algorithm, based on the independence

[4]In Bucklew et al., (1993), it is shown that the fluctuations of the weight-error vector $\boldsymbol{\epsilon}(n)$ about zero are asymptotically Gaussian in distribution. This asymptotic distribution is the result of two notions:

- a form of the central limit theorem
- assumed ergodicity of the input and disturbances in the LMS algorithm.

assumption. Specifically, we may proceed as follows. First, we use Eqs. (9.6) and (9.7) to express the estimation error, $e(n)$, produced by the LMS algorithm as

$$\begin{aligned} e(n) &= d(n) - \hat{\mathbf{w}}^H(n)\mathbf{u}(n) \\ &= d(n) - \mathbf{w}_o^H\mathbf{u}(n) - \boldsymbol{\epsilon}^H(n)\mathbf{u}(n) \\ &= e_o(n) - \boldsymbol{\epsilon}^H(n)\mathbf{u}(n) \end{aligned} \tag{9.63}$$

where $e_o(n)$ is the estimation error in the optimum Wiener solution, and $\boldsymbol{\epsilon}(n)$ is the tap-weight error vector. Let $J(n)$ denote the mean-squared error due to the LMS algorithm at iteration n. Hence, using Eq. (9.63) to evaluate $J(n)$ and then invoking the independence assumption, we get

$$\begin{aligned} J(n) &= E[|e(n)|^2] \\ &= E[(e_o(n) - \boldsymbol{\epsilon}^H(n)\mathbf{u}(n))(e_o^*(n) - \mathbf{u}^H(n)\boldsymbol{\epsilon}(n))] \\ &= J_{\min} + E[\boldsymbol{\epsilon}^H(n)\mathbf{u}(n)\mathbf{u}^H(n)\boldsymbol{\epsilon}(n)] \end{aligned} \tag{9.64}$$

where $J_{\min}$ is the minimum mean-squared error produced by the optimum Wiener filter.

Our next task is to evaluate the expectation term in the final line of Eq. (9.64). Here we note that this term is the expected value of a scalar random variable represented by a triple vector product; and the trace of a scalar is the scalar itself. We may therefore rewrite it as[5]

$$\begin{aligned} E[\boldsymbol{\epsilon}^H(n)\mathbf{u}(n)\mathbf{u}^H(n)\boldsymbol{\epsilon}(n)] &= E[\mathrm{tr}\{\boldsymbol{\epsilon}^H(n)\mathbf{u}(n)\mathbf{u}^H(n)\boldsymbol{\epsilon}(n)\}] \\ &= E[\mathrm{tr}\{\mathbf{u}(n)\mathbf{u}^H(n)\boldsymbol{\epsilon}(n)\boldsymbol{\epsilon}^H(n)\}] \\ &= \mathrm{tr}\{E[\mathbf{u}(n)\mathbf{u}^H(n)\boldsymbol{\epsilon}(n)\boldsymbol{\epsilon}^H(n)]\} \end{aligned}$$

Invoking the independence assumption again, we may reduce this expectation to

$$\begin{aligned} E[\boldsymbol{\epsilon}^H(n)\mathbf{u}(n)\mathbf{u}^H(n)\boldsymbol{\epsilon}(n)] &= \mathrm{tr}\{E[\mathbf{u}(n)\mathbf{u}^H(n)]E[\boldsymbol{\epsilon}(n)\boldsymbol{\epsilon}^H(n)]\} \\ &= \mathrm{tr}[\mathbf{R}\mathbf{K}(n)] \end{aligned} \tag{9.65}$$

where $\mathbf{R}$ is the correlation matrix of the tap inputs and $\mathbf{K}(n)$ is the weight-error correlation matrix.

[5]Let $\mathbf{A}$ and $\mathbf{B}$ denote a pair of matrices of compa ble dimensions. The trace of matrix product $\mathbf{AB}$ equals the trace of matrix product $\mathbf{BA}$; that is,

$$\mathrm{tr}[\mathbf{AB}] = \mathrm{tr}[\mathbf{BA}]$$

For the problem at hand, we have

$$\begin{aligned} \mathbf{A} &= \boldsymbol{\epsilon}^H(n) \\ \mathbf{B} &= \mathbf{u}(n)\mathbf{u}^H(n)\boldsymbol{\epsilon}(n) \end{aligned}$$

Therefore,

$$\mathrm{tr}[\boldsymbol{\epsilon}^H(n)\mathbf{u}(n)\mathbf{u}^H(n)\boldsymbol{\epsilon}(n)] = \mathrm{tr}[\mathbf{u}(n)\mathbf{u}^H(n)\boldsymbol{\epsilon}(n)\boldsymbol{\epsilon}^H(n)]$$

Accordingly, using Eq. (9.65) in Eq. (9.64), we may rewrite the expression for the mean-squared error in the LMS algorithm simply as

$$J(n) = J_{\min} + \text{tr}[\mathbf{RK}(n)] \tag{9.66}$$

Equation (9.66) indicates that for all n, the mean-square value of the estimation error in the LMS algorithm consists of two components: the minimum mean-squared error $J_{\min}$, and a component depending on the transient behavior of the weight-error correlation matrix $\mathbf{K}(n)$. Since the latter component is positive definite for all n, *the LMS algorithm always produces a mean-squared error* $J(n)$ *that is in excess of the minimum mean-squared error* $J_{\min}$.

We now formally define the *excess mean-squared error* as the difference between the mean-squared error, $J(n)$, produced by the adaptive algorithm at time n and the minimum value, $J_{\min}$, pertaining to the optimum Wiener solution. Denoting the excess mean-squared error by $J_{ex}(n)$, we have

$$\begin{aligned} J_{ex}(n) &= J(n) - J_{\min} \\ &= \text{tr}[\mathbf{RK}(n)] \end{aligned} \tag{9.67}$$

For $\mathbf{K}(n)$ we use the recursive relation of Eq. (9.62). However, when the mean-squared error is of primary interest, another form of this equation obtained by a simple rotation of coordinates is more useful. The particular rotation of coordinates we have in mind is described by the unitary similarity transformation of Eq. (4.30), reproduced here for convenience:

$$\mathbf{Q}^H\mathbf{RQ} = \mathbf{\Lambda} \tag{9.68}$$

where $\mathbf{\Lambda}$ is a diagonal matrix consisting of the eigenvalues of the correlation matrix $\mathbf{R}$, and $\mathbf{Q}$ is the unitary matrix consisting of the eigenvectors associated with these eigenvalues. Note that the matrix $\mathbf{\Lambda}$ is real valued. Furthermore, let

$$\mathbf{Q}^H\mathbf{K}(n)\mathbf{Q} = \mathbf{X}(n) \tag{9.69}$$

In general, $\mathbf{X}(n)$ is not a diagonal matrix. Using Eqs. (9.68) and (9.69), we get

$$\begin{aligned} \text{tr}[\mathbf{RK}(n)] &= \text{tr}[\mathbf{Q\Lambda Q}^H\mathbf{QX}(n)\mathbf{Q}^H] \\ &= \text{tr}[\mathbf{Q\Lambda X}(n)\mathbf{Q}^H] \\ &= \text{tr}[\mathbf{Q}^H\mathbf{Q\Lambda X}(n)] \\ &= \text{tr}[\mathbf{\Lambda X}(n)] \end{aligned} \tag{9.70}$$

where, in the third line, we used the matrix property described in footnote 5, and in the second and last lines we used the property $\mathbf{Q}^H\mathbf{Q} = \mathbf{I}$. Accordingly, we have

$$J_{ex}(n) = \text{tr}[\mathbf{\Lambda X}(n)] \tag{9.71}$$

Since $\mathbf{\Lambda}$ is a diagonal matrix, we may also write[6]

$$J_{\text{ex}}(n) = \sum_{i=1}^{M} \lambda_i x_i(n) \tag{9.72}$$

where the $x_i(n)$, $i = 1, 2, \ldots, M$, are the diagonal elements of the matrix $\mathbf{X}(n)$, and λ_i are the eigenvalues of the correlation matrix $\mathbf{R}$.

Next, using the transformations described by Eqs. (9.68) and (9.69), we may rewrite the recursive equation (9.62) in terms of $\mathbf{X}(n)$ and $\mathbf{\Lambda}$ as follows:

$$\mathbf{X}(n+1) = (\mathbf{I} - \mu\mathbf{\Lambda})\mathbf{X}(n)(\mathbf{I} - \mu\mathbf{\Lambda}) + \mu^2 J_{\text{min}}\mathbf{\Lambda} \tag{9.73}$$

We observe from Eq. (9.72) that $J_{\text{ex}}(n)$ depends on the $x_i(n)$. This suggests that we need only look at the diagonal terms of the recursive equation (9.73). Because of the form of this equation, the x_i decouple from the off-diagonal terms, and so we have

$$x_i(n) = (1 - \mu\lambda_i)^2 x_i(n) + \mu^2 J_{\text{min}}\lambda_i, \qquad i = 1, 2, \ldots, M \tag{9.74}$$

Define the M-by-1 vectors $\mathbf{x}(n)$ and $\boldsymbol{\lambda}$ as follows, respectively:

$$\mathbf{x}(n) = [x_1(n), x_2(n), \ldots, x_M(n)]^T$$

$$\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \ldots, \lambda_M]^T$$

Then we may rewrite Eq. (9.74) in matrix form as

$$\mathbf{x}(n+1) = \mathbf{B}\mathbf{x}(n) + \mu^2 J_{\text{min}}\boldsymbol{\lambda} \tag{9.75}$$

where $\mathbf{B}$ is an M-by-M matrix with elements

$$b_{ij} = \begin{cases} (1 - \mu\lambda_i)^2, & i = j \\ \mu^2\lambda_i\lambda_j, & i \neq j \end{cases} \tag{9.76}$$

From Eq. (9.76), we readily see that the matrix $\mathbf{B}$ is *real, positive*, and *symmetric*.

[6]The two-fold inequality (9.60), referred to earlier under the subsection on convergence criteria, may be derived from Eq. (9.72) as follows. Starting with the definition of matrix $\mathbf{X}(n)$ given in Eq. (9.69), we note that

$$\mathbf{X}(n) = \mathbf{Q}^H E[\boldsymbol{\epsilon}(n)\boldsymbol{\epsilon}^H(n)]\mathbf{Q} = E[\mathbf{Q}^H\boldsymbol{\epsilon}(n)\boldsymbol{\epsilon}^H(n)\mathbf{Q}]$$

Let $\boldsymbol{\omega}(n) = \mathbf{Q}^H\boldsymbol{\epsilon}(n)$. Then, noting that $\mathbf{Q}$ is a unitary matrix, we may express the ith diagonal element of matrix $\mathbf{X}(n)$ as

$$x_i(n) = E[|\omega_i(n)|^2] = E[|\epsilon_i(n)|^2] \quad \text{for all } i$$

where $\omega_i(n)$ and $\epsilon_i(n)$ are the ith elements of $\boldsymbol{\omega}(n)$ and $\boldsymbol{\epsilon}(n)$, respectively. Accordingly, using Eq. (9.72), we may write

$$J_{\text{ex}}(n) = \sum_{i=1}^{M} \lambda_i E[|\epsilon_i(n)|^2]$$

Let $\lambda_1 = \lambda_{max}$ and $\lambda_M = \lambda_{min}$. Since $\|\boldsymbol{\epsilon}(n)\|^2 = \sum_{i=1}^{M} |\epsilon_i(n)|^2$, we may bound $J_{\text{ex}}(n)$ as shown by

$$\lambda_{\text{min}} E[\|\boldsymbol{\epsilon}(n)\|^2] \leq J_{\text{ex}}(n) \leq \lambda_{\text{max}} E[\|\boldsymbol{\epsilon}(n)\|^2]$$

from which (9.60) follows immediately(Macchi, 1995).

In Appendix H it is shown that the solution to the difference equation (9.75) is given by

$$\mathbf{x}(n) = \sum_{i=1}^{M} c_i^n \mathbf{g}_i \mathbf{g}_i^T [\mathbf{x}(0) - \mathbf{x}(\infty)] + \mathbf{x}(\infty) \tag{9.77}$$

where the various terms are defined as follows:

- The coefficient c_i is the ith eigenvalue of matrix $\mathbf{B}$, and $\mathbf{g}_i$ is the associated eigenvector; that is,

$$\mathbf{G}^T \mathbf{B} \mathbf{G} = \mathbf{C} \tag{9.78}$$

  where

$$\mathbf{C} = \text{diag}[c_1, c_2, \ldots, c_M]$$
$$\mathbf{G} = [\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_M]$$

- The vector $\mathbf{x}(0)$ is the initial value of $\mathbf{x}(n)$, and $\mathbf{x}(\infty)$ is its final value.

The excess mean-squared error equals [see Eq. (9.72)]

$$\begin{aligned} J_{ex}(n) &= \boldsymbol{\lambda}^T \mathbf{x}(n) \\ &= \sum_{i=1}^{M} c_i^n \boldsymbol{\lambda}^T \mathbf{g}_i \mathbf{g}_i^T [\mathbf{x}(0) - \mathbf{x}(\infty)] + \boldsymbol{\lambda}^T \mathbf{x}(\infty) \\ &= \sum_{i=1}^{M} c_i^n \boldsymbol{\lambda}^T \mathbf{g}_i \mathbf{g}_i^T [\mathbf{x}(0) - \mathbf{x}(\infty)] + J_{ex}(\infty) \end{aligned} \tag{9.79}$$

where

$$\begin{aligned} J_{ex}(\infty) &= \boldsymbol{\lambda}^T \mathbf{x}(\infty) \\ &= \sum_{j=1}^{M} \lambda_j x_j(\infty) \end{aligned} \tag{9.80}$$

In Eq. (9.79), the first term on the right-hand side describes the transient behavior of the mean-squared error, whereas the second term represents the final value of the excess mean-squared error after adaptation is completed (i.e., its steady-state value).

Transient Behavior of the Mean-squared Error

Using Eq. (9.79) and the first line of Eq. (9.67), we may express the time evolution of the mean-squared error for the LMS algorithm by the equation

$$J(n) = \sum_{i=1}^{M} \gamma_i c_i^n + J_{\min} + J_{ex}(\infty) \tag{9.81}$$

where c_i is the ith eigenvalue of matrix $\mathbf{B}$, and γ_i is defined by

$$\gamma_i = \boldsymbol{\lambda}^T \mathbf{g}_i \mathbf{g}_i^T [\mathbf{x}(0) - \mathbf{x}(\infty)], \qquad i = 1, 2, \ldots, M \tag{9.82}$$

Equation (9.81) provides the basis for a deeper understanding of the operation of the LMS algorithm in a wide-sense stationary environment, as described next in the form of four properties.

Property 1. *The transient component of the mean-squared error, $J(n)$, does not exhibit oscillations.*

The transient component of $J(n)$ equals

$$\sum_{i=1}^{M} \gamma_i c_i^n$$

where the γ_i are constant coefficients and the c_i, $i = 1, 2, \ldots, M$, are the eigenvalues of matrix $\mathbf{B}$. These eigenvalues are all real positive numbers, since $\mathbf{B}$ is a real symmetric positive-definite matrix [see Eq. (9.76)]. Hence, the ensemble-averaged learning curve, that is, a plot of the mean-squared error $J(n)$ versus the number of iterations, n, consists only of exponentials.

Note, however, that the learning curve represented by a plot of the squared error $|e(n)|^2$, without ensemble averaging, versus n consists of *noisy* exponentials. The amplitude of the noise becomes smaller as the step-size parameter μ is reduced.

Property 2. *The transient component of the mean-squared error $J(n)$ dies out; that is, the LMS algorithm is convergent in the mean square if and only if the step-size parameter μ satisfies the condition*

$$0 < \mu < \frac{2}{\lambda_{\max}} \tag{9.83}$$

where $\lambda_{\max}$ is the largest eigenvalue of the correlation matrix $\mathbf{R}$.

For this property to hold, all the eigenvalues of matrix $\mathbf{B}$ must be less than 1 in magnitude. Let $\mathbf{g}$ be an eigenvector of matrix $\mathbf{B}$, associated with eigenvalue c. Then, by definition, we have

$$\mathbf{Bg} = c\mathbf{g}$$

Equivalently, we may write

$$\sum_{j=1}^{M} b_{ij} g_j = c g_i, \quad i = 1, 2, \ldots, M \tag{9.84}$$

where the g_i are the elements of the eigenvector $\mathbf{g}$. Using Eq. (9.76) for the elements of matrix $\mathbf{B}$ in Eq. (9.84), we get

$$(1 - \mu\lambda_i)^2 g_i + \mu^2 \lambda_i \sum_{\substack{j=1 \\ j \neq i}}^{M} \lambda_j g_j = c g_i, \quad i = 1, 2, \ldots, M \tag{9.85}$$

Solving Eq. (9.85) for g_i, we may thus write

$$g_i = \frac{\mu^2\lambda_i}{c - (1 - \mu\lambda_i)^2} \sum_{\substack{j=1 \\ j\neq i}}^{M} \lambda_j g_j, \quad i = 1, 2, \ldots, M \tag{9.86}$$

Next, we acknowledge the fact that the square matrix **B** is a *positive matrix* since all of its elements are positive. This means that we may use *Perron's theorem*,[7] which applies to a positive square matrix. Perron's theorem states that (Bellman, 1960):

If **B** is a positive square matrix, there is a unique eigenvalue of **B**, which has the largest magnitude. This eigenvalue is positive and simple (i.e., of multiplicity 1), and its associated eigenvector consists entirely of positive elements.

Accordingly, we may associate a *positive eigenvector* (i.e., a vector consisting entirely of positive elements) with the special eigenvalue of matrix **B** that has the largest magnitude. Thus setting the eigenvalue c equal to 1 in Eq. (9.86), and then simplifying, we get

$$g_i = \frac{\mu}{2 - \mu\lambda_i} \sum_{\substack{j=1 \\ j\neq i}}^{M} \lambda_i g_j, \quad i = 1, 2, \ldots, M \tag{9.87}$$

from which we readily see that for g_i to be positive for all i, the step-size parameter μ must be upper bounded as in (9.83).

Property 3. *The final value of the excess mean-squared error is less than the minimum mean-squared error if the step-size parameter μ satisfies the condition*

$$\sum_{i=1}^{M} \frac{2\lambda_i}{2 - \mu\lambda_i} < 1 \tag{9.88}$$

where the λ_i, $i = 1, 2, \ldots, M$, are the eigenvalues of the correlation matrix **R**.

Given that the LMS algorithm is convergent in the mean square and therefore Property 2 holds, we find that as the number of iterations approaches infinity:

$$J_{ex}(\infty) = J(\infty) - J_{min}$$

To find the final value of the excess mean-squared error, $J_{ex}(\infty)$, we may go back to Eq. (9.74). In particular, setting $n = \infty$ and then solving the resulting equation for $x_i(\infty)$, we get

$$x_i(\infty) = \frac{\mu J_{min}}{2 - \mu\lambda_i}, \quad i = 1, 2, \ldots, M \tag{9.89}$$

[7]Perron's theorem is also known as the *Perron–Frobenius theorem.*

Hence, evaluating Eq. (9.72) for $n = \infty$ and then substituting Eq. (9.89) in the resultant, we get

$$J_{ex}(\infty) = \sum_{i=1}^{M} \lambda_i x_i(\infty)$$

$$= J_{min} \sum_{i=1}^{M} \frac{\mu\lambda_i}{2 - \mu\lambda_i} \tag{9.90}$$

From this equation we readily see that $J_{ex}(\infty)$ is indeed less than J_{min}, provided that the step-size parameter μ satisfies the condition described in Eq. (9.88).

Property 4. *The misadjustment, defined as the ratio of the steady-state value $J_{ex}(\infty)$ of the excess mean-squared error to the minimum mean-squared error J_{min}, equals*

$$\mathcal{M} = \frac{J_{ex}(\infty)}{J_{min}}$$

$$= \sum_{i=1}^{M} \frac{\mu\lambda_i}{2 - \mu\lambda_i} \tag{9.91}$$

which is less than unity if the step-size parameter μ satisfies the condition of Eq. (9.88).

The formula for $\mathcal{M}$ given in Eq. (9.91) follows immediately from (9.90).

The misadjustment $\mathcal{M}$ is a dimensionless quantity, providing a measure of how close the LMS algorithm is to optimality in the mean-square error sense. The smaller $\mathcal{M}$ is compared to unity, the more *accurate* is the adaptive filtering action being performed by the LMS algorithm. It is customary to express the misadjustment $\mathcal{M}$ as a percentage. Thus, for example, a misadjustment of 10 percent means that the LMS algorithm produces a mean-squared error (after adaptation is completed) that is 10 percent greater than the minimum mean-squared error J_{min}. Such performance is ordinarily considered to be satisfactory in practice.

Simple Working Rules

In this rather long section we have presented a theoretical (albeit approximate) analysis of the stability and mean-squared error performance of the LMS algorithm when operating in a wide-sense stationary environment. The analysis has been based on (1) the direct-averaging method, assuming that the step-size parameter μ is assigned a small value and (2) the independence assumption. Notwithstanding these assumptions, it appears that in practice the theory presented herein holds for a reasonably wide range of values of μ (Widrow et al., 1976). In any event, given the lengthy material presented in this section, it is befitting that we finish the discussion with some helpful rules for the LMS design of adaptive filters.

The condition for the LMS algorithm to be convergent in the mean square, described in Eq. (9.83), requires knowledge of the largest eigenvalue λ_{max} of the correlation matrix $\mathbf{R}$. In typical applications of the LMS algorithm, knowledge of λ_{max} is not available. To overcome this practical difficulty, the trace of **R** may be taken as a conservative estimate for λ_{max}, in which case the condition of (9.83) may be reformulated as

$$0 < \mu < \frac{2}{\text{tr}[\mathbf{R}]} \tag{9.92}$$

where tr[**R**] denotes the trace of matrix **R**. We may go one step further by noting that the correlation matrix **R** is not only positive definite but also Toeplitz with all of the elements on its main diagonal equal to $r(0)$. Since $r(0)$ is itself equal to the mean-square value of the input at each of the M taps in the transversal filter, we have

$$\begin{aligned}\text{tr}[\mathbf{R}] &= M\, r(0) \\ &= \sum_{k=0}^{M-1} E[\,|u(n-k)|^2] \end{aligned} \tag{9.93}$$

Thus, using the term "tap-input power" to refer to the sum of the mean-square values of tap inputs $u(n), u(n-1), \ldots, u(n-M+1)$ in Fig. 9.1(b), we may restate the condition of Eq. (9.92) for convergence of the LMS algorithm in the mean square as

$$0 < \mu < \frac{2}{\text{tap-input power}} \tag{9.94}$$

Another formula that needs to be revisited is that of Eq. (9.91), pertaining to the misadjustment $\mathcal{M}$. In its present form, this formula is impractical for it requires knowledge of all the eigenvalues of the correlation matrix **R**. However, assuming that the step-size parameter μ is small compared to the largest eigenvalue λ_{max}, we may approximate Eq. (9.91) as follows:

$$\begin{aligned}\mathcal{M} &\approx \frac{\mu}{2}\sum_{i=1}^{M}\lambda_i \\ &= \frac{\mu}{2}\,(\text{tap-input power})\end{aligned} \tag{9.95}$$

Thus, the practical condition of (9.94) imposed on the step-size parameter μ not only assures the convergence of the LMS algorithm in the mean square, but also results in a misadjustment $\mathcal{M}$ that is less than unity, both of which are desirable goals in their own individual ways.

Define an *average eigenvalue* for the underlying correlation matrix **R** of the tap inputs as

$$\lambda_{av} = \frac{1}{M}\sum_{i=1}^{M}\lambda_i \tag{9.96}$$

Suppose also that the ensemble-averaged learning curve of the LMS algorithm is approximated by a single exponential with time constant $(\tau)_{mse,av}$. We may then use Eq. (8.31), developed for the method of steepest descent, to define the following *average time constant* for the LMS algorithm:

$$(\tau)_{mse,av} \approx \frac{1}{2\mu\lambda_{av}} \tag{9.97}$$

Hence, using the average values of Eqs. (9.96) and (9.97) in (9.95), we may redefine the misadjustment approximately as follows (Widrow and Stearns, 1985):

$$\mathcal{M} \approx \frac{\mu M \lambda_{av}}{2} \simeq \frac{M}{4\tau_{mse,av}} \tag{9.98}$$

On the basis of this formula, we may now make the following observations:

1. The misadjustment $\mathcal{M}$ increases linearly with the filter length (number of taps) denoted by M, for a fixed $\tau_{mse,av}$.
2. The *settling time* of the LMS algorithm (i.e., the time taken for the transients to die out) is proportional to the average time constant $\tau_{mse,av}$. It follows therefore that the misadjustment $\mathcal{M}$ is inversely proportional to the settling time.
3. The misadjustment $\mathcal{M}$ is directly proportional to the step-size parameter μ, whereas the average time constant $\tau_{mse,av}$ is inversely proportional to μ. We therefore have conflicting requirements in that if μ is reduced so as to reduce the misadjustment, then the settling time of the LMS algorithm is increased. Conversely, if μ is increased so as to reduce the settling time, then the misadjustment is increased. Careful attention has therefore to be given to the choice of μ. (In Chapter 16 we will evaluate the additional constraints on the choice of μ when the environment in which the adaptive filter operates is time varying.)

Comparison of the LMS Algorithm with the Method of Steepest Descent

At this point in the discussion, it is informative for us to look at the LMS algorithm for stochastic inputs in light of what we know about the steepest-descent algorithm that we studied in Chapter 8.

Ideally, the minimum mean-squared error J_{min} is realized when the coefficient vector $\hat{\mathbf{w}}(n)$ of the transversal filter approaches the optimum value $\mathbf{w}_o$, defined by the Wiener–Hopf equations. Indeed, as shown in Section 8.5, the steepest-descent algorithm does realize this idealized condition as the number of iterations, n, approaches infinity. The steepest-descent algorithm has the capability to do this, because it uses *exact* measurements of the gradient vector at each iteration of the algorithm. On the other hand, the LMS algorithm relies on a *noisy* estimate for the gradient vector, with the result that the tap-weight vector estimate $\hat{\mathbf{w}}(n)$ only approaches the optimum value $\mathbf{w}_o$. Consequently, use of the LMS algorithm, after a large number of iterations, results in a mean-squared error $J(\infty)$ that is greater than the minimum mean-squared error J_{min}. The amount by which the actual value of $J(\infty)$ is greater than J_{min} is the excess mean-squared error.

There is another basic difference between the steepest-descent algorithm and the LMS algorithm. In Section 8.5, we showed that the steepest-descent algorithm has a well-defined learning curve, obtained by plotting the mean-squared error versus the number of iterations. For this algorithm, the learning curve consists of a sum of decaying exponentials, the number of which equals (in general) the number of tap coefficients. On the other hand, in individual applications of the LMS algorithm, we find that the learning curve consists of *noisy, decaying exponentials*. The amplitude of the noise usually becomes smaller as the step-size parameter μ is reduced.

Imagine now an *ensemble of adaptive transversal filters*. Each filter is assumed to use the LMS algorithm with the same step-size parameter μ and the same initial tap-weight vector $\hat{\mathbf{w}}(0)$. Also, each adaptive filter has individual stationary ergodic inputs that are selected at random from the same statistical population. We compute the *noisy learning curves* for this ensemble of adaptive filters by plotting the squared magnitude of the estimation error $e(n)$ versus the number of iterations n. To compute the *ensemble-averaged learning curve* of the LMS algorithm, that is, the plot of the mean-squared error $J(n)$ versus n, we take the average of these noisy learning curves over the ensemble of adaptive filters.

Thus two entirely different ensemble-averaging operations are used in the steepest-descent and LMS algorithms for determining their learning curves (i.e., plots of their mean-squared errors versus the learning period). In the steepest-descent algorithm, the correlation matrix $\mathbf{R}$ and the cross-correlation vector $\mathbf{p}$ are first computed through the use of ensemble-averaging operations applied to statistical populations of the tap inputs and the desired response; these values are then used to compute the learning curve of the algorithm. In the LMS algorithm, on the other hand, noisy learning curves are computed for an ensemble of adaptive LMS filters with identical parameters; the learning curve is then smoothed by averaging over the ensemble of noisy learning curves.

9.5 SUMMARY OF THE LMS ALGORITHM

Parameters: M = number of taps

μ = step-size parameter

$$0 < \mu < \frac{2}{\text{tap-input power}}$$

$$\text{tap-input power} = \sum_{k=0}^{M-1} E[|u(n-k)|^2]$$

Initialization: If prior knowledge on the tap-weight vector $\hat{\mathbf{w}}(n)$ is available, use it to select an appropriate value for $\hat{\mathbf{w}}(0)$. Otherwise, set $\hat{\mathbf{w}}(0) = \mathbf{0}$.

Data:

- Given: $\mathbf{u}(n)$ = M-by-1 tap-input vector at time n
  $d(n)$ = desired response at time n
- To be computed:
  $\hat{\mathbf{w}}(n+1)$ = estimate of tap-weight vector at time $n+1$

Computation: For $n = 0, 1, 2, \ldots$, compute

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

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \mu\mathbf{u}(n)e^*(n)$$

9.6 COMPUTER EXPERIMENT ON ADAPTIVE PREDICTION

For our first computer experiment involving the LMS algorithm, we use a first-order, autoregressive (AR) process to study the effects of ensemble averaging on the transient characteristics of the LMS algorithm for *real data*.

Consider then an AR process $u(n)$ of order 1, described by the difference equation

$$u(n) = -au(n-1) + v(n) \tag{9.99}$$

where a is the (one and only) parameter of the process, and $v(n)$ is a zero-mean white-noise process of variance σ_v^2. To estimate the parameter a, we may use an adaptive predictor of order 1, as depicted in Fig. 9.13. The *real* LMS algorithm for the adaptation of the (one and only) tap weight of the predictor is written as

$$\hat{w}(n+1) = \hat{w}(n) + \mu u(n-1)f(n) \tag{9.100}$$

where $f(n)$ is the prediction error, defined by

$$f(n) = u(n) - \hat{w}(n)u(n-1) \tag{9.101}$$

Figure 9.14 shows plots of $\hat{w}(n)$ versus the number of iterations n for a single trial of the experiment, and the following two sets of conditions:

1. AR parameter: $a = -0.99$
   Variance of AR process $u(n)$: $\sigma_u^2 = 0.93627$
2. AR parameter: $a = +0.99$
   Variance of AR process $u(n)$: $\sigma_u^2 = 0.995$

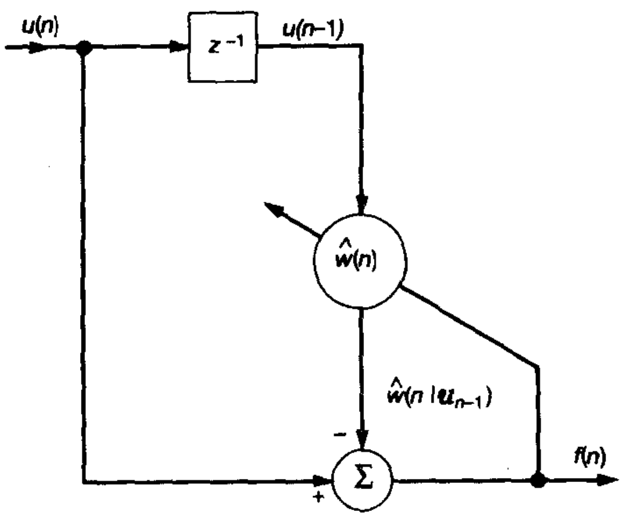


Figure 9.13 Adaptive first-order predictor.

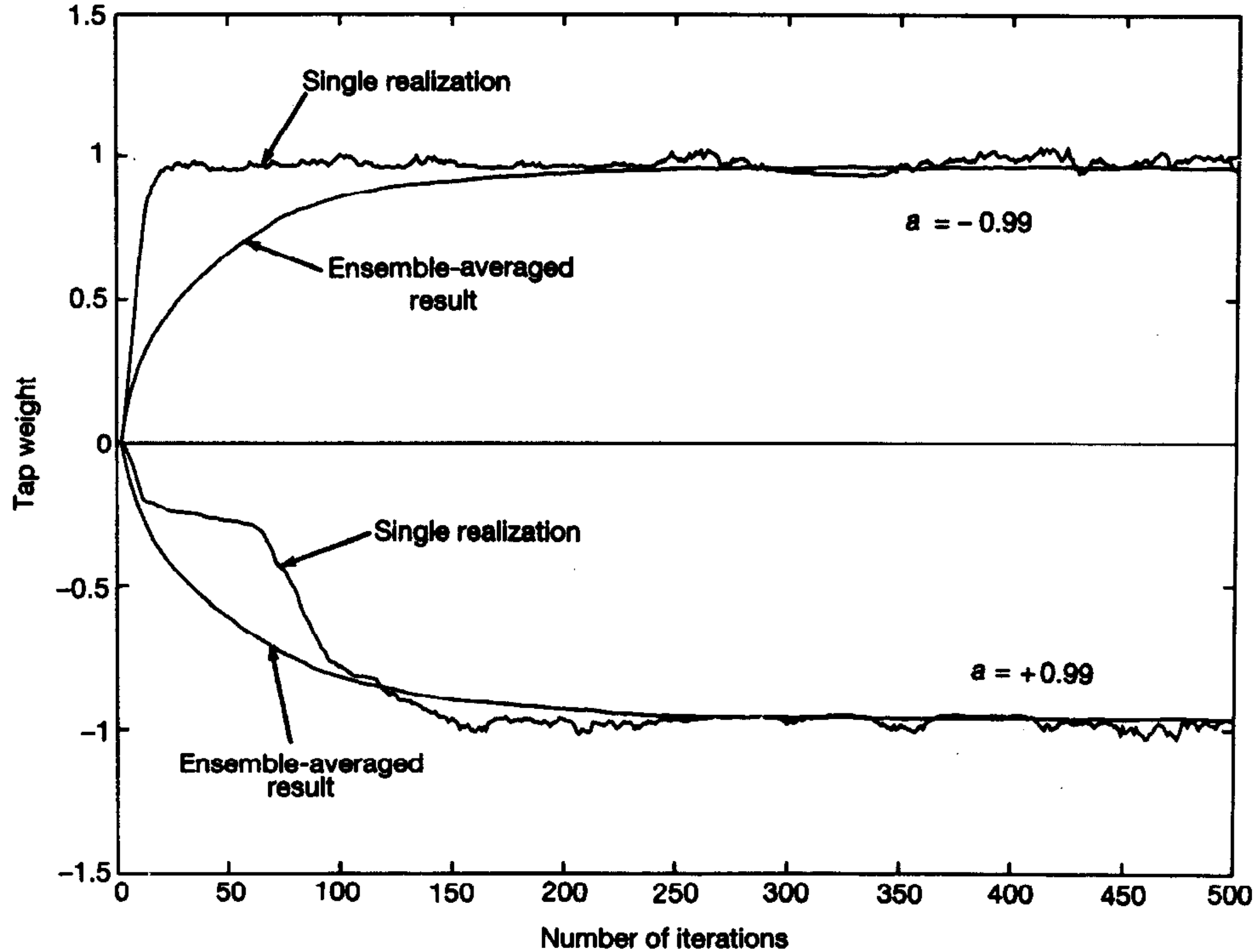


Figure 9.14 Transient behavior of weight $\hat{w}(n)$ of adaptive first-order predictor.

In both cases, the step-size parameter $\mu = 0.05$, and the initial condition is $\hat{w}(0) = 0$. We see that the transient behavior of $\hat{w}(n)$ follows a *noisy exponential curve*. Figure 9.14 also includes the corresponding plot of $E[\hat{w}(n)]$ obtained by ensemble averaging over 100 *independent trials* of the experiment. For each trial, a different computer realization of the AR process $u(n)$ is used. We see that the ensemble averaging has the effect of smoothing out the effects of gradient noise.

Figure 9.15 shows a plot of the squared prediction error $f^2(n)$ versus the number of iterations n for the set of AR parameters listed under (1) specified above; the step-size parameter $\mu = 0.05$, as before. We see that the learning curve for a single realization of the LMS algorithm exhibits a very noise form. Figure 9.15 also includes the corresponding plot of $E[f^2(n)]$ obtained by ensemble averaging over 100 independent trials of the experiment. The smoothing effect of the ensemble averaging operation on the learning curve of the LMS algorithm is again visible in the figure.

Figure 9.16 shows experimental plots of the learning curves of the LMS algorithm [i.e., the mean-squared error $J(n)$ versus the number of iterations n] for the set of AR parameters listed under (1) specified above and varying step-size parameter μ. Specifically, the values used for μ are 0.01, 0.05, and 0.1. The ensemble averaging was performed over 100 independent trials of the experiment. From Fig. 9.16, we observe the following:

- As the step-size parameter μ is reduced, the rate of convergence of the LMS algorithm is correspondingly decreased.
- A reduction in the step-size parameter μ also has the effect of reducing the variation in the experimentally computed learning curve.

Comparison of Experimental Results with Theory

In Fig. 9.17, we have plotted two pairs of curves for $\hat{w}(n)$ versus n, corresponding to the two different sets of parameter values listed under (1) and (2) above, and step-size parameter $\mu = 0.05$. One pair of curves corresponds to experimentally derived results obtained by ensemble-averaging over 100 independent trials of the experiment; these curves are labeled "Experiment" in Fig. 9.17. The other pair of curves is computed from theory. In particular, for the situation at hand, we have:

1. The autocorrelation function of the AR process $u(n)$ for zero lag is

$$r(0) = \sigma_u^2$$

2. The Wiener solution for the tap weight of the order-1 predictor is

$$w_o = -a$$

Taking the expectation of Eq. (9.100) and invoking the use of Eqs. (9.99) and (9.101), we find that in light of the independence assumption:

$$E[\hat{w}(n+1)] = (1 - \mu\sigma_u^2)E[\hat{w}(n)] - \mu a \sigma_u^2 \tag{9.102}$$

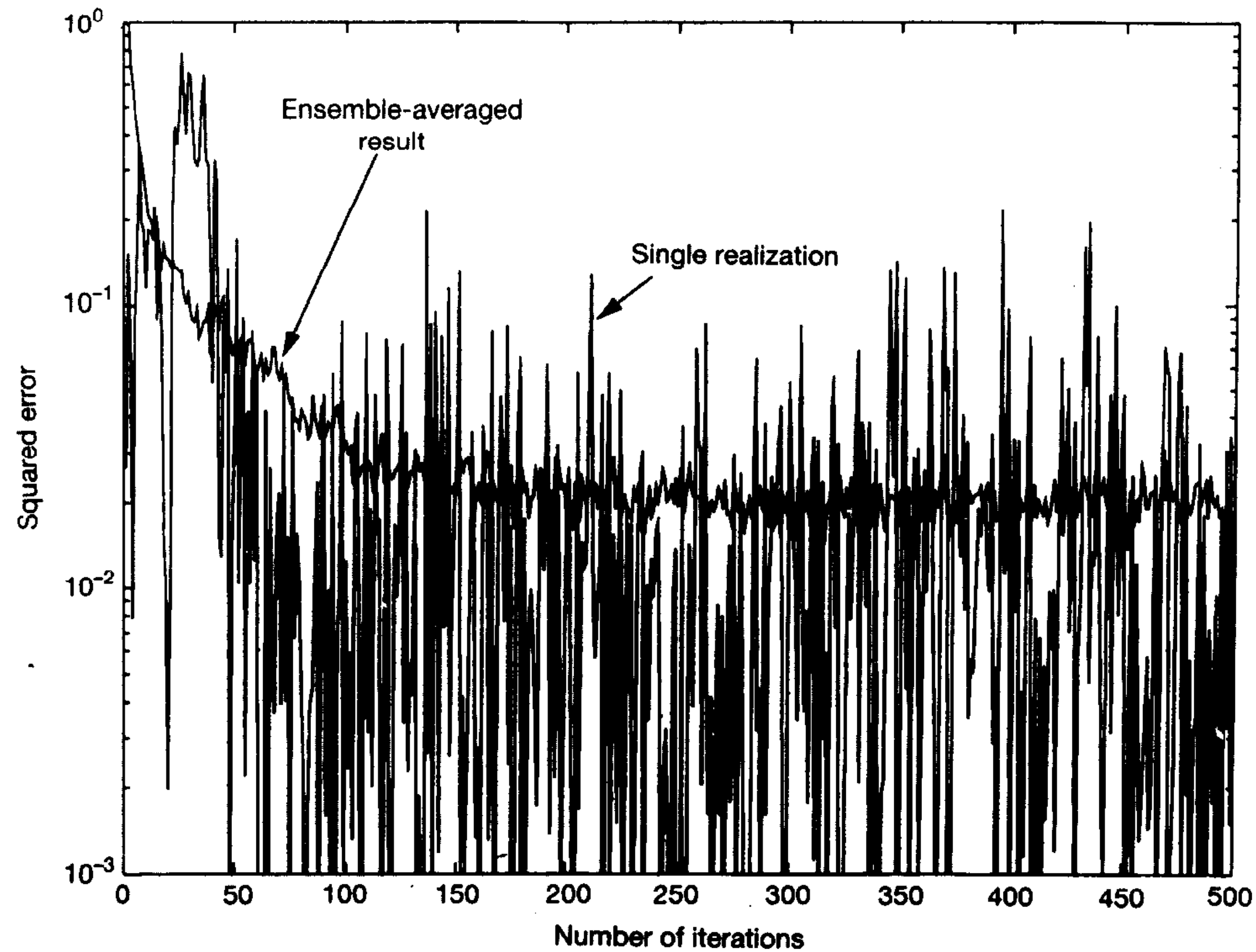


Figure 9.15 Transient behavior of squared prediction error in adaptive first-order for $\mu = 0.05$.

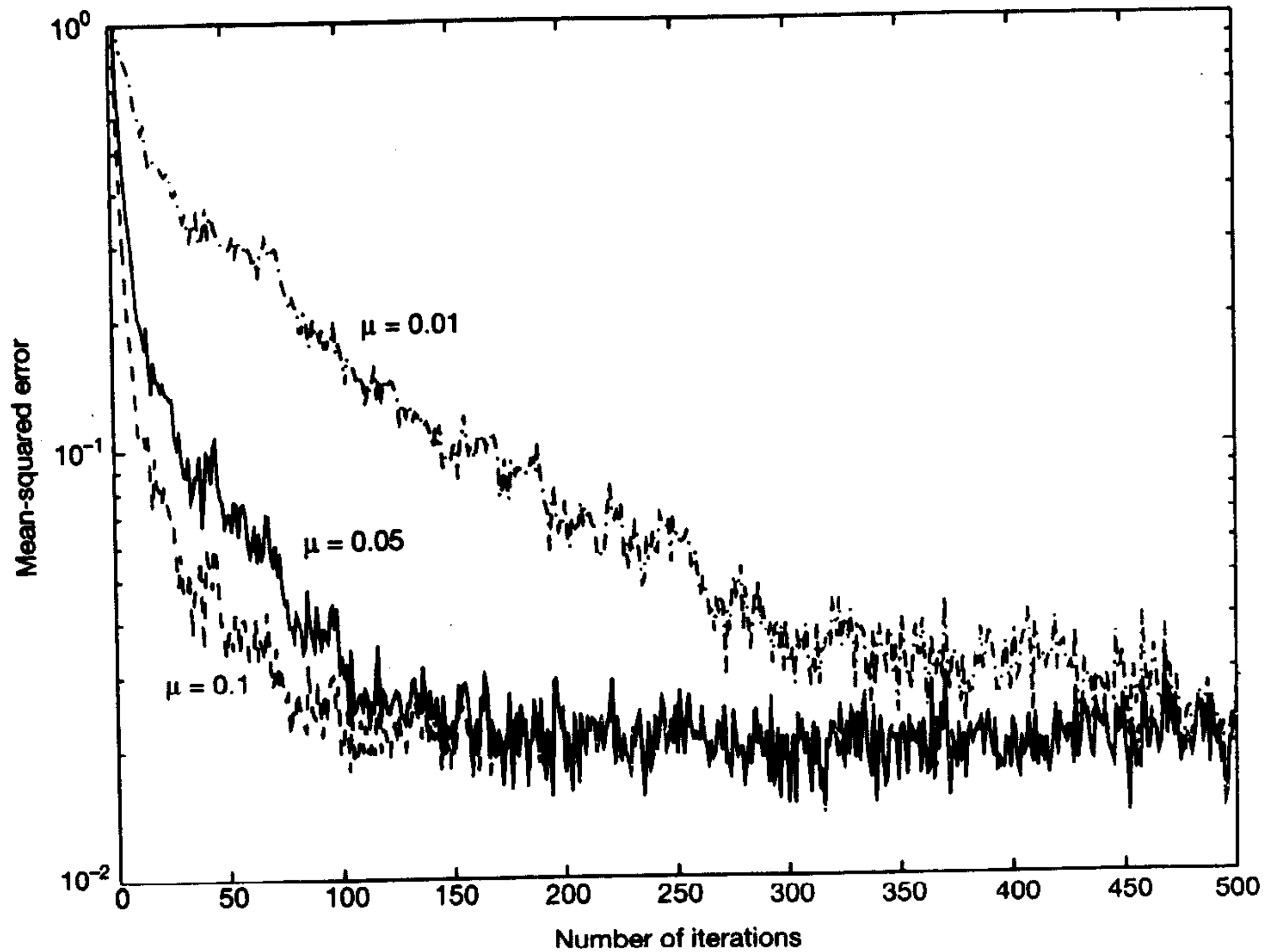


Figure 9.16 Experimental learning curves of adaptive first-order predictor for varying step-size parameter μ.

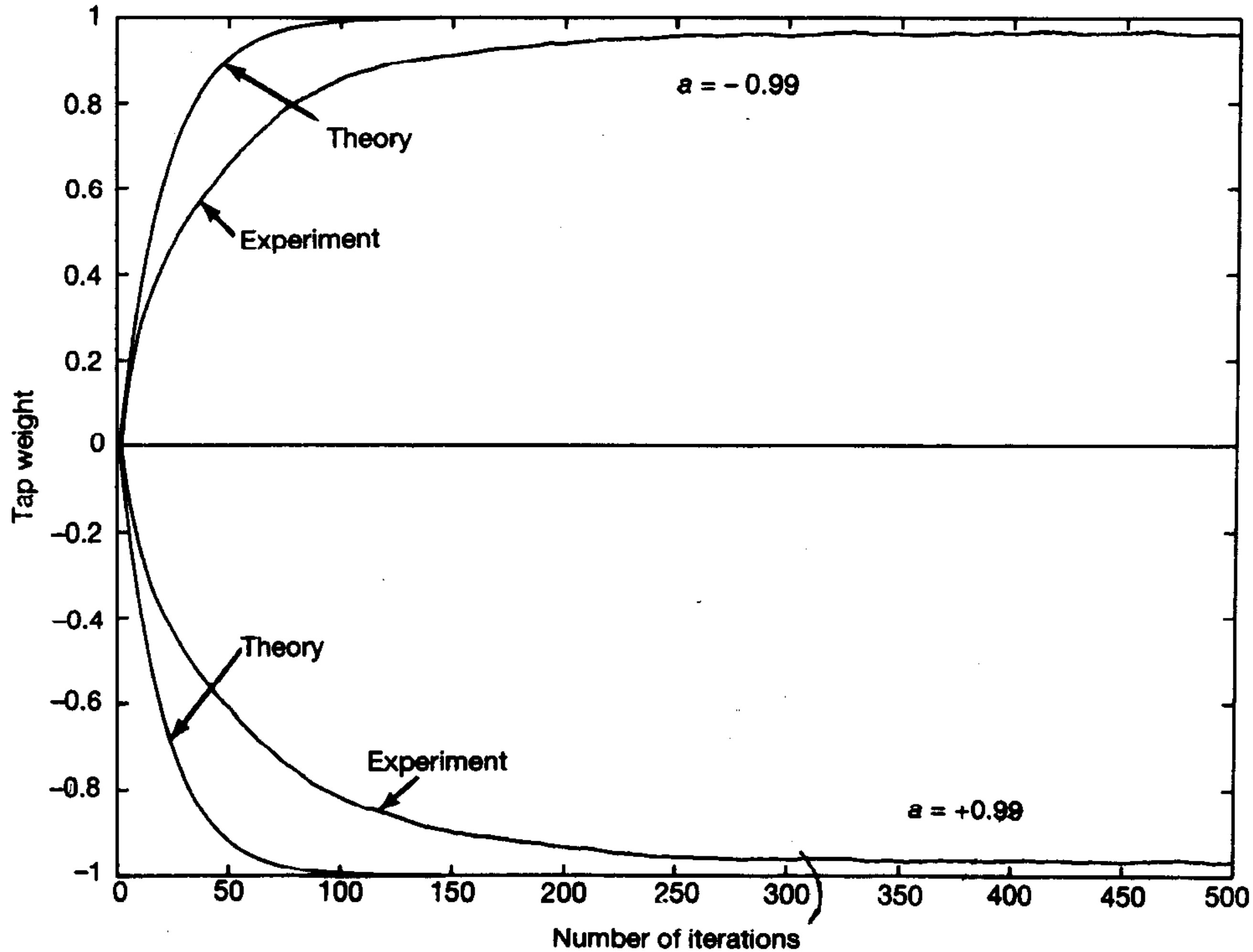


Figure 9.17 Comparison of experimental results with theory, based on $\hat{w}(n)$.

The use of Eq. (9.102), for the two sets of AR parameters listed under (1) and (2) and $\mu = 0.05$, yields the two curves labeled "theoretical" in Fig. 9.17. This figure demonstrates a reasonably good agreement between theory and experiment, which improves with increasing number of iterations.

In Fig. 9.18, we have plotted two learning curves for the LMS algorithm, one obtained experimentally and the other computed from theory. The experimental curve, labeled "Experiment," was obtained by ensemble-averaging the squared value of the prediction error $f(n)$ over 100 independent trials and for varying n. The theoretical curve, labeled "Theory" in Fig. 9.18, was obtained from the following equation:

$$J(n) = \left(\sigma_u^2 - \sigma_v^2\left(1 + \frac{\mu}{2}\sigma_u^2\right)\right)(1 - \mu\sigma_u^2)^{2n} + \sigma_v^2\left(1 + \frac{\mu}{2}\sigma_u^2\right) \tag{9.103}$$

where

$$\sigma_v^2 = (1 - a^2)\sigma_u^2 \tag{9.104}$$

Equation (9.103) follows from the simple working rules presented at the end of Section 9.4, as explained here for the problem at hand:

- The initial value of the mean-squared error $J(n)$ is

$$J(0) = \sigma_u^2$$

- The final value of $J(n)$ is

$$J(\infty) = \sigma_v^2\left(1 + \frac{\mu}{2}\sigma_u^2\right)$$

  which represents the sum of the minimum mean-squared error

$$J_{\min} = \sigma_v^2$$

  and the excess mean-squared error

$$J_{ex}(\infty) \simeq \frac{\mu}{2}\sigma_v^2\lambda_1 = \frac{\mu}{2}\sigma_u^2\sigma_v^2$$

- The average time constant (for small μ) is

$$(\tau)_{\mathrm{mse,av}} = -\frac{1}{2\ln(1 - \mu\lambda_1)} = -\frac{1}{2\ln(1 - \mu\sigma_u^2)} \simeq \frac{1}{2\mu\sigma_u^2}$$

Here also we observe reasonably good agreement between theory and experiment, with the agreement improving as the number of iterations is increased.

9.7 COMPUTER EXPERIMENT ON ADAPTIVE EQUALIZATION

In this second computer experiment we study the use of the LMS algorithm for *adaptive equalization* of a linear dispersive channel that produces (unknown) distortion. Here again we assume that the data are all *real valued*. Figure 9.19 shows the block diagram of the

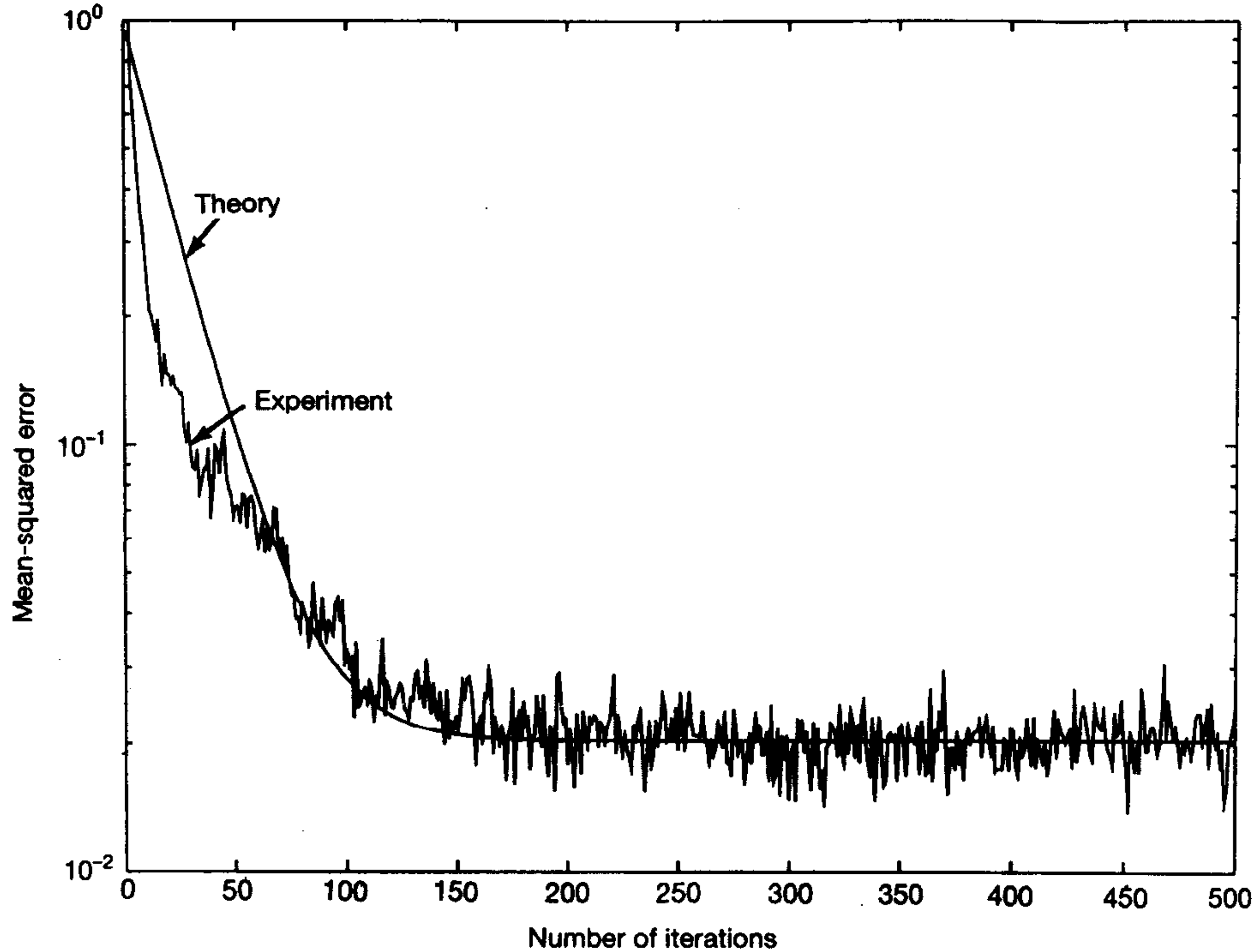


Figure 9.18 Comparison of experimental results with theory for the adaptive predictor, based on the mean-squared error.

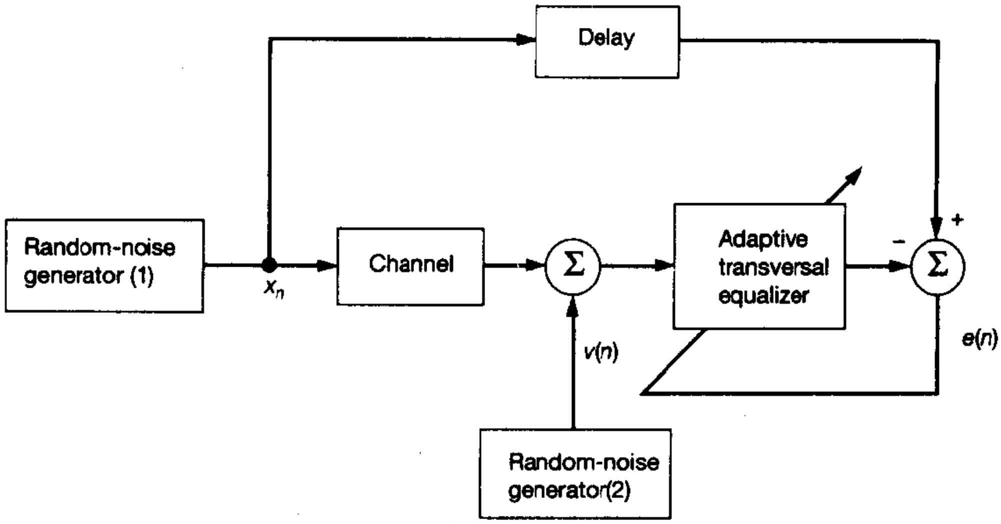


Figure 9.19 Block diagram of adaptive equalizer experiment.

system used to carry out the study. Random number generator 1 provides the test signal x_n, used for probing the channel, whereas random-number generator 2 serves as the source of additive white noise $v(n)$ that corrupts the channel output. These two random-number generators are independent of each other. The adaptive equalizer has the task of correcting for the distortion produced by the channel in the presence of the additive white noise. Random-number generator 1, after suitable delay, also supplies the desired response applied to the adaptive equalizer in the form of a training sequence.

The random sequence $\{x_n\}$ applied to the channel input consists of a *Bernoulli sequence*, with $x_n = \pm 1$ and the random variable x_n having zero mean and unit variance. The impulse response of the channel is described by the raised cosine:[8]

$$h_n = \begin{cases} \frac{1}{2}\left[1 + \cos\left(\frac{2\pi}{W}(n-2)\right)\right], & n = 1, 2, 3 \\ 0, & \text{otherwise} \end{cases} \tag{9.105}$$

where the parameter W controls the amount of amplitude distortion produced by the channel, with the distortion increasing with W.

Equivalently, the parameter W controls the eigenvalue spread $\chi(\mathbf{R})$ of the correlation matrix of the tap inputs in the equalizer, with the eigenvalue spread increasing with W. The

[8]The parameters specified in this experiment closely follow the paper by Satorius and Alexander (1979).

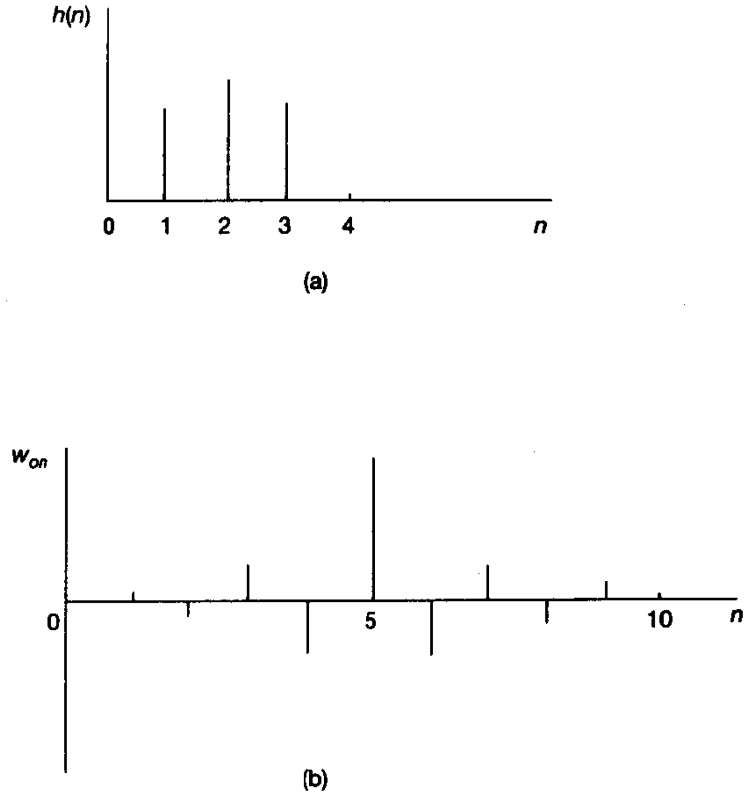


Figure 9.20 (a) Impulse response of channel; (b) impulse response of optimum transversal equalizer.

sequence $v(n)$, produced by the second random generator, has zero mean and variance $\sigma_v^2 = 0.001$.

The equalizer has $M = 11$ taps. Since the channel has an impulse response h_n that is symmetric about time $n = 2$, as depicted in Fig. 9.20(a), it follows that the optimum tap weights w_{on} of the equalizer are likewise symmetric about time $n = 5$, as depicted in Fig. 9.20(b). Accordingly, the channel input x_n is delayed by $\delta = 2 + 5 = 7$ samples to provide the desired response for the equalizer. By selecting the delay δ to match the midpoint of the transversal equalizer, the LMS algorithm is enabled to provide an approximate inversion of both the minimum-phase and nonminimum-phase components of the channel response.

The experiment is in two parts that are intended to evaluate the response of the adaptive equalizer using the LMS algorithm to changes in the eigenvalue spread $\chi(\mathbf{R})$ and step-size parameter μ. Before proceeding to describe the results of the experiment, however, we first compute the eigenvalues of the correlation matrix $\mathbf{R}$ of the 11 tap inputs in the equalizer.

Correlation Matrix of the Equalizer Input

The first tap input of the equalizer at time n equals

$$u(n) = \sum_{k=1}^{3} h_k a(n-k) + v(n) \tag{9.106}$$

where all the parameters are real valued. Hence, the correlation matrix $\mathbf{R}$ of the 11 tap inputs of the equalizer, $u(n), u(n-1), \ldots, u(n-10)$, is a symmetric 11-by-11 matrix. Also, since the impulse response h_n has nonzero values only for $n = 1, 2, 3$, and the noise process $v(n)$ is white with zero mean and variance σ_v^2, the correlation matrix $\mathbf{R}$ is *quintdiagonal*. That is, the only nonzero elements of $\mathbf{R}$ are on the main diagonal and the four diagonals directly above and below it, two on either side, as shown by the special structure:

$$\mathbf{R} = \begin{bmatrix} r(0) & r(1) & r(2) & 0 & \cdots & 0 \\ r(1) & r(0) & r(1) & r(2) & \cdots & 0 \\ r(2) & r(1) & r(0) & r(1) & \cdots & 0 \\ 0 & r(2) & r(1) & r(0) & \cdots & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & 0 & \cdots & r(0) \end{bmatrix} \tag{9.107}$$

where

$$r(0) = h_1^2 + h_2^2 + h_3^3 + \sigma_v^2$$
$$r(1) = h_1 h_2 + h_2 h_3$$
$$r(2) = h_1 h_3$$

The variance $\sigma_v^2 = 0.001$; hence, h_1, h_2, h_3 are determined by the value assigned to parameter W in Eq. (9.105).

In Table 9.1, we have listed (1) values of the autocorrelation function $r(l)$ for lag $l = 0, 1, 2$, and (2) the smallest eigenvalue, λ_{min}, the largest eigenvalue, λ_{max}, and the

TABLE 9.1 SUMMARY OF PARAMETERS FOR THE EXPERIMENT ON ADAPTIVE EQUALIZATION

W	2.9	3.1	3.3	3.5
$r(0)$	1.0963	1.1568	1.2264	1.3022
$r(1)$	0.4388	0.5596	0.6729	0.7774
$r(2)$	0.0481	0.0783	0.1132	0.1511
λ_{min}	0.3339	0.2136	0.1256	0.0656
λ_{max}	2.0295	2.3761	2.7263	3.0707
$\chi(\mathbf{R}) = \lambda_{max}/\lambda_{min}$	6.0782	11.1238	21.7132	46.8216

eigenvalue spread $\chi(\mathbf{R}) = \lambda_{max}/\lambda_{min}$. We thus see that the eigenvalue spread ranges from 6.0782 (for $W = 2.9$) to 46.8216 (for $W = 3.5$).

Experiment 1: Effect of Eigenvalue Spread. For the first part of the experiment, the step-size parameter was held fixed at $\mu = 0.075$. This is in accordance with the condition of Eq. (9.94) for convergence in the mean square for the worst eigenvalue spread of 46.8216 (corresponding to $W = 3.5$):

$$\mu_{crit} = \frac{2}{\text{tap-input power}}$$
$$= \frac{2}{Mr(0)}$$
$$= 0.14$$

The choice of $\mu = 0.075$ therefore assures the convergence of the adaptive equalizer in the mean square for all the conditions listed in Table 9.1.

For each eigenvalue spread, an approximation to the ensemble-averaged learning curve of the adaptive equalizer is obtained by averaging the instantaneous squared error "$e^2(n)$ versus n" curve over 200 independent trials of the computer experiment. The results of this computation are shown in Fig. 9.21.

We thus see from Fig. 9.21 that increasing the eigenvalue spread $\chi(\mathbf{R})$ has the effect of slowing down the rate of convergence of the adaptive equalizer and also increasing the steady-state value of the average squared error. For example, when $\chi(\mathbf{R}) = 6.0782$, approximately 80 iterations are required for the adaptive equalizer to converge in the mean square, and the average squared error (after 500 iterations) approximately equals 0.003. On the other hand, when $\chi(\mathbf{R}) = 46.8216$ (i.e., the equalizer input is ill conditioned), the equalizer requires approximately 200 iterations to converge in the mean square, and the resulting average squared error (after 500 iterations) approximately equals 0.03.

In Fig. 9.22, we have plotted the ensemble-averaged impulse response of the adaptive equalizer after 1000 iterations for each of the four eigenvalue spreads of interest. As before, the ensemble averaging was carried out over 200 independent trials of the experiment. We see that in each case the ensemble-averaged impulse response of the adaptive equalizer is very close to being symmetric with respect to the center tap, as expected. The variation in the impulse response from one eigenvalue spread to another merely reflects the effect of a corresponding change in the impulse response of the channel.

Experiment 2: Effect of Step-Size Parameter. For the second part of the experiment, the parameter W in Eq. (9.105) was fixed at 3.1, yielding an eigenvalue spread of 11.1238 for the correlation matrix of the tap inputs in the equalizer. The step-size parameter μ was this time assigned one of three values: 0.075, 0.025, 0.0075.

Figure 9.23 shows the results of this computation. As before, each learning curve is the result of ensemble averaging the instantaneous squared error "$e^2(n)$ versus n" curve over 200 independent trials of the computer experiment.

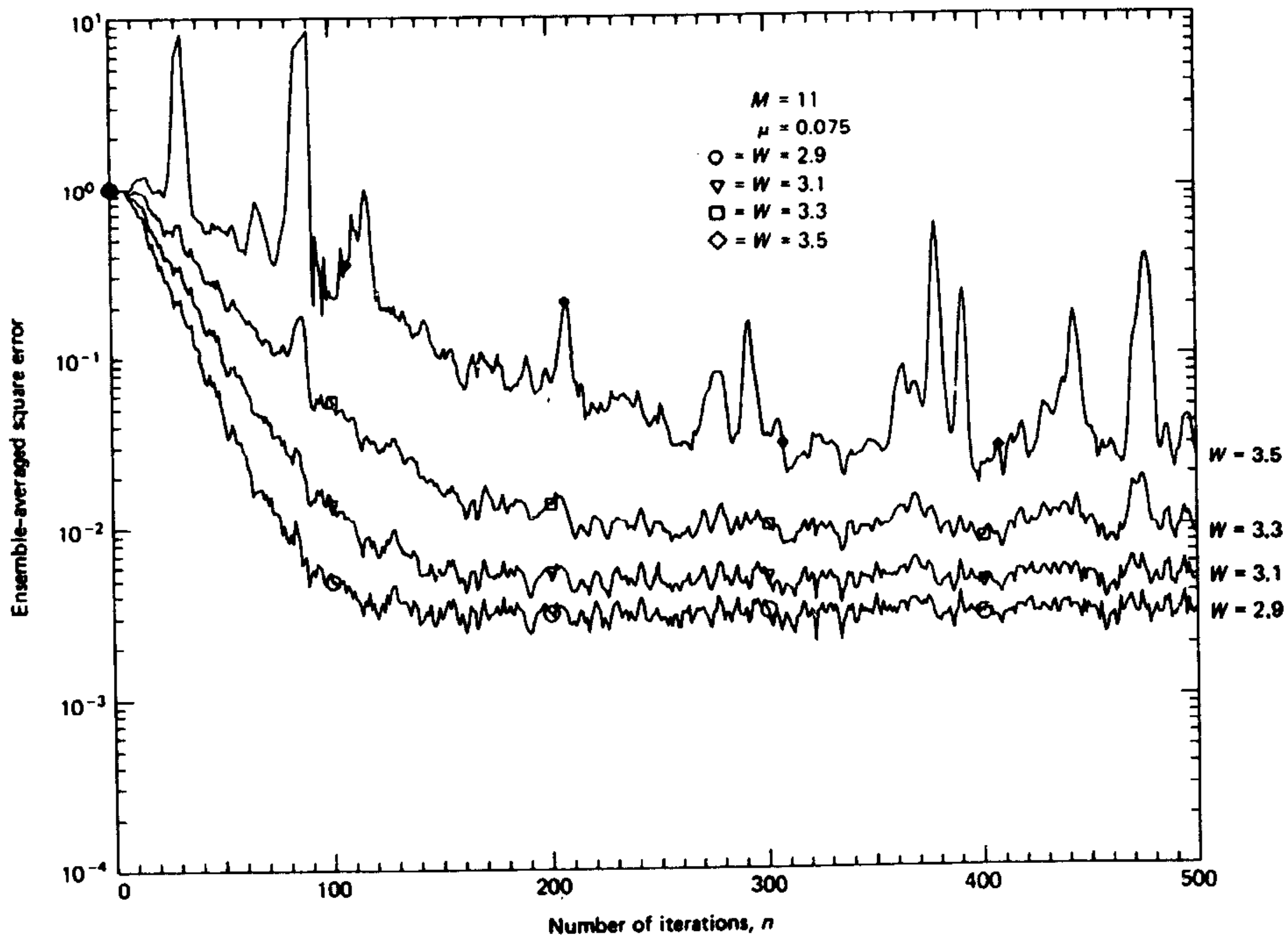


Figure 9.21 Learning curves of the LMS algorithm for adaptive equalizer with number of taps $M = 11$, step-size parameter $\mu = 0.075$, and varying eigenvalue spread $\chi(\mathbf{R})$.

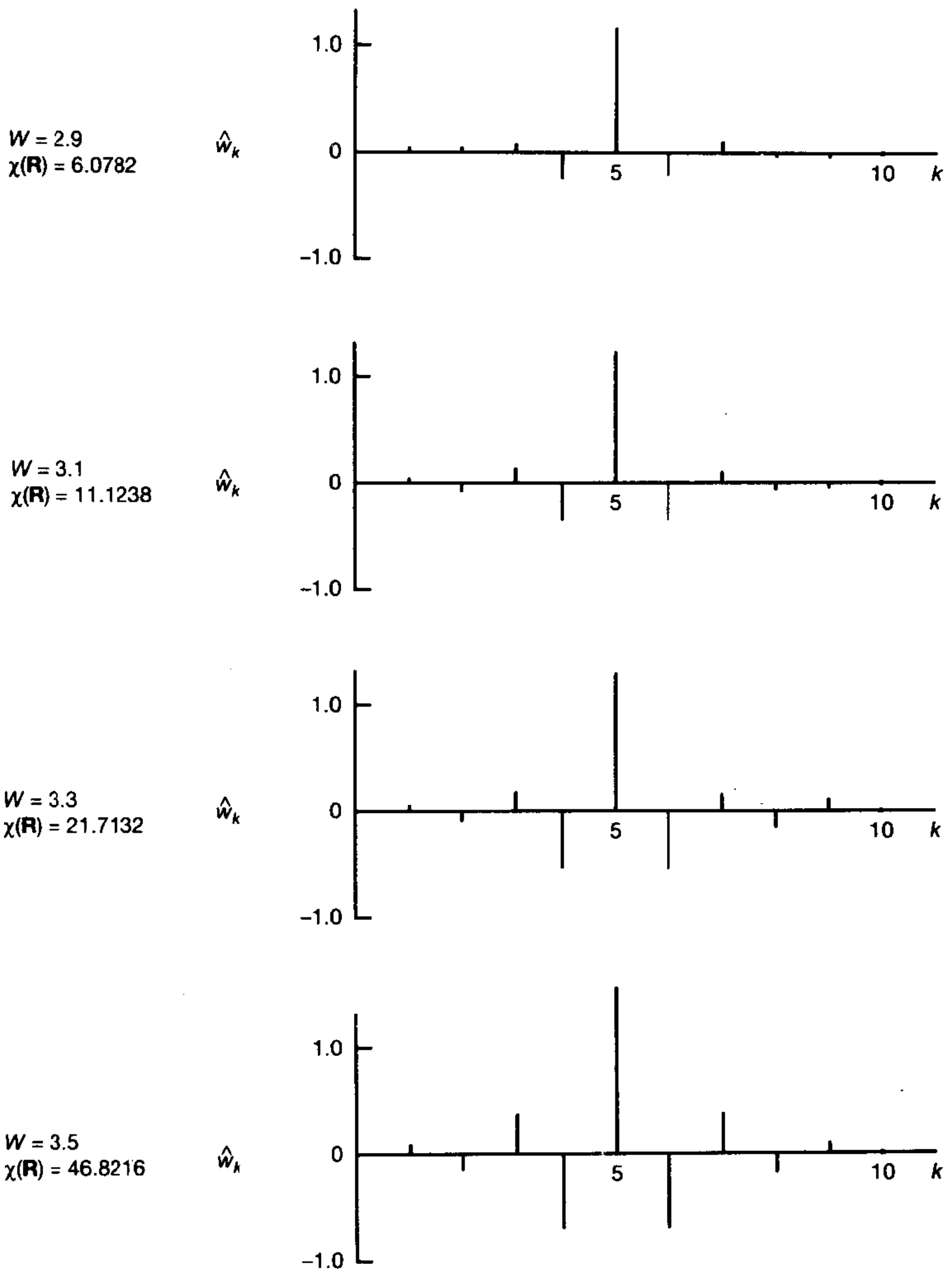


Figure 9.22 Ensemble-averaged impulse response of the adaptive equalizer (after 1000 iterations) for each of four different eigenvalue spreads.

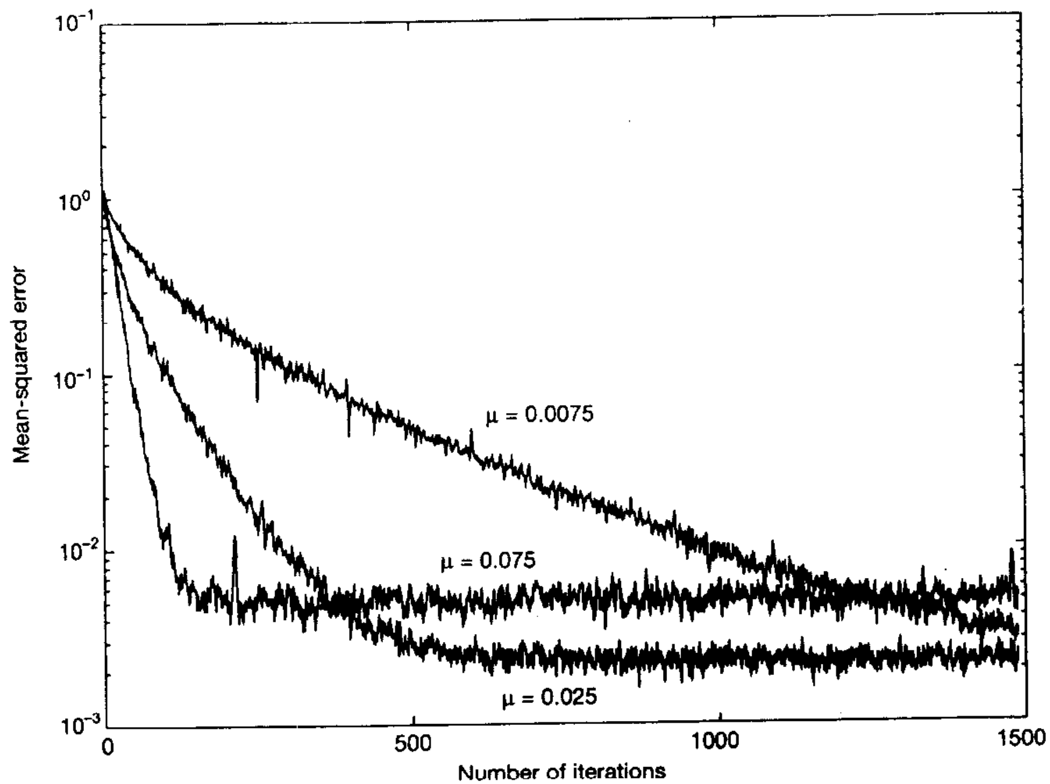


Figure 9.23 Learning curves of the LMS algorithm for adaptive equalizer with the number of taps $M = 11$, fixed eigenvalue spread, and varying step-size parameter μ.

The results confirm that the rate of convergence of the adaptive equalizer is highly dependent on the step-size parameter μ. For large step-size parameter ($\mu = 0.075$), the equalizer converged to steady-state conditions in approximately 120 iterations. On the other hand, when μ is small (equal to 0.0075), the rate of convergence slowed down by more than an order of magnitude. The results also show that the steady-state value of the average squared error (and hence the misadjustment) increases with increasing μ.

9.8 COMPUTER EXPERIMENT ON MINIMUM-VARIANCE DISTORTIONLESS RESPONSE BEAMFORMER

For our final experiment we consider the LMS algorithm applied to an adaptive minimum-variance distortionless response (MVDR) beamformer consisting of a linear array of five uniformly spaced sensors (e.g., antenna elements), as depicted in Fig. 9.24. The spacing d between adjacent elements of the array equals one half of the received wavelength so as to avoid the appearance of grating lobes. The beamformer operates in an environment that consists of two components: a target signal impinging on the array along a direction of interest, and a single source of interference originating from an unknown direction. It is assumed that these two components originate from independent sources, and that the received signal includes additive white Gaussian noise at the output of each sensor.

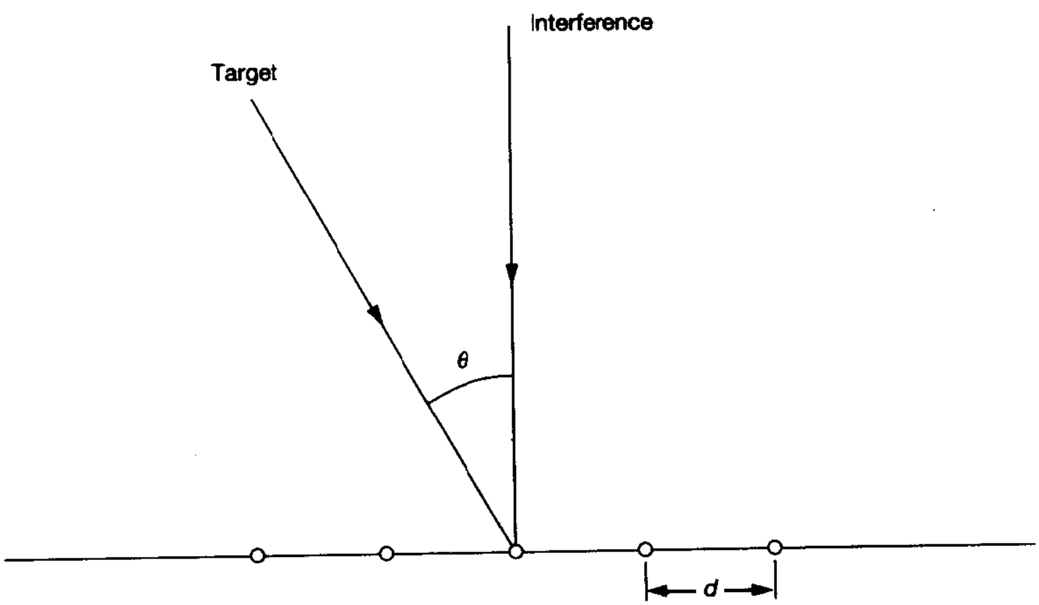


Figure 9.24 Linear array antenna.

The aims of the experiment are twofold:

- To examine the evolution of the adapted spatial response (pattern) of the MVDR beamformer with time for a prescribed target signal-to-interference ratio
- To evaluate the effect of varying the target-to-interference ratio on the interference-nulling performance of the beamformer

The angles of incidence of the target and interfering signals, measured in radians with respect to the normal to the line of the array, are as follows:

- *Target signal*:

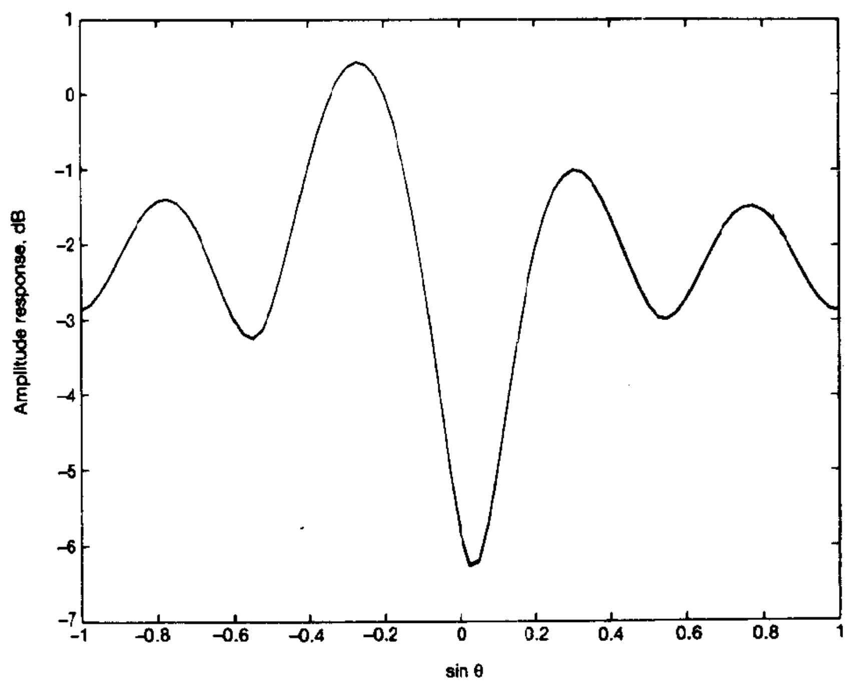
$$\theta_{\text{target}} = \sin^{-1}(-0.2)$$

- *Interference*:

$$\theta_{\text{interf}} = \sin^{-1}(0)$$

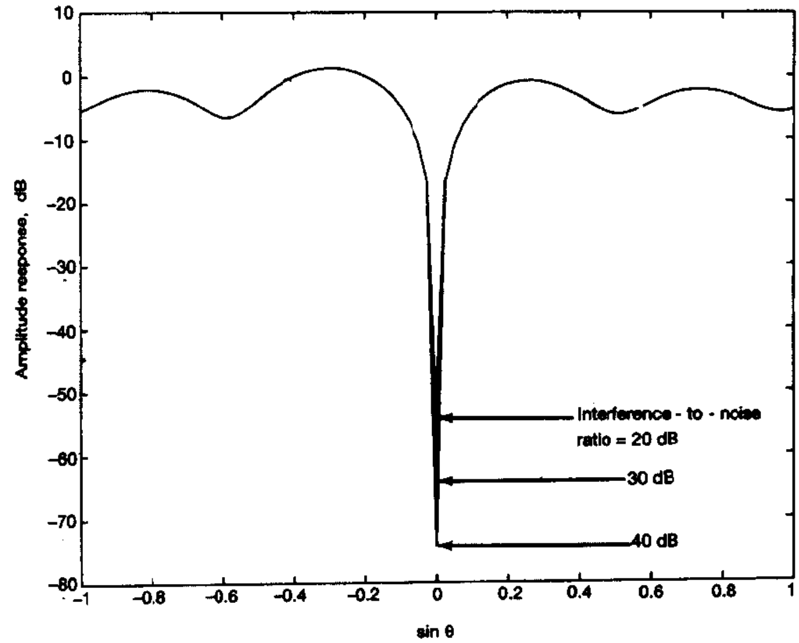
The design of the LMS algorithm for adjusting the weight vector of the adaptive MVDR beamformer follows the theory presented in Section 5.8. For the application at hand, the gain vector $\mathbf{g} = 1$.

Figure 9.25 shows the adapted spatial response of the MVDR beamformer for signal-to-noise ratio = 10 dB, varying interference-to-noise ratio (INR) and varying number

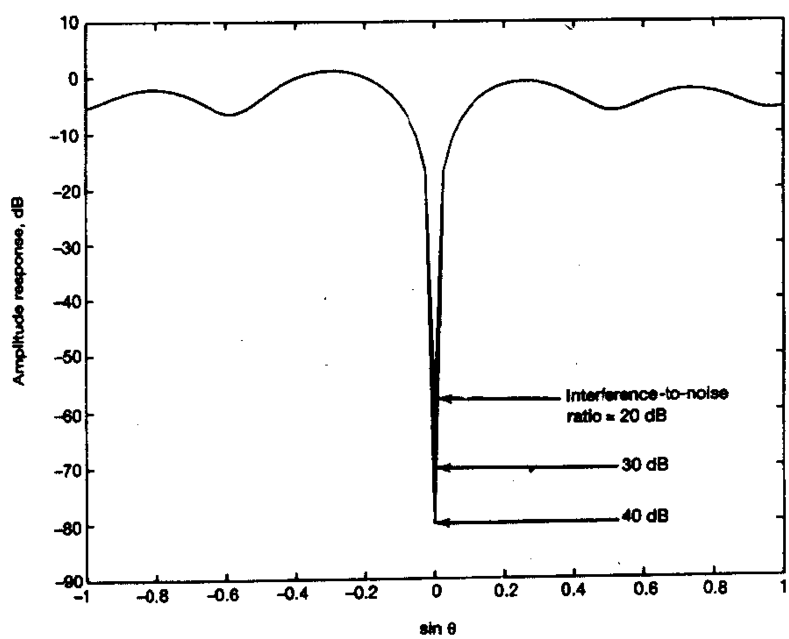


(a)

Figure 9.25 Adapted spatial response of MVDR beamformer for varying interference-to-noise ratio, and varying number of iterations. (a) $n = 20$. (b) $n = 100$. (c) $n = 200$. In each part of the figure, the interference-to-noise ratio assumes one of three values; in part (a) the number of iterations is too small for these variations to have a noticeable effect. Parts (b) and (c) are shown on the next page.



(b)



(c)

Figure 9.25 (*Contd.*)

of iterations. The spatial response is defined by $20 \log_{10}|\hat{\mathbf{w}}^H(n)\mathbf{s}(\phi)|^2$, where $\mathbf{s}(\phi)$ is the steering vector:

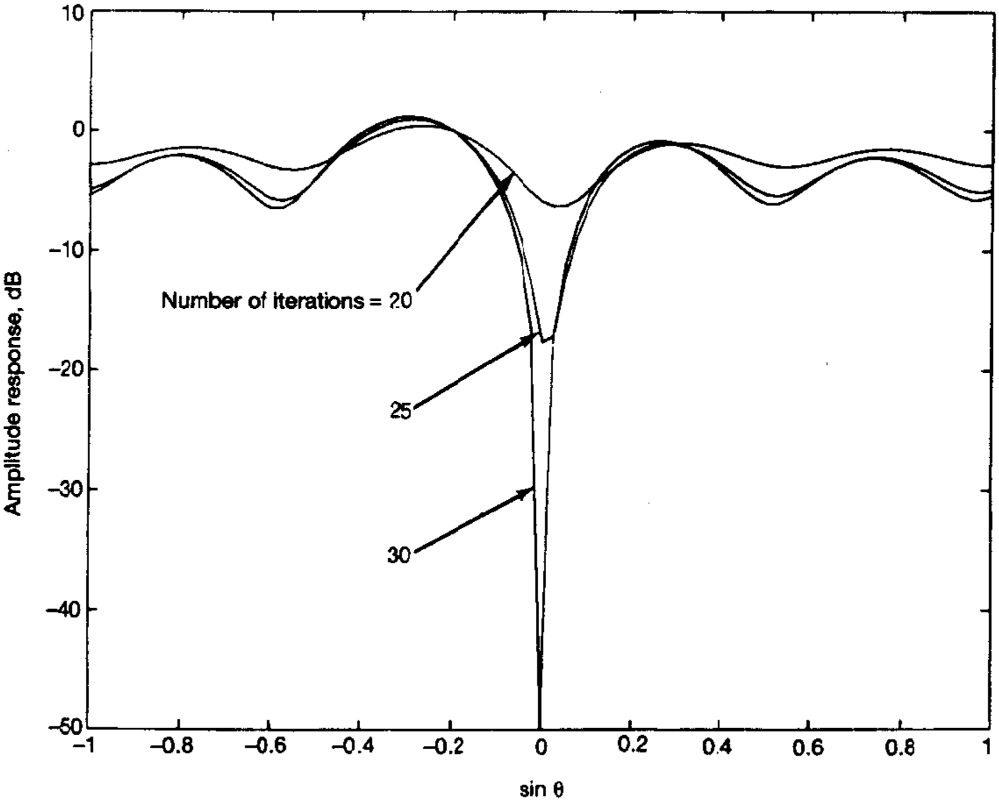
$$\mathbf{s}(\phi) = [1, e^{-j\phi}, e^{-j2\phi}, e^{-j3\phi}, e^{-j4\phi}]^T$$

The electrical angle ϕ, measured in radians, is related to the angle of incidence θ as follows:

$$\phi = \pi \sin\theta$$

The weight vector $\hat{\mathbf{w}}(n)$ of the beamformer is computed using the LMS algorithm with step-size parameter $\mu = 10^{-8}, 10^{-9}$, and 10^{-10} for INR = 20, 30, and 40 dB, respectively. The reason for varying μ is to ensure convergence for a prescribed interference-to-noise ratio, as the largest eigenvalue λ_{max} of the correlation matrix of the input data depends on the interference-to-noise ratio.

Figure 9.26 shows the adapted spatial response of the MVDR beamformer after 20, 25, and 30 iterations. The three curves of the figure pertain to INR = 20 dB and a fixed target signal-to-noise ratio = 10 dB.



Figures 9.26 Adapted spatial response of MVDR beamformer for signal-to-noise ratio = 10dB, interference-to-noise ratio = 20dB, step-size parameter = 10^{-8}, and varying number of iterations.

On the basis of the results presented in Figs. 9.25 and 9.26, we may make the following observations:

- The response of the MVDR beamformer is always held fixed at the value of unity along the prescribed angle of incidence $\theta_{target} = \sin^{-1}(-0.2)$, as required.
- The interference-nulling capability of the beamformer improves with (a) increasing number of iterations (snapshots of data), and (b) increasing interference-to-target signal ratio.

9.9 DIRECTIONALITY OF CONVERGENCE OF THE LMS ALGORITHM FOR NON-WHITE INPUTS

The eigenstructure of the correlation matrix **R** of a transversal filter's tap inputs has a profound impact on the convergence behavior of the LMS algorithm used to adapt the filter's tap weights. When the tap inputs are drawn from a white noise process, the tap inputs are uncorrelated and the eigenvalue spread $\chi(\mathbf{R})$ of the correlation matrix **R** is unity, with the result that the LMS algorithm enjoys a non-directional convergence. At the other extreme, when the tap inputs are highly correlated and the eigenvalue spread $\chi(\mathbf{R})$ is large, which does happen under non-white inputs, convergence of the LMS algorithm (like the steepest descent algorithm from which it is derived) takes on a directional nature. Moreover, in such an environment, initialization may play a significant role in determining the rate of convergence. Indeed, it is the combination of these factors that is responsible for the relatively slow rate of convergence observed in the results on adaptive equalization presented in Fig. 9.21.

The *directionality of convergence*[9], exhibited by the LMS algorithm under non-white inputs, manifests itself in two ways:

1. The speed of convergence of the LMS algorithm is faster in some directions in the algorithm's weight space than in some other directions.
2. Depending on the direction along which the convergence of the LMS algorithm takes place, it is possible for the convergence to be accelerated by an increase in the eigenvalue spread $\chi(\mathbf{R})$.

These two aspects of the directionality of convergence are illustrated in the following example.

[9]The material presented in this section, including Example 7, is based on: LeBlanc, J.P., private communication, 1995.

Example 7: Sinusoidal Process

Consider the simple example of a two-tap transversal filter, whose true tap-weight vector (i.e., the Wiener solution) is denoted by $\mathbf{w}_o$. The tap inputs $u(n)$ and $u(n-1)$ are drawn from a deterministic process consisting of two sinusoids, as shown by

$$u(n) = A_1 \cos(\omega_1 n) + A_2 \cos(\omega_2 n)$$

where ω_1 and ω_2 are the angular frequencies of the two sinusoids, and A_1 and A_2 are their respective amplitudes. The correlation matrix of the tap inputs is

$$\mathbf{R} = \begin{bmatrix} E[u^2(n)] & E[u(n-1)u(n)] \\ E[u(n)u(n-1)] & E[u^2(n-1)] \end{bmatrix}$$

$$= \frac{1}{2}\begin{bmatrix} A_1^2 + A_2^2 & A_1^2 \cos\omega_1 + A_2^2 \cos\omega_2 \\ A_1^2 \cos\omega_1 + A_2^2 \cos\omega_2 & A_1^2 + A_2^2 \end{bmatrix}$$

This two-by-two matrix is doubly symmetric, which means that its two eigenvalues and associated eigenvectors are as follows (see Problem 17 of Chapter 4):

$$\lambda_1 = \frac{1}{2}A_1^2(1 + \cos\omega_1) + \frac{1}{2}A_2^2(1 + \cos\omega_2); \qquad \mathbf{q}_1 = [1, 1]^T$$

$$\lambda_2 = \frac{1}{2}A_1^2(1 - \cos\omega_1) + \frac{1}{2}A_2^2(1 - \cos\omega_2); \qquad \mathbf{q}_2 = [-1, 1]^T$$

In the sequel, we study the convergence behavior of the LMS algorithm with the following specifications:

step-size parameter, $\mu = 0.01$
initial condition, $\hat{\mathbf{w}}(0) = [0, 0]^T$
total number of iterations, $n = 200$

In particular, we consider two different filters and two different inputs. One filter is the *minimum eigenfilter*, whose tap-weight vector is defined by the eigenvector $\mathbf{q}_2$ associated with the smallest eigenvalue (i.e., λ_2) of the correlation matrix $\mathbf{R}$, and the other filter is the *maximum eigenfilter* defined by the eigenvector $\mathbf{q}_1$ associated with the largest eigenvalue (i.e., λ_1). The two inputs are

$$u_a(n) = \cos(1.2n) + 0.5\cos(0.1n)$$

$$u_b(n) = \cos(0.6n) + 0.5\cos(0.23n)$$

The first input has an eigenvalue spread $\chi(\mathbf{R}) = 2.9$, and the second input has an eigenvalue spread $\chi(\mathbf{R}) = 12.9$. Thus, there are four distinct combinations to be considered, which we do under the following two cases.

Case 1. Minimum eigenfilter. For this case, the true tap-weight vector of the transversal filter is

$$\mathbf{w}_o = \mathbf{q}_2 = [-1, 1]^T$$

Here under input $u_a(n)$, the convergence of the LMS algorithm is along a "slow" trajectory, and transverses about halfway to the true parameterization of the filter in 200 iterations of the algorithm starting from $\mathbf{w}(0) = [0, 0]^T$, as shown in Fig. 9.27(a).

Next, the input signal is chosen as $u_b(n)$, for which the eigenvalue spared $\chi(\mathbf{R})$ is 12.9, compared to 2.9 for $u_a(n)$. The increased eigenvalue spread is evidenced by an increased eccentricity of the error surface contours, as portrayed in Fig. 9.27(b). Comparing Fig. 9.27(b) with 9.27(a), we see that in Case 1 the convergence of the LMS algorithm has been *decelerated* by the increase in the eigenvalue spread $\chi(\mathbf{R})$.

Case 2. Maximum eigenfilter. For this second case, the true tap-weight vector of the transversal filter is

$$\mathbf{w}_o = \mathbf{q}_1 = [1, 1]^T$$

Reverting to the input $u_a(n)$, we now find that the convergence of the LMS algorithm is along a "fast" trajectory, and traverses the error surface contours, as shown in Fig. 9.28(a). Moreover, when the input signal $u_b(n)$ is used, the convergence of the LMS algorithm is *accelerated* by the increase in the eigenvalue spread $\chi(\mathbf{R})$, as shown in Fig. 9.28(b).

In the example described here, the initial condition $\hat{\mathbf{w}}(0)$ is fixed, but the true tap-weight vector $\mathbf{w}_o$ is varied from case 1 to case 2. In the usual application of the LMS algorithm to stationary inputs, the true tap-weight vector $\mathbf{w}_o$ is fixed but unknown. For some fixed $\mathbf{w}_o$, we may equivalently specify the initial condition for this example as follows:

$$\hat{\mathbf{w}}(0) = \begin{cases} \mathbf{w}_o - \mathbf{q}_2 \text{ for case 1 (minimum eigenfilter)} \\ \mathbf{w}_o - \mathbf{q}_1 \text{ for case 2 (maximum eigenfilter)} \end{cases}$$

Thus, in light of the results presented in this example, the directionality of convergence of the LMS algorithm may be exploited by choosing a suitable value for the initial condition $\hat{\mathbf{w}}(0)$, such that the algorithm is guided along a fast trajectory. This, of course, assumes the availability of *prior knowledge* about the environment in which the LMS algorithm is operating. In such a scenario, the LMS algorithm performs essentially the role of "tuning" the tap-weights of the transversal filter.

9.10 ROBUSTNESS OF THE LMS ALGORITHM

The development of the LMS algorithm presented in Section 9.2 was carried out in a heuristic manner, starting from the method of steepest descent as the basis for computing the Wiener solution of an adaptive transversal filter. However, once instantaneous estimates of the correlation matrix $\mathbf{R}$ and cross-correlation vector $\mathbf{p}$ are invoked in this development, links with the least-mean-square estimate implicit in the Wiener solution are destroyed. If then a "single" realization of the LMS algorithm is not optimum in the least-mean-square sense, what is the actual criterion on the basis of which it is optimum? The

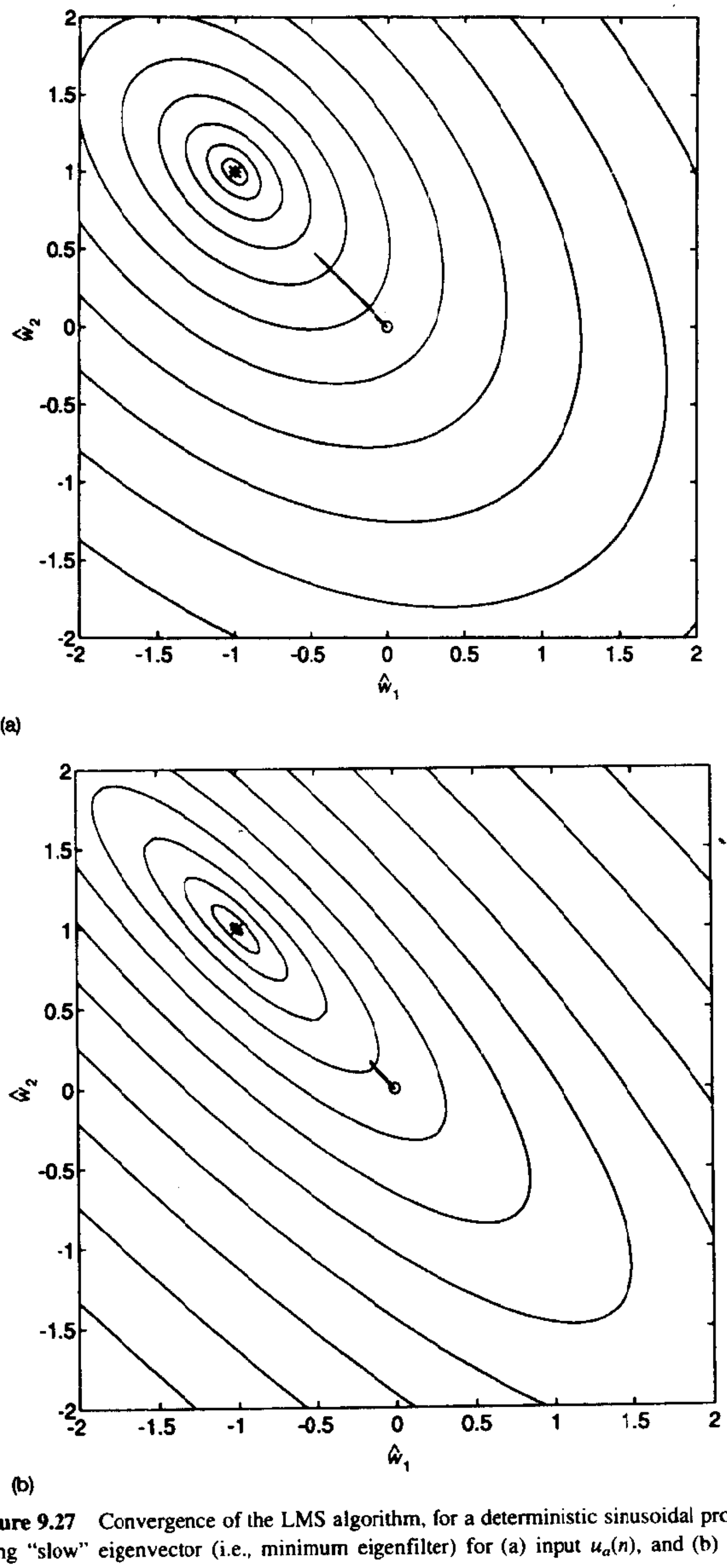
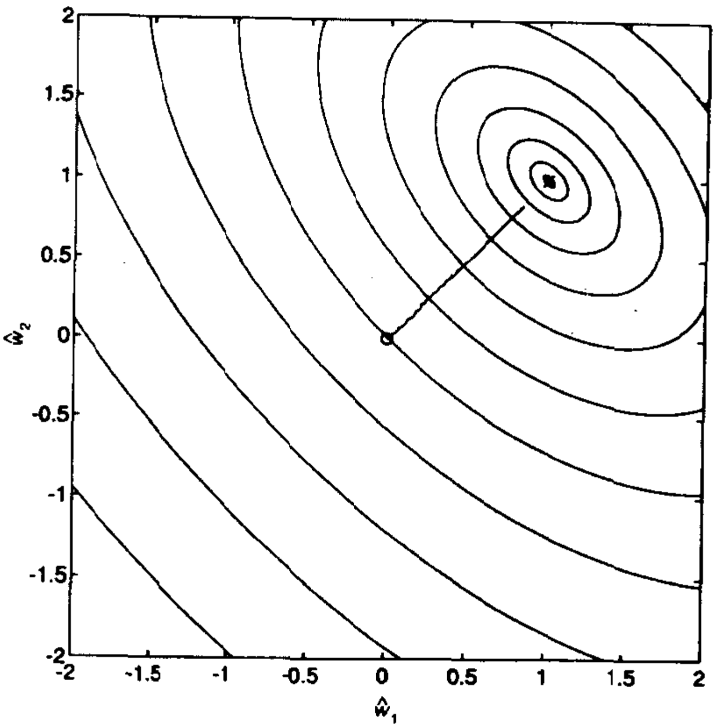
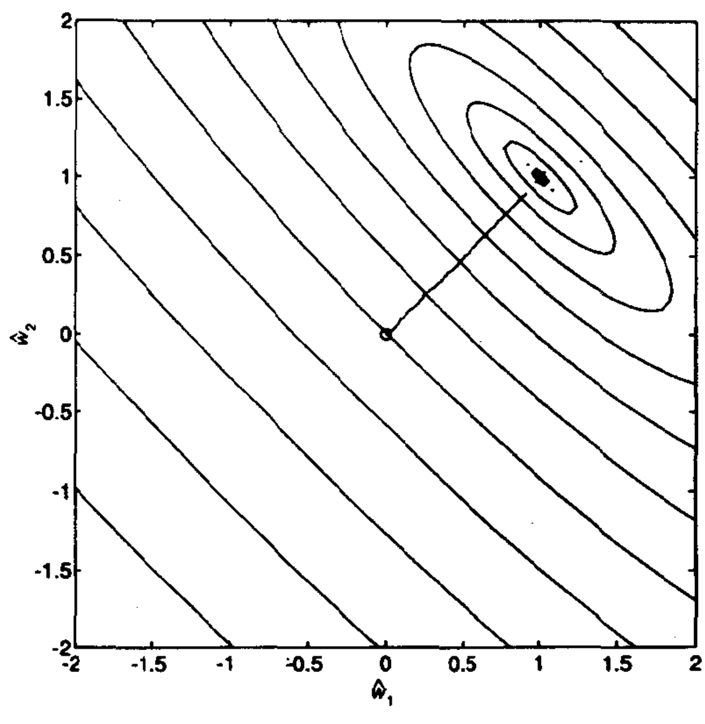


Figure 9.27 Convergence of the LMS algorithm, for a deterministic sinusoidal process, along "slow" eigenvector (i.e., minimum eigenfilter) for (a) input $u_a(n)$, and (b) input $u_b(n)$.



(a)



(b)

Figure 9.28 Convergence of the LMS algorithm, for deterministic sinusoidal process, along "fast" eigenvector (i.e., maximum eigenfilter) for (a) input $u_a(n)$, and (b) input $u_b(n)$.

answer to this fundamental question lies in the so-called H^∞ *(or minimax) criterion*,[10] on which extensive studies were carried out in the field of robust control during the 1980s.

To proceed, suppose that we have a set of noisy measurements that fit into a multiple regression model of order M as follows:

$$d(i) = \mathbf{w}_o^H \mathbf{u}(i) + v(i), \quad i = 0, 1, \ldots, n \tag{9.108}$$

The issue of concern is to formulate a recursive estimate of the unknown weight vector $\mathbf{w}_o$, given input-output pairs $\{\mathbf{u}(i), d(i) | i = 0, 1, \ldots, n\}$ that are corrupted by the additive white noise $v(i)$. The estimate, denoted by $\hat{\mathbf{w}}(i)$, is required to be optimum in a certain sense, yet to be defined. Suppose, further, we do two other things:

- A positive real number μ is chosen so as to satisfy the condition

$$0 < \mu \le \frac{1}{\|\mathbf{u}(i)\|^2} \qquad \text{for } 0 \le i \le n \tag{9.109}$$

- Subsequently, an estimate $\hat{\mathbf{w}}(i)$ is chosen *at will* for the unknown weight vector $\mathbf{w}_o$ at iteration i. This estimate always satisfies the condition

$$\frac{|\hat{\mathbf{w}}^H(i)\mathbf{u}(i) - \mathbf{w}_o^H\mathbf{u}(i)|^2}{\mu^{-1}\|\hat{\mathbf{w}}(i) - \mathbf{w}_o\|^2} \le 1 \qquad \text{for } 0 \le i \le n \tag{9.110}$$

  where the numerator of the left-hand side is the squared error between the estimate $\hat{\mathbf{w}}^H(i)\mathbf{u}(i)$ and the true inner product $\mathbf{w}_o^H\mathbf{u}(i)$ in Eq. (9.108), and the denominator (except for the scaling factor μ^{-1}) is the squared Euclidean distance between the estimate $\hat{\mathbf{w}}(i)$ and the true weight vector $\mathbf{w}_o$.

Note that the condition described in (9.110) follows from that of (9.109). First, we write

$$\begin{aligned} |\hat{\mathbf{w}}^H(i)\mathbf{u}(i) - \mathbf{w}_o^H\mathbf{u}(i)|^2 &= |(\hat{\mathbf{w}}(i) - \mathbf{w}_o)^H\mathbf{u}(i)|^2 \\ &\le \|\hat{\mathbf{w}}(i) - \mathbf{w}_o\|^2\|\mathbf{u}(i)\|^2 \end{aligned} \tag{9.111}$$

where, in the last line, we have invoked the *Cauchy–Schwarz inequality*.[11] Hence, in light of the inequality in (9.109), we may readily recast that of Eq. (9.111) in the form previously specified in (9.110).

[10] The material presented in this section follows Sayed and Kailath (1994); see also Sayed and Rupp (1994). The H^∞ criterion is due to Zames (1981), and it is developed in Zames and Francis (1983) and Kimura (1984). The criterion is discussed in Doyle et al. (1989), Khargonekar and Nagpal (1991), and Green and Limebeer (1995). The paper by Sayed and Rupp (1994) also deals with time-variant step-sizes $\mu(i)$, and their results are therefore applicable to other forms of the LMS algorithm, e.g., normalized LMS; the latter algorithm is discussed in the next section.

[11] Consider the inner product $\mathbf{a}^H\mathbf{b}$ of two vectors $\mathbf{a}$ and $\mathbf{b}$ of compatible dimensions. The Cauchy–Schwarz inequality states that

$$|\mathbf{a}^H\mathbf{b}|^2 \le \|\mathbf{a}\|^2\,\|\mathbf{b}\|^2$$

where $\|\bullet\|$ denotes Euclidean norm of the enclosed vector:

Given that the inequality of (9.110) holds for arbitrary $\hat{\mathbf{w}}(i)$, then it will still hold if, say, the denominator is increased by the squared amplitude of the noise, namely, $|v(i)|^2$, as shown by

$$\frac{|\hat{\mathbf{w}}^H(i)\mathbf{u}(i) - \mathbf{w}_o^H\mathbf{u}(i)|^2}{\mu^{-1}\|\hat{\mathbf{w}}(i) - \mathbf{w}_o\|^2 + |v(i)|^2} \leq 1, \qquad 0 \leq i \leq n \tag{9.112}$$

At this point in the discussion, we may ask: In what particular way would this inequality be modified if $\hat{\mathbf{w}}(i)$ is chosen according to the weight update computed by the LMS algorithm? It turns out that, in fact, the inequality is further tightened by such an update, in that we may also write

$$\frac{|\hat{\mathbf{w}}^H(i)\mathbf{u}(i) - \mathbf{w}_o^H\mathbf{u}(i)|^2}{\mu^{-1}\|\hat{\mathbf{w}}(i) - \mathbf{w}_o\|^2 - \mu^{-1}\|\hat{\mathbf{w}}(i+1) - \mathbf{w}_o\|^2 + |v(i)|^2} \leq 1 \tag{9.113}$$

Comparing this inequality with that of (9.112), we see that the denominator has been reduced by subtracting the new term $\mu^{-1}\|\hat{\mathbf{w}}(i+1) - \mathbf{w}_o\|^2$. To verify the validity of the inequality of (9.113), we first rewrite it in the equivalent form

$$\mu^{-1}\|\hat{\mathbf{w}}(i) - \mathbf{w}_o\|^2 - \mu^{-1}\|\hat{\mathbf{w}}(i+1) - \mathbf{w}_o\|^2 + |v(i)|^2 - |\hat{\mathbf{w}}^H(i)\mathbf{u}(i) - \mathbf{w}_o^H\mathbf{u}(i)|^2 \geq 0 \tag{9.114}$$

Next, using the update recursion for the LMS algorithm, namely,

$$\hat{\mathbf{w}}(i+1) = \hat{\mathbf{w}}(i) + \mu\, \mathbf{u}(i)\, (d(i) - \hat{\mathbf{w}}^H(i)\mathbf{u}(i))^* \tag{9.115}$$

it is a straightforward matter to show that the left-hand side of the inequality of (9.114) may be simplified, as shown here (see Problem 11):

$$(1 - \mu\|\mathbf{u}(i)\|^2)|d(i) - \hat{\mathbf{w}}^H(i)\mathbf{u}(i)|^2 \geq 0 \tag{9.116}$$

Finally, we see that this inequality does indeed hold, provided that in the first place the step-size parameter μ of the LMS algorithm is chosen to satisfy the condition of (9.109).

Suppose, now, the LMS algorithm is computed up to some iteration n, such that we satisfy the requirement[12]

$$0 < \mu < \min_{0 \leq i \leq n} \frac{1}{\|\mathbf{u}(i)\|^2} \tag{9.117}$$

Then, by virtue of the inequality described in (9.114), we have for each iteration i of the LMS algorithm:

$$|\hat{\mathbf{w}}^H(i)\mathbf{u}(u) - \mathbf{w}_o^H\mathbf{u}(i)\|^2 \leq \mu^{-1}\|\hat{\mathbf{w}}(i) - \mathbf{w}_o\|^2 - \mu^{-1}\|\hat{\mathbf{w}}(i+1) - \mathbf{w}_o\|^2 + |v(i)|^2 \tag{9.118}$$

Hence, summing both sides of this inequality over $0 \leq i \leq n$, we get

$$\sum_{i=0}^{n} |\hat{\mathbf{w}}^H(i)\mathbf{u}(i) - \mathbf{w}_o^H\mathbf{u}(i)|^2 \leq \mu^{-1}\|\hat{\mathbf{w}}(0) - \mathbf{w}_o\|^2 - \mu^{-1}\|\hat{\mathbf{w}}(n+1) - \mathbf{w}_o\|^2 + \sum_{i=0}^{n} |v(i)|^2$$

[12]Note that the restriction imposed on the step-size parameter μ in Eq. (9.117) is somewhat different from the condition imposed on μ for convergence of the LMS algorithm in the mean square.

If this condition holds, then we certainly have

$$\sum_{i=0}^{n} |\hat{\mathbf{w}}^H(i)\mathbf{u}(i) - \mathbf{w}_o^H\mathbf{u}(i)|^2 \le \mu^{-1}\|\hat{\mathbf{w}}(0) - \mathbf{w}_o\|^2 + \sum_{i=0}^{n} |\nu(i)|^2 \qquad (9.119)$$

On the basis of this last inequality, we may equivalently write

$$\frac{\sum_{i=0}^{n} |\hat{\mathbf{w}}^H(i)\mathbf{u}(i) - \mathbf{w}_o^H\mathbf{u}(i)|^2}{\mu^{-1}\|\hat{\mathbf{w}}(0) - \mathbf{w}_o\|^2 + \sum_{i=0}^{n} |\nu(i)|^2} \le 1 \qquad (9.120)$$

which is the fundamental result that we have been seeking. The numerator and denominator of the left-hand side of this inequality may be interpreted as follows:

- The difference term $\hat{\mathbf{w}}^H(i)\mathbf{u}(i) - \mathbf{w}_o^H\mathbf{u}(i)$ represents the error produced in using the inner product $\hat{\mathbf{w}}^H(i)\mathbf{u}(i)$ as an estimate of the true quantity $\mathbf{w}_o^H\mathbf{u}(i)$. Accordingly, the numerator may be viewed as the *sum of squared errors* so incurred over the entire computation interval $0 \le i \le n$.
- The denominator term consists of the sum of two terms: a scaled version of the squared Euclidean distance between the initial weight vector $\hat{\mathbf{w}}(0)$ and the true weight vector $\mathbf{w}_o$, and the sum of squared noise $\nu(i)$ over the same interval $0 \le i \le n$.

Let T denote the operator that maps the set of disturbances $\{\nu(i)|i = 0, 1, \ldots, n\}$ and the initial weight uncertainty $\mu^{-1/2}(\hat{\mathbf{w}}(0) - \mathbf{w}_o)$ to the corresponding set of errors $\{\hat{\mathbf{w}}^H(i)\mathbf{u}(i) - \mathbf{w}_o^H\mathbf{u}(i) \mid i = 0, 1, \ldots, n\}$. Then, the inequality described in (9.120) states that *the Euclidean norm induced by the operator T is always bounded by one* (Sayed and Kailath, 1994). Stated in another way, the sum of squared errors is always upper bounded by the combined effects of the initial weight uncertainty $(\hat{\mathbf{w}}(0) - \mathbf{w}_o)$ and the noise $\nu(i)$, which explains the "robust" behavior of the LMS algorithm in its endeavour to estimate the uncorrupted term $\mathbf{w}_o^H\,\mathbf{u}(i)$ defined in (9.108).

In conclusion, we may say that a single realization of the LMS algorithm (despite its name) is not optimal in the least-mean-square sense, but is optimal in the H^∞ sense.

9.11 NORMALIZED LMS ALGORITHM

In the standard form of LMS algorithm, the correction $\mu\mathbf{u}(n)e^*(n)$ applied to the tap-weight vector $\hat{\mathbf{w}}(n)$ at iteration $n + 1$ is directly proportional to the tap-input vector $\mathbf{u}(n)$.

Therefore, when $\mathbf{u}(n)$ is large, the LMS algorithm experiences a *gradient noise amplification* problem. To overcome this difficulty, we may use the *normalized LMS algorithm*,[13] which is the companion to the ordinary LMS algorithm. In particular, the correction applied to the tap-weight vector $\hat{\mathbf{w}}(n)$ at iteration $n + 1$ is "normalized" with respect to the squared Euclidean norm of the tap-input vector $\mathbf{u}(n)$ at iteration n, hence the term "normalized."

We may formulate the normalized LMS algorithm as a natural modification of the ordinary LMS algorithm. Alternatively, we may derive the normalized LMS algorithm in its own rightful manner; we follow the latter procedure here as it provides insight into its operation.

Normalized LMS Algorithm as the Solution to a Constrained Optimization Problem

The normalized LMS algorithm may be viewed as the solution to a constrained optimization (minimization) problem (Goodwin and Sin, 1984). Specifically, the problem of interest may be stated as follows:

> Given the tap-input vector $\mathbf{u}(n)$ and the desired response $d(n)$, determine the tap-weight vector $\hat{\mathbf{w}}(n + 1)$ so as to minimize the squared Euclidean norm of the change
>
> $$\delta\hat{\mathbf{w}}(n + 1) = \hat{\mathbf{w}}(n + 1) - \hat{\mathbf{w}}(n) \tag{9.121}$$
>
> in the tap-weight vector $\hat{\mathbf{w}}(n + 1)$ with respect to its old value $\hat{\mathbf{w}}(n)$, subject to the constraint
>
> $$\hat{\mathbf{w}}^H(n + 1)\mathbf{u}(n) = d(n) \tag{9.122}$$

To solve this constrained optimization problem, we may use the *method of Lagrange multipliers*.[14]

The squared norm of the change $\delta\hat{\mathbf{w}}(n + 1)$ in the tap-weight vector $\hat{\mathbf{w}}(n + 1)$ may be expressed as

$$\begin{aligned}\|\delta\hat{\mathbf{w}}(n + 1)\|^2 &= \delta\hat{\mathbf{w}}^H(n + 1)\delta\hat{\mathbf{w}}(n + 1)\\ &= [\hat{\mathbf{w}}(n + 1) - \hat{\mathbf{w}}(n)]^H[\hat{\mathbf{w}}(n + 1) - \hat{\mathbf{w}}(n)]\\ &= \sum_{k=0}^{M-1} |\hat{w}_k(n + 1) - \hat{w}_k(n)|^2\end{aligned} \tag{9.123}$$

[13]The stochastic gradient algorithm known as the normalized LMS algorithm was suggested independently by Nagumo and Noda (1967) and Albert and Gardner (1967). Nagumo and Noda did not use any special name for the algorithm, whereas Albert and Gardner referred to it as a "quick and dirty regression" scheme. It appears that Bitmead and Anderson (1980) coined the name "normalized LMS algorithm."

[14]For a discussion of the method of Lagrange multipliers, see Appendix C.

Define the tap weight $\hat{w}_k(n)$ for $k = 0, 1, \ldots, M-1$ in terms of its real and imaginary parts by writing

$$\hat{w}_k(n) = a_k(n) + jb_k(n), \qquad k = 0, 1, \ldots, M-1 \tag{9.124}$$

We then have

$$\|\delta\hat{\mathbf{w}}(n+1)\|^2 = \sum_{k=0}^{M-1} \left([a_k(n+1) - a_k(n)]^2 + [b_k(n+1) - b_k(n)]^2\right) \tag{9.125}$$

Let the tap input $u(n-k)$ and the desired response $d(n)$ be defined in terms of their respective real and imaginary parts as follows:

$$d(n) = d_1(n) + jd_2(n) \tag{9.126}$$

$$u(n-k) = u_1(n-k) + ju_2(n-k) \tag{9.127}$$

Accordingly, we may rewrite the complex constraint of Eq. (9.122) as an equivalent pair of real constraints:

$$\sum_{k=0}^{M-1} \left(a_k(n+1)u_1(n-k) + b_k(n+1)u_2(n-k)\right) = d_1(n) \tag{9.128}$$

and

$$\sum_{k=0}^{M-1} \left(a_k(n+1)u_2(n-k) - b_k(n+1)u_1(n-k)\right) = d_2(n) \tag{9.129}$$

We are now ready to formulate a real-valued cost function $J(n)$ for the constrained optimization problem at hand. In particular, we combine Eqs. (9.125), (9.128), and (9.129) into a single relation:

$$\begin{aligned} J(n) = &\sum_{k=0}^{M-1} \left([a_k(n+1) - a_k(n)]^2 + [b_k(n+1) - b_k(n)]^2\right) \\ &+ \lambda_1\left[d_1(n) - \sum_{k=0}^{M-1} \left(a_k(n+1)u_1(n-k) + b_k(n+1)u_2(n-k)\right)\right] \\ &+ \lambda_2\left[d_2(n) - \sum_{k=0}^{M-1} \left(a_k(n+1)u_2(n-k) - b_k(n+1)u_1(n-k)\right)\right] \end{aligned} \tag{9.130}$$

where λ_1 and λ_2 are *Lagrange multipliers.* To find the optimum values of $a_k(n+1)$ and $b_k(n+1)$, we differentiate the cost function $J(n)$ with respect to these two parameters and then set the results equal to zero. Hence, the use of Eq. (9.130) in the equation

$$\frac{\partial J(n)}{\partial a_k(n+1)} = 0$$

yields the result

$$2[a_k(n+1) - a_k(n)] - \lambda_1 u_1(n-k) - \lambda_2 u_2(n-k) = 0 \tag{9.131}$$

Similarly, the use of Eq. (9.130) in the complementary equation

$$\frac{\partial J(n)}{\partial b_k(n+1)} = 0$$

yields the complementary result

$$2[b_k(n+1) - b_k(n)] - \lambda_1 u_2(n-k) + \lambda_2 u_1(n-k) = 0 \tag{9.132}$$

Next, we use the definitions of Eqs. (9.124) and (9.127) to combine these two real results into a single complex one, as shown by

$$2[\hat{w}_k(n+1) - \hat{w}_k(n)] = \lambda^* u(n-k), \qquad k = 0, 1, \ldots, M-1 \tag{9.133}$$

where λ *is a* complex Lagrange multiplier:

$$\lambda = \lambda_1 + j\lambda_2 \tag{9.134}$$

To solve for the unknown λ^*, we multiply both sides of Eq. (9.133) by $u^*(n-k)$ and then sum over all possible integer values of k for 0 to $M-1$. We thus get

$$\begin{aligned}\lambda^* &= \frac{2}{\sum_{k=0}^{M-1} |u(n-k)|^2}\left[\sum_{k=0}^{M-1} \hat{w}_k(n+1)u^*(n-k) - \sum_{k=0}^{M-1} \hat{w}_k(n)u^*(n-k)\right] \\ &= \frac{2}{\|\mathbf{u}(n)\|^2}[\hat{\mathbf{w}}^T(n+1)\mathbf{u}^*(n) - \hat{\mathbf{w}}^T(n)\mathbf{u}^*(n)]\end{aligned} \tag{9.135}$$

where $\|\mathbf{u}(n)\|$ is the Euclidean norm of the tap-input vector $\mathbf{u}(n)$. Next, we use the complex constraint of Eq. (9.122) in (9.135) and thus formulate λ^* as follows:

$$\lambda^* = \frac{2}{\|\mathbf{u}(n)\|^2}[d^*(n) - \hat{\mathbf{w}}^T(n)\mathbf{u}^*(n)] \tag{9.136}$$

However, from the definition of the estimation error $e(n)$, we have

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

Accordingly, we may further simplify the expression given in Eq. (9.136) and thus write

$$\lambda^* = \frac{2}{\|\mathbf{u}(n)\|^2} e^*(n) \tag{9.137}$$

Finally, we substitute Eq. (9.137) into (9.133), obtaining

$$\begin{aligned}\delta\hat{w}_k(n+1) &= \hat{w}_k(n+1) - \hat{w}_k(n) \\ &= \frac{1}{\|\mathbf{u}(n)\|^2} u(n-k)e^*(n), \qquad k = 0, 1, \ldots, M-1\end{aligned} \tag{9.138}$$

In vector form, we may equivalently write

$$\delta\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n+1) - \hat{\mathbf{w}}(n)$$
$$= \frac{1}{\|\mathbf{u}(n)\|^2}\,\mathbf{u}(n)e^*(n) \tag{9.139}$$

In order to exercise control over the change in the tap-weight vector from one iteration to the next without changing its direction, we introduce a positive real scaling factor denoted by $\tilde{\mu}$. That is, we redefine the change $\delta\hat{\mathbf{w}}(n+1)$ simply as

$$\delta\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n+1) - \hat{\mathbf{w}}(n)$$
$$= \frac{\tilde{\mu}}{\|\mathbf{u}(n)\|^2}\,\mathbf{u}(n)e^*(n) \tag{9.140}$$

Equivalently, we may write

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \frac{\tilde{\mu}}{\|\mathbf{u}(n)\|^2}\,\mathbf{u}(n)e^*(n) \tag{9.141}$$

Indeed, this is the desired recursion for computing the M-by-1 tap-weight vector in the normalized LMS algorithm.

Equation (9.141) clearly shows the reason for using the term "normalized." In particular, we see that the product vector $\mathbf{u}(n)e^*(n)$ is normalized with respect to the squared Euclidean norm of the tap-input vector $\mathbf{u}(n)$.

The important point to note from the analysis presented above is that given new input data (at time n) represented by the tap-input vector $\mathbf{u}(n)$ and desired response $d(n)$, the normalized LMS algorithm updates the tap-weight vector in such a way that the value $\hat{\mathbf{w}}(n+1)$ computed at time $n+1$ exhibits the *minimum change* (in a Euclidean norm sense) *with respect to the known value* $\hat{\mathbf{w}}(n)$ at time n; for example, *no* charge may represent minimum change. Hence, the normalized LMS algorithm (and for that matter the conventional LMS algorithm) is a manifestation of the *principle of minimal disturbance* (Widrow and Lehr, 1990). The principle of minimal disturbance states that, *in the light of new input data, the parameters of an adaptive system should only be disturbed in a minimal fashion.*

Moreover, comparing the recursion of Eq. (9.141) for the normalized LMS algorithm with that of Eq. (9.8) for the conventional LMS algorithm, we may make the following observations:

- The adaptation constant $\tilde{\mu}$ for the normalized LMS algorithm is *dimensionless*, whereas the adaptation constant μ for the LMS algorithm has the dimensions of *inverse power*.
- Setting

$$\mu(n) = \frac{\tilde{\mu}}{\|\mathbf{u}(n)\|^2} \tag{9.142}$$

TABLE 9.2 SUMMARY OF THE NORMALIZED LMS ALGORITHM

Parameters: M = number of taps
$\tilde{\mu}$ = adaptation constant
$0 < \tilde{\mu} < 2$
a = positive constant

Initialization. If prior knowledge on the tap-weight vector $\hat{\mathbf{w}}(n)$ is available, use it to select an appropriate value for $\hat{\mathbf{w}}(0)$. Otherwise, set $\hat{\mathbf{w}}(0) = \mathbf{0}$.

Data
(a) Given: $\mathbf{u}(n)$: M-by-1 tap input vector at time n
$d(n)$: desired response at time n

(b) To be computed: $\hat{\mathbf{w}}(n+1)$ = estimate of tap-weight vector at time $n+1$
Computation: $n = 0, 1, 2, \ldots$

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

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \frac{\tilde{\mu}}{a + \|\mathbf{u}(n)\|^2}\mathbf{u}(n)e^*(n)$$

we may view the normalized LMS algorithm as an LMS algorithm with a *time-varying step-size parameter.*

- The normalized LMS algorithm is *convergent in the mean square* if the adaptation constant $\tilde{\mu}$ satisfies the following condition (Weiss and Mitra, 1979; Hsia, 1983):

$$0 < \tilde{\mu} < 2 \tag{9.143}$$

Most importantly, the normalized LMS algorithm exhibits a rate of convergence that is potentially faster than that of the standard LMS algorithm for both uncorrelated and correlated input data (Nagumo and Noda, 1967; Douglas and Meng, 1994). Another point of interest is that in overcoming the gradient noise amplification problem associated with the LMS algorithm, the normalized LMS algorithm introduces a problem of its own. Specifically, when the tap-input vector $\mathbf{u}(n)$ is small, numerical difficulties may arise because then we have to divide by a small value for the squared norm $\|\mathbf{u}(n)\|^2$. To overcome this problem, we slightly modify the recursion of Eq. (9.141) as follows:

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \frac{\tilde{\mu}}{a + \|\mathbf{u}(n)\|^2}\mathbf{u}(n)e^*(n) \tag{9.144}$$

where $a > 0$, and as before $0 < \tilde{\mu} < 2$. For $a = 0$, Eq. (9.144) reduces to the previous form given in Eq. (9.141). The normalized LMS algorithm is summarized in Table 9.2.

9.12 SUMMARY AND DISCUSSION

In this rather long chapter, we have presented a detailed study of the least-mean-square (LMS) algorithm, which represents the workhorse of linear adaptive filtering. The practical importance of the LMS algorithm is largely due to two unique attributes:

- Simplicity of implementation
- Model-independent and therefore robust performance

The main limitation of the LMS algorithm is its relatively slow rate of convergence.

Two principal factors affect the convergence behavior of the LMS algorithm: the step-size parameter μ, and the eigenvalues of the correlation matrix $\mathbf{R}$ of the tap-input vector. In light of the analysis of the LMS algorithm, using the independence theory, their individual effects may be summarized as follows:

1. Convergence of the LMS algorithm in the mean square is assured by choosing the step-size parameter μ in accordance with the practical condition:

$$0 < \mu < \frac{2}{\text{tap-input power}}$$

   where the tap-input power is the sum of the mean-square values of all the tap inputs in the transversal filter.
2. When a small value is assigned to μ, the adaptation is slow, which is equivalent to the LMS algorithm having a long "memory." Correspondingly, the excess mean-squared error after adaptation is small, on the average, because of the large amount of data used by the algorithm to estimate the gradient vector. On the other hand, when μ is large, the adaptation is relatively fast, but at the expense of an increase in the average excess mean-squared error after adaptation. In this case, less data enter the estimation, hence a degraded estimation error performance. Thus, the reciprocal of the parameter μ may be viewed as the *memory* of the LMS algorithm.
3. When the eigenvalues of the correlation matrix $\mathbf{R}$ are widely spread, the excess mean-squared error produced by the LMS algorithm is primarily determined by the largest eigenvalues, and the time taken by the average tap-weight vector $E[\hat{\mathbf{w}}(n)]$ to converge is limited by the smallest eigenvalues. However, the speed of convergence of the mean-squared error, $J(n)$, is affected by a spread of the eigenvalues of $\mathbf{R}$ to a lesser extent than the convergence of $E[\hat{\mathbf{w}}(n)]$. When the eigenvalue spread is large (i.e., when the correlation matrix of the tap inputs is ill conditioned), the convergence of the LMS algorithm may slow down. However, this need not always be so, as the convergence behavior of the LMS algorithm takes on a directional nature under non-white inputs. This property may indeed be exploited in initializing the LMS algorithm, thereby improving the convergence process; for this to be possible, prior knowledge is required.

A basic limitation of the independence theory is the fact that it ignores the statistical dependence between the "gradient" directions as the algorithm proceeds from one iteration to the next. Several worthwhile results have been obtained in the literature on the practical case of *statistically dependent inputs*. In this regard, special mention should be made of the papers by Mazo (1979), Farden (1981), Jones et al. (1982), Macchi and Eweda (1984), and Gardner (1984), and the book by Macchi (1995).

Sethares (1993) presents a detailed discussion of the convergence behavior of the LMS algorithm using two other approaches: the stochastic approximation approach and the deterministic approach. In the *stochastic approximation approach*, developed independently by Ljung (1977) and Kushner and Clark (1978), the discrete-time evolution of the parameter estimation errors of the LMS algorithm is related to the behavior of an unforced deterministic ordinary differential equation; in particular, it is shown that stability of the ordinary differential equation so derived implies convergence of the LMS algorithm. In the *deterministic approach*, the basic update equation of the LMS algorithm [i.e., Eq. (9.8)] is interpreted as the state equation of a nonlinear, time-varying system; the system is then linearized and averaged to derive the operating conditions for which the LMS algorithm may be expected to succeed in its linear adaptive filtering task.

In yet another approach described in Butterweck (1994), a steady-state analysis of the LMS algorithm is presented, which relies on the use of a power series solution for the weight-error vector $\boldsymbol{\epsilon}(n)$ without invoking the independence assumption. The essence of this latter approach is described in Appendix I.

One last comment is in order. In the study of digital filters, frequency-domain performance measures play a central role alongside their time-domain counterparts. Yet the convergence analysis of the LMS algorithm presented in this chapter (and for that matter, in much of the literature on adaptive filters) has been confined to the time domain. The paper by Johnson et al. (1994) attempts to redress this imbalance by presenting a fundamental re-evaluation of adaptive filter performance in frequency-domain terms, and uses an adaptive equalizer as an illustrative example. An interesting point that emerges from the study presented therein is that there is a correspondence between (a) the rates at which the different bands in the equalizer's frequency response adapt toward their steady-state values, and (b) the way in which the eigenfilters are grouped according to the eigenvalues of the correlation matrix of the channel output (i.e., the equalizer's input). Another noteworthy paper on the frequency-domain analysis of linear adaptive filters is that of Gunnarsson and Ljung (1989). The focal point of this latter paper is the formulation of a performance measure in terms of the mean-square error between the true (momentary) transfer function and the one being estimated by the adaptive filter. The evaluation is done in the context of tracking a linear time-variant system, which is the subject matter of Chapter 16.

PROBLEMS

1. The LMS algorithm is used to implement a dual-input, single-weight adaptive noise canceler. Set up the equations that define the operation of this algorithm.

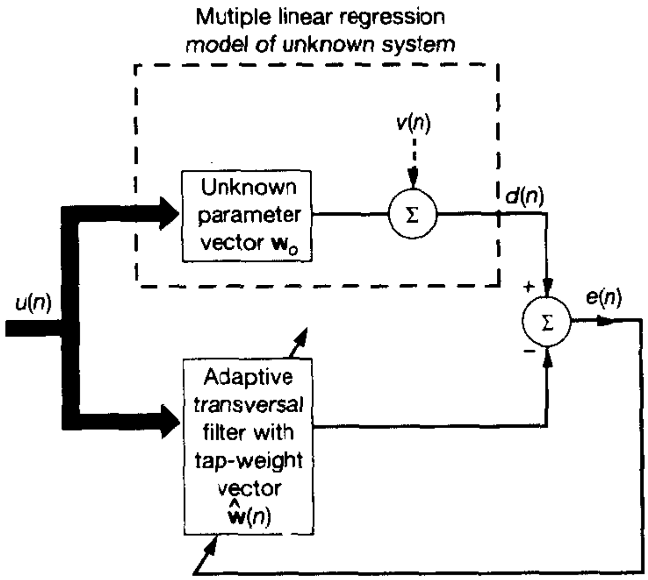


Figure P9.1

2. The LMS-based adaptive deconvolution procedure discussed in Example 2 of Section 9.3 applies to forward-time adaptation (i.e., forward prediction). Reformulate this procedure for reverse-time adaptation (i.e., backward prediction).

3. The zero-mean output $d(n)$ of an unknown real-valued system is represented by the *multiple linear regression model*

$$d(n) = \mathbf{w}_o^T\mathbf{u}(n) + \nu(n)$$

where $\mathbf{w}_o$ is the (unknown) parameter vector of the model, $\mathbf{u}(n)$ is the input vector (regressor), and $\nu(n)$ is the sample value of an immeasurable white-noise process of zero mean and variance σ_ν^2. The block diagram of Fig. P9.1 shows the adaptive modeling of the unknown system, in which the adaptive transversal filter is controlled by a *modified* version of the LMS algorithm. In particular, the tap-weight vector $\mathbf{w}(n)$ of the transversal filter is chosen to minimize the index of performance

$$J(\mathbf{w}, K) = E[e^{2K}(n)]$$

for $K = 1, 2, 3, \ldots$

(a) By using the instantaneous gradient vector, show that the new adaptation rule for the corresponding estimate of the tap-weight vector is

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \mu K\mathbf{u}(n)e^{2K-1}(n)$$

where μ is the step-size parameter, and $e(n)$ is the estimation error

$$e(n) = d(n) - \mathbf{w}^T(n)\mathbf{u}(n)$$

(b) Assume that the weight-error vector

$$\boldsymbol{\epsilon}(n) = \hat{\mathbf{w}}(n) - \mathbf{w}_o$$

is close to zero, and that $v(n)$ is independent of $\mathbf{u}(n)$. Hence, show that

$$E[\boldsymbol{\epsilon}(n+1)] = (\mathbf{I} - \mu K(2K-1)E[v^{2K-2}(n)]\mathbf{R}E[\boldsymbol{\epsilon}(n)]$$

where $\mathbf{R}$ is the correlation matrix of the input vector $\mathbf{u}(n)$.

(c) Show that the modified LMS algorithm described in part (a) converges in the mean value if the step-size parameter μ satisfies the condition

$$0 < \mu < \frac{2}{K(2K-1)E[v^{2(K-1)}(n)]\lambda_{\max}}$$

where $\lambda_{\max}$ is the largest eigenvalue of matrix $\mathbf{R}$.

(d) For $K = 1$, show that the results given in parts (a), (b), and (c) reduce to those in the conventional LMS algorithm.

4. (a) Let $\mathbf{m}(n)$ denote the *mean weight vector* in the LMS algorithm at iteration n; that is

$$\mathbf{m}(n) = E[\hat{\mathbf{w}}(n)]$$

Using the independence assumption of Section 9.4, show that

$$\mathbf{m}(n) = (\mathbf{I} - \mu\mathbf{R})^n[\mathbf{m}(0) - \mathbf{m}(\infty)] + \mathbf{m}(\infty)$$

where μ is the step-size parameter, $\mathbf{R}$ is the correlation matrix of the input vector, and $\mathbf{m}(0)$ and $\mathbf{m}(\infty)$ are the initial and final values of the mean weight vector, respectively.

(b) Hence, show that for convergence of the mean value $\mathbf{m}(n)$, the step-size parameter μ must satisfy the condition

$$0 < \mu < \frac{2}{\lambda_{\max}}$$

where $\lambda_{\max}$ is the largest eigenvalue of the correlation matrix $\mathbf{R}$.

5. Consider the product $x_1x_2^*x_3x_4^*$, where x_1, x_2, x_3, and x_4 are complex Gaussian random variables. The Gaussian mean factoring theorem states that (see Chapter 2)

$$E[x_1x_2^*x_3x_4^*] = E[x_1x_2^*]E[x_3x_4^*] + E[x_1x_4^*]E[x_3x_2^*]$$

Using this theorem, show that

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

6. Consider the use of a white noise sequence of zero mean and variance σ^2 as the input to the LMS algorithm. Evaluate the following:

(a) Condition for convergence of the algorithm in the mean square.

(b) The excess mean-squared error.

7. *The leaky LMS algorithm.* Consider the time-varying cost function

$$J(n) = |e(n)|^2 + \alpha\|\mathbf{w}(n)\|^2$$

where $\mathbf{w}(n)$ is the tap-weight vector of a transversal filter, $e(n)$ is the estimation error, and α is a constant. As usual, $e(n)$ is defined by

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

where $d(n)$ is the desired response, and $\mathbf{u}(n)$ is the tap-input vector. In the *leaky LMS algorithm*, the cost function $J(n)$ is minimized with respect to the weight vector $\mathbf{w}(n)$.

(a) Show that the time update for the tap-weight vector $\hat{\mathbf{w}}(n)$ is defined by

$$\hat{\mathbf{w}}(n+1) = (1 - \mu\alpha)\hat{\mathbf{w}}(n) + \mu\mathbf{u}(n)e^*(n)$$

(b) Using the independence theory, show that

$$\lim_{n \to \infty} E[\hat{\mathbf{w}}(n)] = (\mathbf{R} + \alpha\mathbf{I})^{-1}\mathbf{p}$$

where $\mathbf{R}$ is the correlation matrix of the tap inputs and $\mathbf{p}$ is the cross-correlation vector between the tap inputs and the desired response. What is the condition for the algorithm to converge in the mean value?

(c) How would you modify the tap-input vector in the conventional LMS algorithm to get the equivalent result described in part (a)?

8. Consider the operation of an adaptive line enhancer using the LMS algorithm under a low signal-to-noise ratio condition. The correlation matrix of the input vector is defined by

$$\mathbf{R} = \sigma_\nu^2\mathbf{I}$$

where $\mathbf{I}$ is the identity matrix. Show that the steady-state value of the weight-error correlation matrix $\mathbf{K}(n)$ is given by

$$\mathbf{K}(\infty) \simeq \frac{\mu}{2} J_{\min} \mathbf{I}$$

where μ is the step-size parameter, and $J_{\min}$ is the minimum mean-squared error. You may assume that the number of taps in the adaptive transversal filter is large.

9. (a) The LMS algorithm is usually referred to as a stochastic gradient algorithm. Yet, in examining Figs. 9.27 and 9.28 of Example 7 involving a purely sinusoidal process, the trajectories displayed therein are all well defined (i.e., the parameter estimates produced by the LMS algorithm are deterministic). Both of these statements are valid in their own ways; how do you reconcile them?

(b) Suppose a white noise process of zero mean and various σ_ν^2 is added to the sinusoidal process considered in Example 7. How would the results of that example be modified?

10. The convergence ratio, $\mathscr{C}(n)$, of an adaptive algorithm is defined by

$$\mathscr{C}(n) = \frac{E[\|\boldsymbol{\epsilon}(n+1)\|^2]}{E[\|\boldsymbol{\epsilon}(n)\|^2]}$$

Show that, for small n, the convergence ratio of the LMS algorithm for stationary inputs approximately equals

$$\mathscr{C}(n) \simeq (1 - \mu\sigma_u^2)^2 \qquad n \text{ small}$$

Here it is assumed that the correlation matrix of the tap-input vector $\mathbf{u}(n)$ is approximately equal to $\sigma_u^2\,\mathbf{I}$.

11. Starting from Eq. (9.114) and using the update recursion of Eq. (9.115) for the LMS algorithm, verify the validity of the inequality described in Eq. (9.116).

12. Expanding on the result described in (9.120), show that the LMS algorithm also satisfies the bound (Sayed and Rupp, 1994)

$$\frac{\mu^{-1}\|\hat{\mathbf{w}}(n+1)-\mathbf{w}_o\|^2+\sum_{i=0}^{n}|\hat{\mathbf{w}}^H(i)\mathbf{u}(i)-\mathbf{w}_o^H\mathbf{u}(i)|^2}{\mu^{-1}\|\hat{\mathbf{w}}(0)-\mathbf{w}_o\|^2+\sum_{i=1}^{n}|\nu(i)|^2}\leq 1$$

In light of this result, what can you say about the operator that maps $\{\mu^{-1/2}(\hat{\mathbf{w}}(0)-\mathbf{w}_o), \nu(0), \nu(1), \ldots, \nu(n)\}$ to the sequence of errors $\{\mu^{-1/2}(\hat{\mathbf{w}}(n+1)-\mathbf{w}_o), \hat{\mathbf{w}}^H(i)\mathbf{u}(i)-\mathbf{w}_o{}^H\mathbf{u}(i) \mid i=0, 1, \ldots, n\}$?

13. The *normalized LMS algorithm* is described by the following recursion for the tap-weight vector:

$$\hat{\mathbf{w}}(n+1)=\hat{\mathbf{w}}(n)+\frac{\tilde{\mu}}{\|\mathbf{u}(n)\|^2}\mathbf{u}(n)e^*(n)$$

where $\tilde{\mu}$ is a positive constant, and $\|\mathbf{u}(n)\|$ is the *norm* of the tap-input vector. The estimation error $e(n)$ is defined by

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

where $d(n)$ is the desired response.

Using the independence theory, show that the necessary and sufficient condition for the normalized LMS algorithm to be convergent in the mean square is $0<\tilde{\mu}<2$. *Hint*: You may use the following approximation:

$$E\left[\frac{\mathbf{u}(n)\mathbf{u}^H(n)}{\|\mathbf{u}(n)\|^2}\right]\simeq\frac{E[\mathbf{u}(n)\mathbf{u}^H(n)]}{E[\|\mathbf{u}(n)\|^2]}$$

14. In Section 9.11 we presented a derivation of the normalized LMS algorithm in its own right. In this problem we explore another derivation of this algorithm by modifying the method of steepest descent that led to the development of the conventional LMS algorithm. The modification involves writing the tap-weight vector update in the method of steepest descent as follows:

$$\mathbf{w}(n+1)=\mathbf{w}(n)-\frac{1}{2}\mu(n)\nabla(n)$$

where $\mu(n)$ is a *time-varying step-size parameter*, and $\nabla(n)$ is the gradient vector defined by

$$\nabla(n)=2[\mathbf{R}\mathbf{w}(n)-\mathbf{p}]$$

where $\mathbf{R}$ is the correlation matrix of the tap-input vector $\mathbf{u}(n)$, and $\mathbf{p}$ is the cross-correlation vector between the tap-input vector $\mathbf{u}(n)$ and the desired response $d(n)$.

(a) At time $n+1$, the mean-squared error is defined by

$$J(n+1)=E[|e(n+1)|^2]$$

where

$$e(n+1)=d(n+1)-\mathbf{w}^H(n+1)\mathbf{u}(n+1)$$

Determine the value of the step-size parameter $\mu_o(n)$ that minimizes $J(n + 1)$ as a function of $\mathbf{R}$ and $\nabla(n)$.

(b) Using instantaneous estimates for $\mathbf{R}$ and $\nabla(n)$ in the expression for $\mu_o(n)$ derived in part (a), determine the corresponding instantaneous estimate for $\mu_o(n)$. Hence, formulate the update equation for the tap-weight vector $\hat{\mathbf{w}}(n)$, and compare your result with that obtained for the normalized LMS algorithm.

15. When conducting a computer experiment that involves the generation of an AR process, sometimes not enough time is allowed for the transients to die out. The purpose of this experiment is to evaluate the effects of such transients on the operation of the LMS algorithm. Consider then the AR process $u(n)$ of order 1 described in Section 9.6. The parameters of this process are as follows:

AR parameter:	$a = 0.99$
AR process variance:	$\sigma_u^2 = 1.00$
Noise variance:	$\sigma_v^2 = 0.02$

Generate the process $u(n)$ so described for $1 \le n \le 100$, assuming zero initial conditions. Use the process $u(n)$ as the input of a linear adaptive predictor that is based on the LMS algorithm using a step-size parameter $\mu = 0.05$. In particular, plot the learning curve of the predictor by ensemble averaging over 100 independent realizations of the squared value of its output versus time n for $1 \le n \le 100$. Unlike the normal operation of the LMS algorithm, the learning curve so computed should start at the origin, rise to a peak, and then decay toward a steady-state value. Explain the reasons for this phenomenon.