ADAPTIVE FILTERING 9

9.1 INTRODUCTION

In the previous five chapters, we have considered a variety of different problems including signal modeling, Wiener filtering, and spectrum estimation. In each case, we made an important assumption that the signals that were being analyzed were stationary. In signal modeling, for example, it was assumed that the signal to be modeled could be approximated as the output of a linear *shift-invariant* filter $h(n)$ with an input that is either a unit sample, in the case of deterministic signal modeling, or stationary white noise, in the case of stochastic signal modeling. In Chapter 7, we looked at the problem of designing a linear *shift-invariant* filter that would produce the minimum mean-square estimate of a wide-sense stationary process $d(n)$ and in Chapter 8 we considered the problem of estimating the power spectrum $P_x(e^{j\omega})$ of a wide-sense stationary process $x(n)$. Unfortunately, since the signals that arise in almost every application will be nonstationary, the approaches and techniques that we have been considering thus far would not be appropriate. One way to circumvent this difficulty would be to process these nonstationary processes in blocks, over intervals for which the process may be assumed to be approximately stationary. This approach, however, is limited in its effectiveness for several reasons. First, for rapidly varying processes, the interval over which a process may be assumed to be stationary may be too small to allow for sufficient accuracy or resolution in the estimation of the relevant parameters. Second, this approach would not easily accommodate step changes within the analysis intervals. Third, and perhaps most important, this solution imposes an incorrect model on the data, i.e., piecewise stationary. Therefore, a better approach would be start over and begin with a nonstationarity assumption at the outset.

In order to motivate the approach that we will be considering in this chapter, let us reconsider the Wiener filtering problem within the context of nonstationary processes. Specifically, let $w(n)$ denote the unit sample response of the FIR Wiener filter that produces the minimum mean-square estimate of a desired process $d(n)$,

$$\hat{d}(n) = \sum_{k=0}^{p} w(k)x(n-k) \tag{9.1}$$

As we saw in Chap. 7, if $x(n)$ and $d(n)$ are jointly wide-sense stationary processes, with $e(n) = d(n) - \hat{d}(n)$, then the filter coefficients that minimize the mean-square error

$E\{|e(n)|^2\}$ are found by solving the Wiener-Hopf equations

$$\mathbf{R}_x \mathbf{w} = \mathbf{r}_{dx} \tag{9.2}$$

However, if $d(n)$ and $x(n)$ are nonstationary, then the filter coefficients that minimize $E\{|e(n)|^2\}$ will depend on n, and the filter will be shift-varying, i.e.,

$$\hat{d}(n) = \sum_{k=0}^{p} w_n(k)x(n-k) \tag{9.3}$$

where $w_n(k)$ is the value of the kth filter coefficient at time n. Using vector notation, this estimate may be expressed as

$$\hat{d}(n) = \mathbf{w}_n^T \mathbf{x}(n)$$

where

$$\mathbf{w}_n = \Big[w_n(0),\ w_n(1),\ \ldots,\ w_n(p)\Big]^T$$

is the vector of filter coefficients at time n, and

$$\mathbf{x}(n) = \Big[x(n),\ x(n-1),\ \ldots,\ x(n-p)\Big]^T$$

In many respects, the design of a shift-varying (adaptive) filter is much more difficult than the design of a (shift-invariant) Wiener filter since, for each value of n, it is necessary to find the set of optimum filter coefficients, $w_n(k)$, for $k = 0, 1, \ldots, p$. However, the problem may be simplified considerably if we relax the requirement that $\mathbf{w}_n$ minimize the mean-square error at each time n and consider, instead, a coefficient update equation of the form

$$\boxed{\mathbf{w}_{n+1} = \mathbf{w}_n + \Delta\mathbf{w}_n} \tag{9.4}$$

where $\Delta\mathbf{w}_n$ is a *correction* that is applied to the filter coefficients $\mathbf{w}_n$ at time n to form a new set of coefficients, $\mathbf{w}_{n+1}$, at time $n+1$. This update equation is the heart of the adaptive filters that we will be designing in this chapter. As illustrated in Fig. 9.1, the design of an adaptive filter involves defining how this correction is to be formed. Even for stationary processes, there are several reasons why we may prefer to implement a time-invariant Wiener filter using Eq. (9.4). First, if the order of the filter p is large, then it may be difficult or impractical to solve the Wiener-Hopf equations directly. Second, if $\mathbf{R}_x$ is ill-conditioned (almost singular) then the solution to the Wiener-Hopf equations will be numerically sensitive to round-off errors and finite precision effects. Finally, and perhaps most importantly, is the fact that solving the Wiener-Hopf equations requires that the autocorrelation $r_x(k)$ and the cross-correlation $r_{dx}(k)$ be known. Since these ensemble averages are typically unknown, then it is necessary to estimate them from measurements of the processes. Although we could use estimates such as

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

$$\hat{r}_{dx}(k) = \frac{1}{N}\sum_{n=0}^{N-1} d(n)x^*(n-k) \tag{9.5}$$

doing so would result in a delay of N samples. Even more importantly, in an environment for which the ensemble averages are changing in time, these estimates would need to be updated continuously.

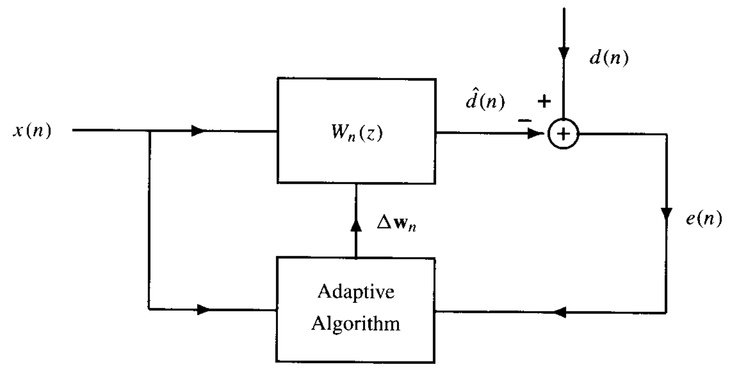


Figure 9.1 *Block diagram of an adaptive filter consisting of a shift-varying filter $W_n(z)$ and an adaptive algorithm for updating the filter coefficients $w_n(k)$.*

The key component of an adaptive filter is the set of rules, or algorithm, that defines how the correction $\Delta\mathbf{w}_n$ is to be formed. Although it is not yet clear what this correction should be, what is clear is that the sequence of corrections should decrease the mean-square error. In fact, whatever algorithm is used, the adaptive filter should have the following properties:

1. In a stationary environment, the adaptive filter should produce a sequence of corrections $\Delta\mathbf{w}_n$ in such a way that $\mathbf{w}_n$ converges to the solution to the Wiener-Hopf equations,

$$\lim_{n\to\infty} \mathbf{w}_n = \mathbf{R}_x^{-1}\mathbf{r}_{dx}$$

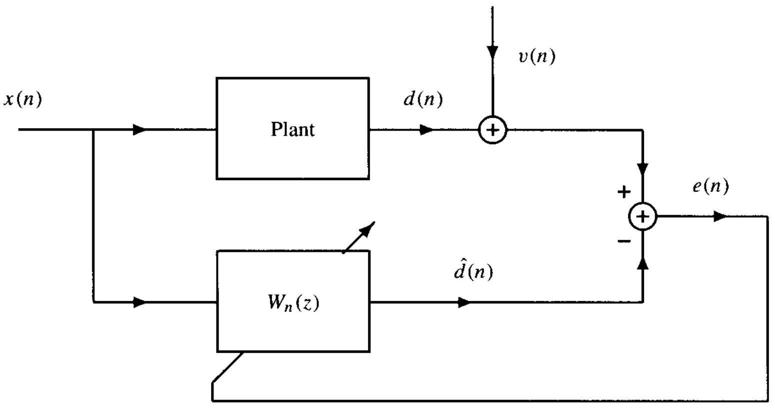
2. It should not be necessary to know the signal statistics $r_x(k)$ and $r_{dx}(k)$ in order to compute $\Delta\mathbf{w}_n$. The estimation of these statistics should be "built into" the adaptive filter.

3. For nonstationary signals, the filter should be able to adapt to the changing statistics and "track" the solution as it evolves in time.

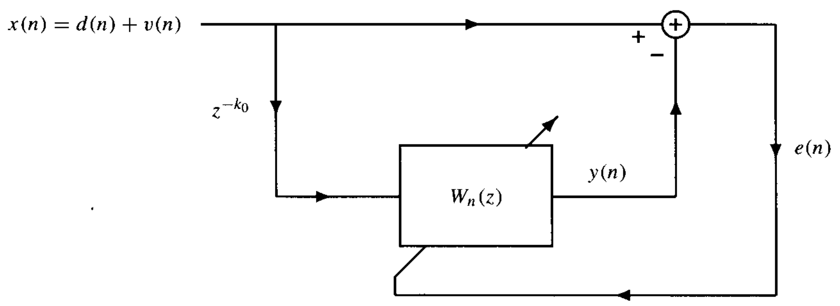
An important issue in the implementation of an adaptive filter that is apparent from Fig. 9.1 is the requirement that the error signal, $e(n)$, be available to the adaptive algorithm. Since it is $e(n)$ that allows the filter to *measure* its performance and determine how the filter coefficients should be modified, without $e(n)$ the filter would not be able to adapt. In some applications, acquiring $d(n)$ is straightforward, which makes the evaluation of $e(n)$ easy. Consider, for example, the problem of system identification shown in Fig. 9.2*a* in which a plant produces an output $d(n)$ in response to a known input $x(n)$. The goal is to develop a model for the plant, $W_n(z)$, that produces a response $\hat{d}(n)$ that is as close as possible to $d(n)$. To account for observation noise in the measurement of the plant output, an additive noise source $v(n)$ is shown in the figure. In other applications, the acquisition of $d(n)$ is not as straightforward and some ingenuity is required. For example, consider the problem of interference cancellation in which a signal $d(n)$ is observed in the presence of an interfering signal $v(n)$,

$$x(n) = d(n) + v(n)$$

Since $d(n)$ is unknown, the error sequence cannot be generated directly. In some circumstances, however, the system shown in Fig. 9.2*b* will generate a sequence that may be used by the adaptive filter to estimate $d(n)$. Specifically, suppose that $d(n)$ and $v(n)$ are



(a) System identification



(b) Noise cancellation

Figure 9.2 *Two adaptive filtering applications that illustrate how an error sequence, $e(n)$, may be generated.*

real-valued, uncorrelated zero mean processes. In addition, suppose that $d(n)$ is a narrowband process and that $v(n)$ is a broadband process with an autocorrelation sequence that is approximately zero for lags $k \geq k_0$. The mean-square error that is formed by taking the difference between $x(n)$ and the output of the adaptive filter, $y(n)$, may be expressed as follows:

$$\begin{aligned} E\{e^2(n)\} &= E\Big\{\big[d(n) + v(n) - y(n)\big]^2\Big\} \\ &= E\Big\{v^2(n)\Big\} + E\Big\{\big[d(n) - y(n)\big]^2\Big\} + 2E\Big\{v(n)\big[d(n) - y(n)\big]\Big\} \end{aligned} \tag{9.6}$$

Since $v(n)$ and $d(n)$ are uncorrelated, then $E\{v(n)d(n)\} = 0$ and the last term becomes

$$2E\Big\{v(n)\big[d(n) - y(n)\big]\Big\} = -2E\Big\{v(n)y(n)\Big\}$$

In addition, since the input to the adaptive filter is $x(n - k_0)$, then the output $y(n)$ is

$$y(n) = \sum_{k=0}^{p} w_n(k)x(n - k_0 - k) = \sum_{k=0}^{p} w_n(k)\Big[d(n - k_0 - k) + v(n - k_0 - k)\Big]$$

Thus,

$$E\{v(n)y(n)\} = \sum_{k=0}^{p} w_n(k)\Big[E\{v(n)d(n-k_0-k)\} + E\{v(n)v(n-k_0-k)\}\Big]$$

Finally, since $v(n)$ is uncorrelated with $d(n)$ as well as with $v(n-k_0-k)$, then $E\{v(n)y(n)\} = 0$ and the mean-square error becomes

$$E\{e^2(n)\} = E\{v^2(n)\} + E\{[d(n) - y(n)]^2\}$$

Therefore, minimizing $E\{e^2(n)\}$ is equivalent to minimizing $E\{[d(n)-y(n)]^2\}$, the mean-square error between $d(n)$ and the output of the adaptive filter, $y(n)$. Thus, the output of the adaptive filter is the minimum mean-square estimate of $d(n)$.

In this chapter, we will consider a variety of different methods for designing and implementing adaptive filters. As we will see, the efficiency of the adaptive filter and its performance in estimating $d(n)$ will depend on a number of factors including the type of filter (FIR or IIR), the filter structure (direct form, parallel, lattice, etc.), and the way in which the performance measure is defined (mean-square error, least squares error). The organization of this chapter is as follows. In Section 9.2, we begin with the development of the FIR adaptive filters that are based on the method of steepest descent. Of primary interest will be the LMS adaptive filter. We will consider direct form as well as lattice filter structures. In Section 9.3, we will look briefly at the design of adaptive IIR filters. Being more difficult to characterize in terms of their properties and performance, the focus will be on the LMS adaptive recursive filter. Finally, in Section 9.4, the Recursive Least Squares (RLS) algorithm will be developed. There is a wide variety of applications in which adaptive filters have been successfully used such as linear prediction, echo cancellation, channel equalization, interference cancellation, adaptive notch filtering, adaptive control, system identification, and array processing. Therefore, as we progress through this chapter and look at specific adaptive filtering algorithms, we will briefly introduce some of these applications.

9.2 FIR ADAPTIVE FILTERS

In this section, we begin our study of adaptive filters by looking at the design of FIR (non-recursive) adaptive filters. In contrast to IIR or recursive adaptive filters, FIR filters are routinely used in adaptive filtering applications that range from adaptive equalizers in digital communication systems [26] to adaptive noise control systems [14]. There are several reasons for the popularity of FIR adaptive filters. First, stability is easily controlled by ensuring that the filter coefficients are bounded. Second, there are simple and efficient algorithms for adjusting the filter coefficients. Third, the performance of these algorithms is well understood in terms of their convergence and stability. Finally, FIR adaptive filters very often perform well enough to satisfy the design criteria.

An FIR adaptive filter for estimating a desired signal $d(n)$ from a related signal $x(n)$, as illustrated in Fig. 9.3, is

$$\hat{d}(n) = \sum_{k=0}^{p} w_n(k)x(n-k) = \mathbf{w}_n^T\mathbf{x}(n)$$

Here it is assumed that $x(n)$ and $d(n)$ are nonstationary random process and the goal is to

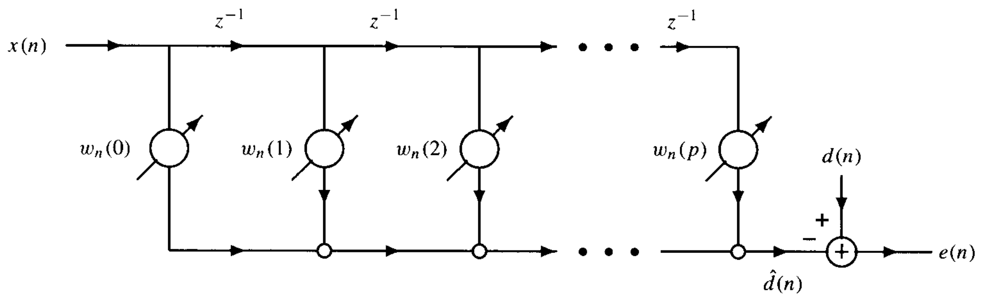


Figure 9.3 *A direct-form FIR adaptive filter.*

find the coefficient vector $\mathbf{w}_n$ at time n that minimizes the mean-square error,

$$\xi(n) = E\{|e(n)|^2\}$$

where

$$e(n) = d(n) - \hat{d}(n) = d(n) - \mathbf{w}_n^T\mathbf{x}(n) \tag{9.7}$$

As in the derivation of the FIR Wiener filter in Section 7.2, the solution to this minimization problem may be found by setting the derivative of $\xi(n)$ with respect to $w_n^*(k)$ equal to zero for $k = 0, 1, \ldots, p$. The result is

$$E\{e(n)x^*(n-k)\} = 0 \quad ; \quad k = 0, 1, \ldots, p \tag{9.8}$$

Substituting Eq. (9.7) into Eq. (9.8) we have

$$E\left\{\left[d(n) - \sum_{l=0}^{p} w_n(l)x(n-l)\right]x^*(n-k)\right\} = 0 \quad ; \quad k = 0, 1, \ldots, p$$

which, after rearranging terms, becomes

$$\sum_{l=0}^{p} w_n(l)E\left\{x(n-l)x^*(n-k)\right\} = E\left\{d(n)x^*(n-k)\right\} \quad ; \quad k = 0, 1, \ldots, p \tag{9.9}$$

Equation (9.9) is a set of $p+1$ linear equations in the $p+1$ unknowns $w_n(l)$. However, unlike the case of an FIR Wiener filter where it was assumed that $x(n)$ and $d(n)$ are jointly WSS, the solution to these equations depends on n. We may express these equations in vector form as follows:

$$\boxed{\mathbf{R}_x(n)\mathbf{w}_n = \mathbf{r}_{dx}(n)} \tag{9.10}$$

where

$$\mathbf{R}_x(n) = \begin{bmatrix} E\{x(n)x^*(n)\} & E\{x(n-1)x^*(n)\} & \cdots & E\{x(n-p)x^*(n)\} \\ E\{x(n)x^*(n-1)\} & E\{x(n-1)x^*(n-1)\} & \cdots & E\{x(n-p)x^*(n-1)\} \\ \vdots & \vdots & & \vdots \\ E\{x(n)x^*(n-p)\} & E\{x(n-1)x^*(n-p)\} & \cdots & E\{x(n-p)x^*(n-p)\} \end{bmatrix}$$

is a $(p+1) \times (p+1)$ Hermitian matrix of autocorrelations and

$$\mathbf{r}_{dx}(n) = \Big[E\{d(n)x^*(n)\},\ E\{d(n)x^*(n-1)\},\ \cdots,\ E\{d(n)x^*(n-p)\}\Big]^T \tag{9.11}$$

is a vector of cross-correlations between $d(n)$ and $x(n)$. Note that in the case of jointly WSS processes, Eq. (9.10) reduces to the Wiener-Hopf equations, and the solution $\mathbf{w}_n$ becomes independent of time. Instead of solving Eq. (9.10) for each value of n, which would be impractical in most real-time implementations, in the following section, we consider an iterative approach that is based on the method of steepest descent.

9.2.1 The Steepest Descent Adaptive Filter

In designing an FIR adaptive filter, the goal is to find the vector $\mathbf{w}_n$ at time n that minimizes the quadratic function

$$\xi(n) = E\{|e(n)|^2\}$$

Although the vector that minimizes $\xi(n)$ may be found by setting the derivatives of $\xi(n)$ with respect to $w^*(k)$ equal to zero, another approach is to *search* for the solution using the method of steepest descent. The *method of steepest descent* is an iterative procedure that has been used to find extrema of nonlinear functions since before the time of Newton. The basic idea of this method is as follows. Let $\mathbf{w}_n$ be an estimate of the vector that minimizes the mean-square error $\xi(n)$ at time n. At time $n+1$ a new estimate is formed by adding a correction to $\mathbf{w}_n$ that is designed to bring $\mathbf{w}_n$ closer to the desired solution. The correction involves taking a step of size μ in the direction of *maximum descent* down the quadratic error surface. For example, shown in Fig. 9.4*a* is a three-dimensional plot of a quadratic function of two real-valued coefficients, $w(0)$ and $w(1)$, given by[1]

$$\xi(n) = 6 - 6w(0) - 4w(1) + 6\big[w^2(0) + w^2(1)\big] + 6w(0)w(1) \qquad (9.12)$$

Note that the contours of constant error, when projected onto the $w(0)$-$w(1)$ plane, form a set of concentric ellipses. The direction of steepest descent at any point in the plane is the direction that a marble would take if it were placed on the inside of this quadratic *bowl*. Mathematically, this direction is given by the *gradient*, which is the vector of partial

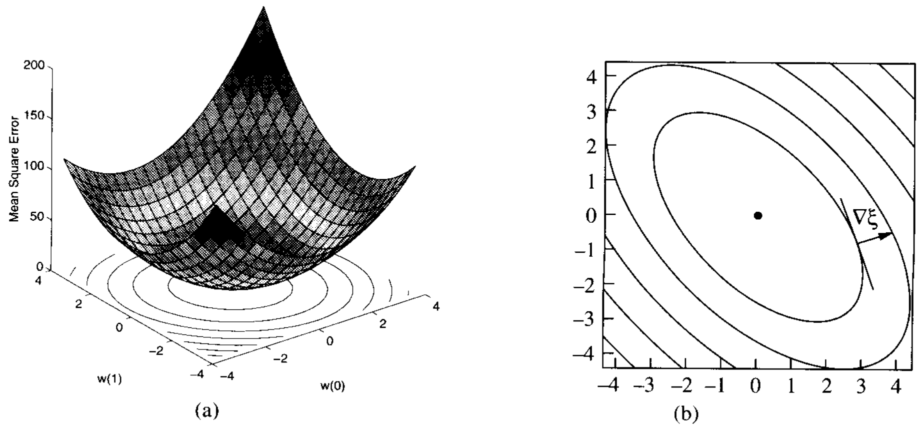


Figure 9.4 *(a) A quadratic function of two weights and (b) the contours of constant error. The gradient vector, which points in the direction of maximum increase in ξ, is orthogonal to the line that is tangent to the contour as illustrated in (b).*

[1] Although this error does not depend on n, we still denote it by $\xi(n)$.

derivatives of $\xi(n)$ with respect to the coefficients $w(k)$. For the quadratic function in Eq. (9.12), the gradient vector is

$$\nabla\xi(n) = \begin{bmatrix} \dfrac{\partial\xi(n)}{\partial w(0)} \\ \dfrac{\partial\xi(n)}{\partial w(1)} \end{bmatrix} = \begin{bmatrix} 12w(0) + 6w(1) - 6 \\ 12w(1) + 6w(0) - 4 \end{bmatrix}$$

As shown in Fig. 9.4*b*, for any vector **w**, the gradient is orthogonal to the line that is tangent to the contour of constant error at **w**. However, since the gradient vector points in the direction of *steepest ascent*, the direction of *steepest descent* points in the negative gradient direction. Thus, the update equation for $\mathbf{w}_n$ is

$$\boxed{\mathbf{w}_{n+1} = \mathbf{w}_n - \mu\nabla\xi(n)}$$

The step size μ affects the rate at which the weight vector moves down the quadratic surface and must be a positive number (a negative value for μ would move the weight vector up the quadratic surface in the direction of maximum ascent and would result in an increase in the error). For very small values of μ, the correction to $\mathbf{w}_n$ is small and the movement down the quadratic surface is slow and, as μ is increased, the rate of descent increases. However, there is an upper limit on how large the step size may be. For values of μ that exceed this limit, the trajectory of $\mathbf{w}_n$ becomes unstable and unbounded. The steepest descent algorithm may be summarized as follows:

1. Initialize the steepest descent algorithm with an initial estimate, $\mathbf{w}_0$, of the optimum weight vector **w**.
2. Evaluate the gradient of $\xi(n)$ at the current estimate, $\mathbf{w}_n$, of the optimum weight vector.
3. Update the estimate at time n by adding a correction that is formed by taking a step of size μ in the negative gradient direction

$$\mathbf{w}_{n+1} = \mathbf{w}_n - \mu\nabla\xi(n)$$

4. Go back to (2) and repeat the process.

Let us now evaluate the gradient vector $\nabla\xi(n)$. Assuming that **w** is complex, the gradient is the derivative of $E\{|e(n)|^2\}$ with respect to $\mathbf{w}^*$. With

$$\nabla\xi(n) = \nabla E\{|e(n)|^2\} = E\{\nabla|e(n)|^2\} = E\{e(n)\nabla e^*(n)\}$$

and

$$\nabla e^*(n) = -\mathbf{x}^*(n)$$

it follows that

$$\nabla\xi(n) = -E\{e(n)\mathbf{x}^*(n)\}$$

Thus, with a step size of μ, the steepest descent algorithm becomes

$$\boxed{\mathbf{w}_{n+1} = \mathbf{w}_n + \mu E\{e(n)\mathbf{x}^*(n)\}} \tag{9.13}$$

To see how this steepest descent update equation for $\mathbf{w}_n$ performs, let us consider what happens in the case of stationary processes. If $x(n)$ and $d(n)$ are jointly WSS then

$$\begin{aligned} E\{e(n)\mathbf{x}^*(n)\} &= E\{d(n)\mathbf{x}^*(n)\} - E\{\mathbf{w}_n^T\mathbf{x}(n)\mathbf{x}^*(n)\} \\ &= \mathbf{r}_{dx} - \mathbf{R}_x\mathbf{w}_n \end{aligned}$$

and the steepest descent algorithm becomes

$$\boxed{\mathbf{w}_{n+1} = \mathbf{w}_n + \mu\big(\mathbf{r}_{dx} - \mathbf{R}_x\mathbf{w}_n\big)} \tag{9.14}$$

Note that if $\mathbf{w}_n$ is the solution to the Wiener-Hopf equations, $\mathbf{w}_n = \mathbf{R}_x^{-1}\mathbf{r}_{dx}$, then the correction term is zero and $\mathbf{w}_{n+1} = \mathbf{w}_n$ for all n. Of greater interest, however, is how the weights evolve in time, beginning with an arbitrary initial weight vector $\mathbf{w}_0$. The following property defines what is required for $\mathbf{w}_n$ to converge to $\mathbf{w}$.

Property 1. For jointly wide-sense stationary processes, $d(n)$ and $x(n)$, the steepest descent adaptive filter converges to the solution to the Wiener-Hopf equations

$$\lim_{n\to\infty} \mathbf{w}_n = \mathbf{R}_x^{-1}\mathbf{r}_{dx}$$

if the step size satisfies the condition

$$0 < \mu < \frac{2}{\lambda_{\max}} \tag{9.15}$$

where $\lambda_{\max}$ is the maximum eigenvalue of the autocorrelation matrix $\mathbf{R}_x$.

To establish this property, we begin by rewriting Eq. (9.14) as follows:

$$\mathbf{w}_{n+1} = \big(\mathbf{I} - \mu\mathbf{R}_x\big)\mathbf{w}_n + \mu\mathbf{r}_{dx} \tag{9.16}$$

Subtracting $\mathbf{w}$ from both sides of this equation and using the fact that $\mathbf{r}_{dx} = \mathbf{R}_x\mathbf{w}$ we have

$$\mathbf{w}_{n+1} - \mathbf{w} = \big(\mathbf{I} - \mu\mathbf{R}_x\big)\mathbf{w}_n + \mu\mathbf{R}_x\mathbf{w} - \mathbf{w} = \big[\mathbf{I} - \mu\mathbf{R}_x\big]\big(\mathbf{w}_n - \mathbf{w}\big) \tag{9.17}$$

If we let $\mathbf{c}_n$ be the *weight error vector*,

$$\boxed{\mathbf{c}_n = \mathbf{w}_n - \mathbf{w}} \tag{9.18}$$

then Eq. (9.17) becomes

$$\mathbf{c}_{n+1} = \big(\mathbf{I} - \mu\mathbf{R}_x\big)\mathbf{c}_n \tag{9.19}$$

Note that, unless $\mathbf{R}_x$ is a diagonal matrix, there will be cross-coupling between the coefficients of the weight error vector. However, we may decouple these coefficients by diagonalizing the autocorrelation matrix as follows. Using the spectral theorem (p. 44) the autocorrelation matrix may be factored as

$$\mathbf{R}_x = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^H$$

where $\mathbf{\Lambda}$ is a diagonal matrix containing the eigenvalues of $\mathbf{R}_x$, and $\mathbf{V}$ is a matrix whose columns are the eigenvectors of $\mathbf{R}_x$. Since $\mathbf{R}_x$ is Hermitian and nonnegative definite, the eigenvalues are real and non-negative, $\lambda_k \geq 0$, and the eigenvectors may be chosen to be

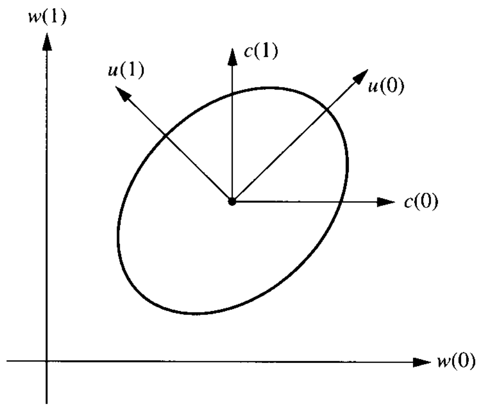


Figure 9.5 *Illustration of the relationships between the vectors* **w**, **c**, *and* **u**.

orthonormal, $\mathbf{V}\mathbf{V}^H = \mathbf{I}$, i.e., $\mathbf{V}$ is *unitary*. Incorporating this factorization into Eq. (9.19) leads to

$$\mathbf{c}_{n+1} = \big(\mathbf{I} - \mu\mathbf{V}\mathbf{\Lambda}\mathbf{V}^H\big)\mathbf{c}_n$$

Using the unitary property of $\mathbf{V}$ we have

$$\mathbf{c}_{n+1} = \big(\mathbf{V}\mathbf{V}^H - \mu\mathbf{V}\mathbf{\Lambda}\mathbf{V}^H\big)\mathbf{c}_n = \mathbf{V}\big(\mathbf{I} - \mu\mathbf{\Lambda}\big)\mathbf{V}^H\mathbf{c}_n$$

Multiplying both sides of the equation by $\mathbf{V}^H$ gives

$$\mathbf{V}^H\mathbf{c}_{n+1} = \big(\mathbf{I} - \mu\mathbf{\Lambda}\big)\mathbf{V}^H\mathbf{c}_n \tag{9.20}$$

If we define

$$\mathbf{u}_n = \mathbf{V}^H\mathbf{c}_n \tag{9.21}$$

then Eq. (9.20) becomes

$$\mathbf{u}_{n+1} = \big(\mathbf{I} - \mu\mathbf{\Lambda}\big)\mathbf{u}_n$$

As illustrated in Fig. 9.5, Eq. (9.21) represents a rotation of the coordinate system for the weight error vector $\mathbf{c}_n$ with the new axes aligned with the eigenvectors $\mathbf{v}_k$ of the autocorrelation matrix. With an initial weight vector $\mathbf{u}_0$, it follows that

$$\mathbf{u}_n = \big(\mathbf{I} - \mu\mathbf{\Lambda}\big)^n\mathbf{u}_0 \tag{9.22}$$

Since $(\mathbf{I} - \mu\mathbf{\Lambda})$ is a diagonal matrix then the kth component of $\mathbf{u}_n$ may be expressed as

$$u_n(k) = (1 - \mu\lambda_k)^n u_0(k) \tag{9.23}$$

In order for $\mathbf{w}_n$ to converge to $\mathbf{w}$ it is necessary that the weight error vector $\mathbf{c}_n$ converge to zero and, therefore, that $\mathbf{u}_n = \mathbf{V}^H\mathbf{c}_n$ converge to zero. This will occur for any $\mathbf{u}_0$ if and only if

$$|1 - \mu\lambda_k| < 1 \quad ; \quad k = 0, 1, \ldots, p$$

which places the following restriction on the step size μ

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

as was to be shown. ■

Having found an expression for the evolution of the vector $\mathbf{u}_n$, we may derive an expression for the evolution of the weight vector $\mathbf{w}_n$. With

$$\mathbf{w}_n = \mathbf{w} + \mathbf{c}_n = \mathbf{w} + \mathbf{V}\mathbf{u}_n = \mathbf{w} + \left[\mathbf{v}_0,\ \mathbf{v}_1,\ \ldots,\mathbf{v}_p\right]\begin{bmatrix} u_n(0) \\ u_n(1) \\ \vdots \\ u_n(p) \end{bmatrix}$$

using Eq. (9.23) we have

$$\mathbf{w}_n = \mathbf{w} + \sum_{k=0}^{p} u_n(k)\mathbf{v}_k = \mathbf{w} + \sum_{k=0}^{p}\left(1 - \mu\lambda_k\right)^n u_0(k)\mathbf{v}_k$$

Since $\mathbf{w}_n$ is a linear combination of the eigenvectors $\mathbf{v}_n$, referred to as the *modes* of the filter, then $\mathbf{w}_n$ will converge no faster than the slowest decaying mode. With each mode decaying as $(1 - \mu\lambda_k)^n$, we may define the time constant τ_k to be the time required for the kth mode to reach $1/e$ of its initial value:

$$(1 - \mu\lambda_k)^{\tau_k} = 1/e$$

Taking logarithms we have

$$\boxed{\tau_k = -\frac{1}{\ln(1 - \mu\lambda_k)}} \qquad (9.24)$$

If μ is small enough so that $\mu\lambda_k \ll 1$, then the time constant may be approximated by

$$\tau_k \approx \frac{1}{\mu\lambda_k}$$

Defining the overall time constant to be the time that it takes for the slowest decaying mode to converge to $1/e$ of its initial value we have

$$\boxed{\tau = \max\left\{\tau_k\right\} \approx \frac{1}{\mu\lambda_{\min}}} \qquad (9.25)$$

Since Property 1 places an upper bound of $2/\lambda_{\max}$ on the step size if $\mathbf{w}_n$ is to converge to $\mathbf{w}$, let us write μ as follows

$$\mu = \alpha\frac{2}{\lambda_{\max}}$$

where α is a normalized step size with $0 < \alpha < 1$. In terms of α, the time constant becomes

$$\tau \approx \frac{1}{2\alpha}\frac{\lambda_{\max}}{\lambda_{\min}} = \frac{1}{2\alpha}\chi$$

where $\chi = \lambda_{\max}/\lambda_{\min}$ is the condition number of the autocorrelation matrix. Thus, the rate of convergence is determined by the eigenvalue spread. The reason for this dependence may be explained geometrically as illustrated in Fig. 9.6. In Fig. 9.6*a* are the error contours for a two-dimensional adaptive filter with $\lambda_1 = \lambda_2 = 1$, i.e., $\chi = 1$, along with the trajectory of $\mathbf{w}_n$. Note that the contours are circles and that, at any point, the direction of steepest descent points toward the minimum of the quadratic function. Fig. 9.6*b*, on the other hand, shows the error contours when $\lambda_1 = 3$ and $\lambda_2 = 1$, i.e., $\chi = 3$, along with the trajectory of $\mathbf{w}_n$.

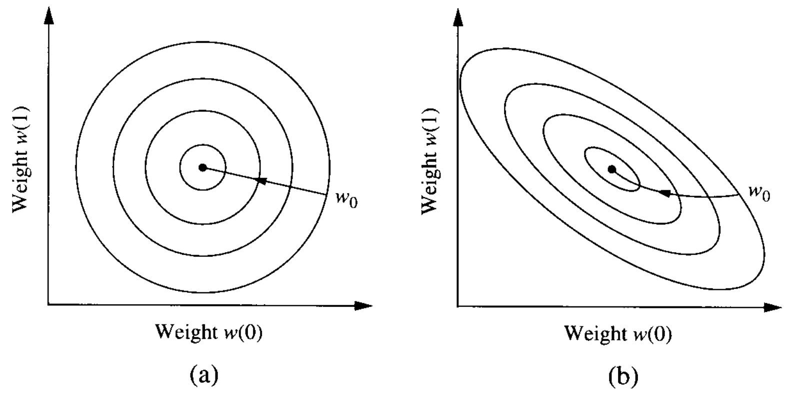


Figure 9.6 *The effect of the condition number on the convergence rate of the steepest descent algorithm. (a) With $\chi = 1$ the direction of steepest descent points toward the minimum of the error surface. (b) With $\chi = 3$ the direction of steepest descent does not, in general, point toward the minimum.*

Note that, unlike the case when $\chi = 1$, the direction of steepest descent is typically not pointing toward the minimum of the quadratic function.

Another important and useful measure of performance is the behavior of the mean-square error as a function of n. From Eq. (7.12) we see that for jointly wide-sense stationary processes the minimum mean-square error is

$$\xi_{\min} = r_d(0) - \mathbf{r}_{dx}^H \mathbf{w} \tag{9.26}$$

For an arbitrary weight vector, $\mathbf{w}_n$, the mean-square error is

$$\begin{aligned}\xi(n) &= E\left\{|e(n)|^2\right\} = E\left\{|d(n) - \mathbf{w}_n^T \mathbf{x}(n)|^2\right\} \\ &= r_d(0) - \mathbf{r}_{dx}^H \mathbf{w}_n - \mathbf{w}_n^H \mathbf{r}_{dx} + \mathbf{w}_n^H \mathbf{R}_x \mathbf{w}_n \end{aligned}\tag{9.27}$$

Expressing $\mathbf{w}_n$ in terms of the weight error vector $\mathbf{c}_n$ this becomes

$$\xi(n) = r_d(0) - \mathbf{r}_{dx}^H(\mathbf{w} + \mathbf{c}_n) - (\mathbf{w} + \mathbf{c}_n)^H \mathbf{r}_{dx} + (\mathbf{w} + \mathbf{c}_n)^H \mathbf{R}_x (\mathbf{w} + \mathbf{c}_n)$$

Expanding the products and using the fact that $\mathbf{R}_x \mathbf{w} = \mathbf{r}_{dx}$ we have

$$\xi(n) = r_d(0) - \mathbf{r}_{dx}^H \mathbf{w} + \mathbf{c}_n^H \mathbf{R}_x \mathbf{c}_n \tag{9.28}$$

Note that the first two terms are equal to the minimum error, $\xi_{\min}$. Therefore, the error at time n is

$$\xi(n) = \xi_{\min} + \mathbf{c}_n^H \mathbf{R}_x \mathbf{c}_n$$

With the decomposition $\mathbf{R}_x = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^H$ and the definition for $\mathbf{u}_n$ given in Eq. (9.21), we have

$$\boxed{\xi(n) = \xi_{\min} + \mathbf{u}_n^H \mathbf{\Lambda}_x \mathbf{u}_n} \tag{9.29}$$

Expanding the quadratic form and using the expression for $u_n(k)$ given in Eq. (9.23) yields

$$\xi(n) = \xi_{\min} + \sum_{k=0}^{p} \lambda_k |u_n(k)|^2 = \xi_{\min} + \sum_{k=0}^{p} \lambda_k (1 - \mu\lambda_k)^{2n} |u_0(k)|^2 \tag{9.30}$$

Thus, if the step size μ satisfies the condition for convergence given in Eq. (9.15), then $\xi(n)$ decays exponentially to $\xi_{\min}$. A plot of $\xi(n)$ versus n is referred to as the *learning curve* and indicates how rapidly the adaptive filter *learns* the solution to the Wiener-Hopf equations.

Although for stationary processes the steepest descent adaptive filter converges to the solution to the Wiener-Hopf equations when $\mu < 2/\lambda_{\max}$, this algorithm is primarily of theoretical interest and finds little use in adaptive filtering applications. The reason for this is that, in order to compute the gradient vector, it is necessary that $E\{e(n)\mathbf{x}^*(n)\}$ be known. For stationary processes this requires that the autocorrelation matrix of $x(n)$, and the cross-correlation between $d(n)$ and $\mathbf{x}(n)$ be known. In most applications, these ensemble averages are unknown and must be estimated from the data. In the next section, we look at the LMS algorithm, which incorporates an *estimate* of the expectation $E\{e(n)\mathbf{x}^*(n)\}$ into the adaptive algorithm.

9.2.2 The LMS Algorithm

In the previous section, we developed the steepest descent adaptive filter, which has a weight-vector update equation given by

$$\mathbf{w}_{n+1} = \mathbf{w}_n + \mu E\{e(n)\mathbf{x}^*(n)\} \tag{9.31}$$

A practical limitation with this algorithm is that the expectation $E\{e(n)\mathbf{x}^*(n)\}$ is generally unknown. Therefore, it must be replaced with an estimate such as the sample mean

$$\hat{E}\{e(n)\mathbf{x}^*(n)\} = \frac{1}{L}\sum_{l=0}^{L-1} e(n-l)\mathbf{x}^*(n-l) \tag{9.32}$$

Incorporating this estimate into the steepest descent algorithm, the update for $\mathbf{w}_n$ becomes

$$\mathbf{w}_{n+1} = \mathbf{w}_n + \frac{\mu}{L}\sum_{l=0}^{L-1} e(n-l)\mathbf{x}^*(n-l) \tag{9.33}$$

A special case of Eq. (9.33) occurs if we use a one-point sample mean ($L = 1$),

$$\hat{E}\{e(n)\mathbf{x}^*(n)\} = e(n)\mathbf{x}^*(n) \tag{9.34}$$

In this case, the weight vector update equation assumes a particularly simple form

$$\boxed{\mathbf{w}_{n+1} = \mathbf{w}_n + \mu e(n)\mathbf{x}^*(n)} \tag{9.35}$$

and is known as the *LMS algorithm* [36]. The simplicity of the algorithm comes from the fact that the update for the kth coefficient,

$$w_{n+1}(k) = w_n(k) + \mu e(n)x^*(n-k)$$

requires only one multiplication and one addition (the value for $\mu e(n)$ need only be computed once and may be used for all of the coefficients). Therefore, an LMS adaptive filter having $p+1$ coefficients requires $p+1$ multiplications and $(p+1)$ additions to update the filter coefficients. In addition, one addition is necessary to compute the error $e(n) = d(n) - y(n)$ and one multiplication is needed to form the product $\mu e(n)$. Finally, $p+1$ multiplications and p additions are necessary to calculate the output, $y(n)$, of the adaptive filter. Thus, a total of $2p+3$ multiplications and $2p+2$ additions per output point are required. The complete LMS algorithm is summarized in Table 9.1 and a MATLAB program is given in

Table 9.1 ***The LMS Algorithm for a pth-Order FIR Adaptive Filter***

Parameters:	p = Filter order
	μ = Step size
Initialization:	$\mathbf{w}_0 = \mathbf{0}$
Computation:	For $n = 0, 1, 2, \ldots$
	(a) $y(n) = \mathbf{w}_n^T \mathbf{x}(n)$
	(b) $e(n) = d(n) - y(n)$
	(c) $\mathbf{w}_{n+1} = \mathbf{w}_n + \mu e(n)\mathbf{x}^*(n)$

The LMS Algorithm

```
function [A,E] = lms(x,d,mu,nord,a0)
%
X=convm(x,nord);
[M,N] = size(X);
if nargin < 5,   a0 = zeros(1,N);   end
a0 = a0(:).';
E(1) = d(1) - a0*X(1,:).';
A(1,:) = a0 + mu*E(1)*conj(X(1,:));
if M>1
for k=2:M-nord+1;
    E(k) = d(k) - A(k-1,:)*X(k,:).';
    A(k,:) = A(k-1,:) + mu*E(k)*conj(X(k,:));
    end;
end;
```

Figure 9.7 *A* MATLAB *program to estimate a process $d(n)$ from a related process $x(n)$ using the LMS algorithm.*

Fig. 9.7. Although based on a very crude estimate of $E\{e(n)\mathbf{x}^*(n)\}$, we will see that the LMS adaptive filter often performs well enough to be used successfully in a number of applications. In the following section, we consider the convergence of the LMS adaptive filter.

9.2.3 Convergence of the LMS Algorithm

In estimating the ensemble average $E\{e(n)\mathbf{x}^*(n)\}$ with a one-point sample average $e(n)\mathbf{x}^*(n)$, the LMS algorithm replaces the gradient in the steepest descent algorithm,

$$\nabla\xi(n) = -E\{e(n)\mathbf{x}^*(n)\}$$

with an estimated gradient

$$\hat{\nabla}\xi(n) = -e(n)\mathbf{x}^*(n) \tag{9.36}$$

When this is done, the correction that is applied to $\mathbf{w}_n$ is generally not aligned with the

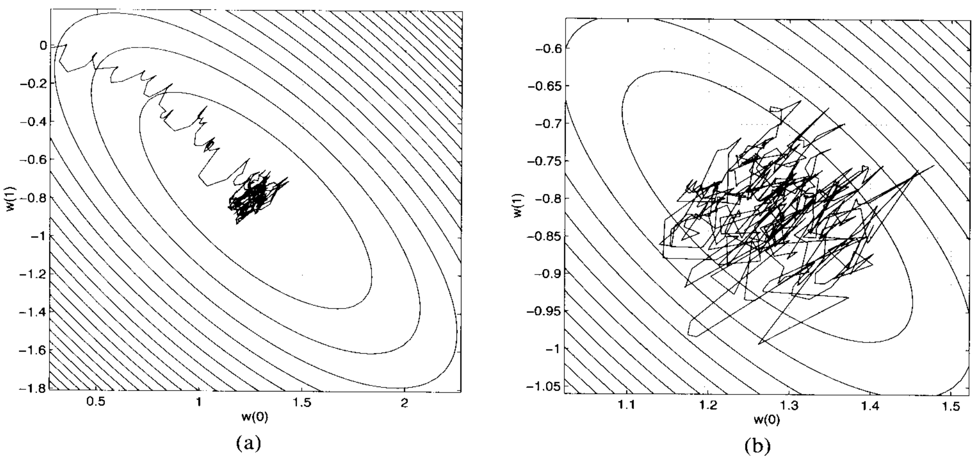


Figure 9.8 *Weight trajectories for a two-coefficient LMS adaptive filter. (a) The trajectory of* $\mathbf{w}_n$ *for 500 iterations with* $\mathbf{w}_0 = \mathbf{0}$. *The solution to the Wiener-Hopf equations is at the center of the ellipses. (b) The trajectory with* $\mathbf{w}_0 = \mathbf{R}_x^{-1}\mathbf{r}_{dx}$.

direction of steepest descent. However, since the gradient estimate is unbiased,

$$E\left\{\hat{\nabla}\xi(n)\right\} = -E\left\{e(n)\mathbf{x}^*(n)\right\} = \nabla\xi(n)$$

then the correction that is applied is, on the average, in the direction of steepest descent. An illustration of the behavior that is typically observed with the LMS algorithm for stationary processes is shown in Fig. 9.8. In Fig. 9.8*a*, with the weight vector initialized to $\mathbf{w}_0 = \mathbf{0}$, we see that the sequence of weights generally moves toward the solution to the Wiener-Hopf equations, $\mathbf{w} = \mathbf{R}_x^{-1}\mathbf{r}_{dx}$. However, as illustrated in Fig. 9.8*b*, if the weight vector is initialized with the solution to the Wiener-Hopf equations then, although the gradient at this point is zero, since a gradient estimate is used, $\mathbf{w}_n$ moves randomly within a neighborhood of this solution. In order to explain these observations, we will now consider the convergence of the LMS adaptive filter.

Since $\mathbf{w}_n$ is a vector of random variables, the convergence of the LMS algorithm must be considered within a statistical framework. Therefore, we will begin by assuming that $x(n)$ and $d(n)$ are jointly wide-sense stationary processes, and will determine when the coefficients $\mathbf{w}_n$ converge in the mean to $\mathbf{w} = \mathbf{R}_x^{-1}\mathbf{r}_{dx}$, i.e.,

$$\lim_{n\to\infty} E\left\{\mathbf{w}_n\right\} = \mathbf{w} = \mathbf{R}_x^{-1}\mathbf{r}_{dx}$$

We begin by substituting Eq. (9.7) into the LMS coefficient update equation as follows:

$$\mathbf{w}_{n+1} = \mathbf{w}_n + \mu\left[d(n) - \mathbf{w}_n^T\mathbf{x}(n)\right]\mathbf{x}^*(n)$$

Taking the expected value, we have

$$E\left\{\mathbf{w}_{n+1}\right\} = E\left\{\mathbf{w}_n\right\} + \mu E\left\{d(n)\mathbf{x}^*(n)\right\} - \mu E\left\{\mathbf{x}^*(n)\mathbf{x}^T(n)\mathbf{w}_n\right\} \tag{9.37}$$

Although the last term in Eq. (9.37) is not easy to evaluate, it may be simplified considerably if we make the following *independence assumption* [20]:

Independence Assumption. The data $\mathbf{x}(n)$ and the LMS weight vector $\mathbf{w}_n$ are statistically independent.[2]

With this assumption, Eq. (9.37) becomes

$$\begin{aligned} E\{\mathbf{w}_{n+1}\} &= E\{\mathbf{w}_n\} + \mu E\{d(n)\mathbf{x}^*(n)\} - \mu E\{\mathbf{x}^*(n)\mathbf{x}^T(n)\}E\{\mathbf{w}_n\} \\ &= (\mathbf{I} - \mu\mathbf{R}_x)E\{\mathbf{w}_n\} + \mu\mathbf{r}_{dx} \end{aligned} \tag{9.38}$$

which is the same as Eq. (9.16) for the weight vector in the steepest descent algorithm. Therefore, the analysis for the steepest descent algorithm is applicable to $E\{\mathbf{w}_{n+1}\}$. In particular, it follows from Eq. (9.22) that

$$E\{\mathbf{u}_n\} = (\mathbf{I} - \mu\mathbf{\Lambda})^n\mathbf{u}_0 \tag{9.39}$$

where

$$\mathbf{u}_n = \mathbf{V}^H[\mathbf{w}_n - \mathbf{w}]$$

Since $\mathbf{w}_n$ will converge in the mean to $\mathbf{w}$ if $E\{\mathbf{u}_n\}$ converges to zero, then we have the following property:

Property 2. For jointly wide-sense stationary processes, the LMS algorithm *converges in the mean* if

$$0 < \mu < \frac{2}{\lambda_{\max}} \tag{9.40}$$

and the independence assumption is satisfied.

Although Eq. (9.40) places a bound on the step size for convergence in the mean, this bound is of limited use for two reasons. First, it is generally acknowledged that the upper bound is too large to ensure stability of the LMS algorithm since it is not sufficient to guarantee that the coefficient vector will remain bounded for all n. For example, although this bound ensures that $E\{\mathbf{w}_n\}$ converges, it places no constraints on how large the variance of $\mathbf{w}_n$ may become. Second, since the upper bound is expressed in terms of the largest eigenvalue of $\mathbf{R}_x$, using this bound requires that $\mathbf{R}_x$ be known. If this matrix is unknown, then it becomes necessary to estimate $\lambda_{\max}$. One way around this difficulty is to use the fact that $\lambda_{\max}$ may be upper bounded by the trace of $\mathbf{R}_x$,

$$\lambda_{\max} \le \sum_{k=0}^{p} \lambda_k = \mathrm{tr}(\mathbf{R}_x)$$

Therefore, if $x(n)$ is wide-sense stationary, then $\mathbf{R}_x$ is Toeplitz and the trace becomes

$$\mathrm{tr}(\mathbf{R}_x) = (p+1)r_x(0) = (p+1)E\{|x(n)|^2\}$$

[2]Since $\mathbf{w}_n$ depends on the previous input vectors $\mathbf{x}(n-1), \mathbf{x}(n-2), \ldots$, this assumption can only be approximately true. However, experience with the LMS algorithm shows that this assumption leads to convergence properties that are generally in close agreement with experiments and computer simulations.

As a result, Eq. (9.40) may be replaced with the more conservative bound

$$\boxed{0 < \mu < \frac{2}{(p+1)E\{|x(n)|^2\}}} \tag{9.41}$$

Although we have simply replaced one unknown with another, $E\{|x(n)|^2\}$ is more easily estimated since it represents the power in $x(n)$. For example, $E\{|x(n)|^2\}$ could be estimated using an average such as

$$\hat{E}\{|x(n)|^2\} = \frac{1}{N}\sum_{k=0}^{N-1}|x(n-k)|^2$$

In the following example we consider the use of an LMS adaptive filter for linear prediction.

Example 9.2.1 *Adaptive Linear Prediction Using the LMS Algorithm*

Let $x(n)$ be a second-order autoregressive process that is generated according to the difference equation

$$x(n) = 1.2728x(n-1) - 0.81x(n-2) + v(n) \tag{9.42}$$

where $v(n)$ is unit variance white noise. As we saw in Section 7.3.4, the optimum causal linear predictor for $x(n)$ is

$$\hat{x}(n) = 1.2728x(n-1) - 0.81x(n-2)$$

However, in order to design this predictor (i.e., to know that the optimum predictor coefficients are 1.2728 and -0.81) it is necessary to know the autocorrelation sequence of $x(n)$. Therefore, suppose we consider an adaptive linear predictor of the form:

$$\hat{x}(n) = w_n(1)x(n-1) + w_n(2)x(n-2)$$

as shown in Fig. 9.9. With the LMS algorithm, the predictor coefficients $w_n(k)$ are updated as follows:

$$w_{n+1}(k) = w_n(k) + \mu e(n)x^*(n-k)$$

If the step size μ is sufficiently small, then the coefficients $w_n(1)$ and $w_n(2)$ will converge in the mean to their optimum values, $w(1) = 1.2728$ and $w(2) = -0.81$, respectively. Note that the prediction error is

$$e(n) = x(n) - \hat{x}(n) = \big[1.2728 - w_n(1)\big]x(n-1) + \big[-0.81 - w_n(1)\big]x(n-2) + v(n)$$

Therefore, when $w_n(1) = 1.2728$ and $w_n(2) = -0.81$, the error becomes $e(n) = v(n)$, and the minimum mean-square error is[3]

$$\xi_{\min} = \sigma_v^2 = 1$$

Although we might expect the mean-square error $E\{|e(n)|^2\}$ to converge to $\xi_{\min}$ as $\mathbf{w}_n$ converges to $\mathbf{w}$, as we will soon discover, this is not the case.

To see how this adaptive linear predictor behaves in practice, suppose that the weight vector is initialized to zero, $\mathbf{w}_0 = \mathbf{0}$, and that the step size is $\mu = 0.02$. Shown in Fig. 9.10*a*

[3]This may also be shown by evaluating the expression for the minimum mean-square error given in Eq. (9.26).

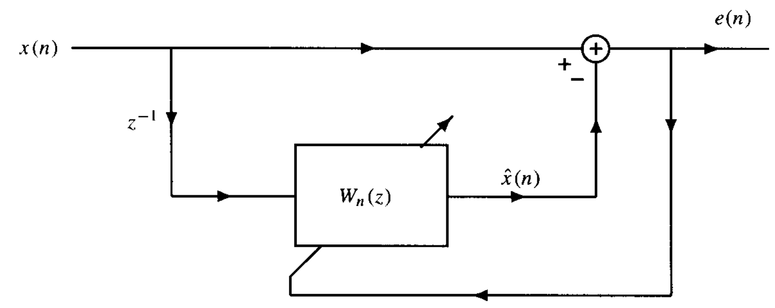


Figure 9.9 *An adaptive filter for linear prediction.*

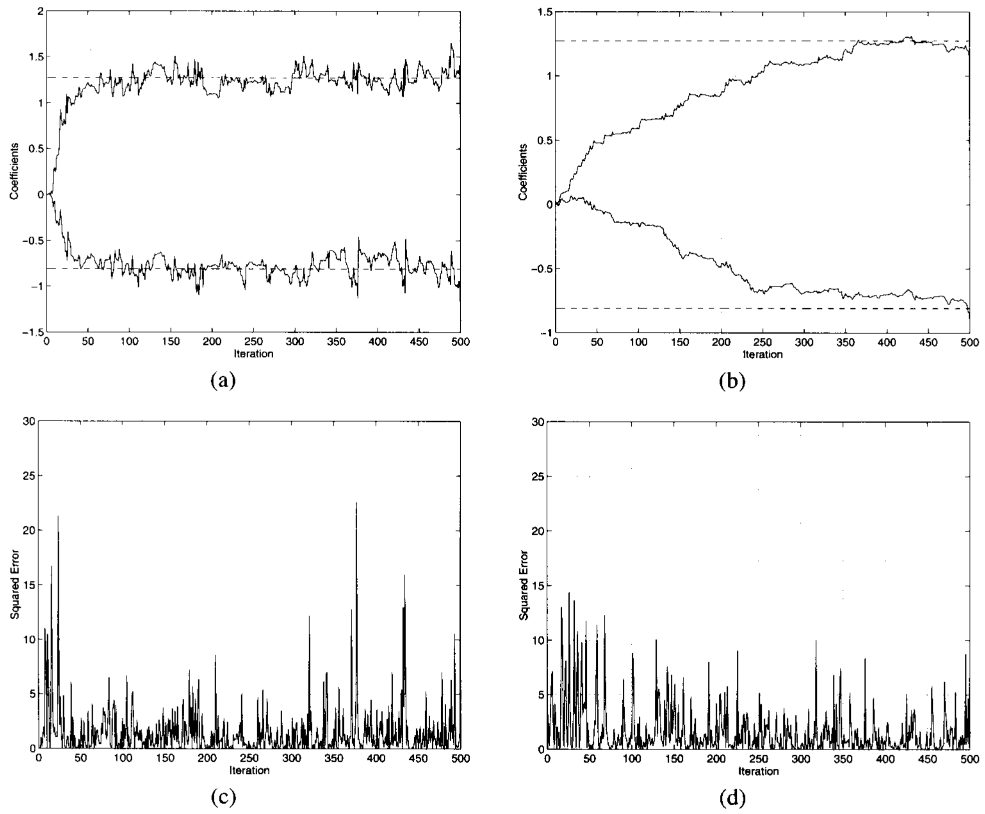


Figure 9.10 *Performance of a two-coefficient LMS adaptive linear predictor. The trajectories of the predictor coefficients are shown for step sizes of (a) $\mu = 0.02$ and (b) $\mu = 0.004$ with the correct values indicated by the dashed lines. A plot of the squared error, $e^2(n)$, is shown in (c) and (d) for step sizes of $\mu = 0.02$ and $\mu = 0.004$, respectively.*

are the trajectories of $w_n(1)$ and $w_n(2)$ versus n. As we see, there is a great deal of fluctuation in the weights, even after they have converged to within a neighborhood of their steady-state values. By contrast, shown in Fig. 9.10*b* are the trajectories of the predictor coefficients for a step size $\mu = 0.004$. Compared to a step size of $\mu = 0.02$, we see that, although the weights take longer to converge, the trajectories are much smoother, illustrating the basic trade-off between rate of convergence and stability (variance) of the final solution. Shown in parts c and d are plots of the squared error $e^2(n)$ for $\mu = 0.02$ and $\mu = 0.004$, respectively. What is so striking in these plots is the large amount of variation in $e^2(n)$ versus n compared to the learning curves for the steepest descent algorithm.

Before leaving this example, let us compute the maximum step size that is allowed for the adaptive linear predictor to converge in the mean. Given the coefficients of the filter that are used to generate $x(n)$, we may use the inverse Levinson-Durbin recursion to find the autocorrelation sequence $r_x(k)$. Using the m-file `ator.m` we find

$$r_x(0) = 5.7523 \quad ; \quad r_x(1) = 4.0450$$

Therefore, $\lambda_{\max} = r_x(0) + r_x(1) = 9.7973$ and the bound on the step size is

$$0 < \mu < 0.2041$$

Note that the step sizes used in Fig. 9.10 are at least an order of magnitude smaller than the largest value allowed for convergence in the mean. This is typically the case and, in fact, with a step size much larger than $\mu = 0.02$, the coefficients $w_n(k)$ begin to fluctuate wildly and eventually become unstable.

As we observed in the previous example, as the weight vector begins to converge in the mean, the coefficients begin to fluctuate about their optimum values. These fluctuations are due to the noisy gradient vectors that are used to form the corrections to $\mathbf{w}_n$. As a result, the variance of the weight error vector does not go to zero and the mean-square error is larger than the minimum mean-square error by an amount referred to as the *excess mean-square error*. This behavior is illustrated in Fig. 9.8*b* which shows that, as $\mathbf{w}_n$ oscillates about $\mathbf{w} = \mathbf{R}_x^{-1}\mathbf{r}_{dx}$, the corresponding mean-square error $\xi(n)$ has a value that, on the average, exceeds the minimum mean-square error. In order to quantify this excess mean-square error, we will write the error at time n as follows:

$$e(n) = d(n) - \mathbf{w}_n^T\mathbf{x}(n) = d(n) - \big(\mathbf{w} + \mathbf{c}_n\big)^T\mathbf{x}(n) = e_{\min}(n) - \mathbf{c}_n^T\mathbf{x}(n)$$

where $e_{\min}(n)$ is the error that would occur if the optimum filter coefficients were used, i.e.,

$$e_{\min}(n) = d(n) - \mathbf{w}^T\mathbf{x}(n)$$

Assuming that the filter is in the steady-state with $E\{\mathbf{c}_n\} = 0$, the mean-square error may be expressed as

$$\xi(n) = E\big\{|e(n)|^2\big\} = \xi_{\min} + \xi_{\text{ex}}(n)$$

where

$$\xi_{\min} = E\big\{|e_{\min}(n)|^2\big\}$$

is the minimum mean-square error and $\xi_{ex}(n)$ is the *excess mean-square error*, which depends on the statistics of $\mathbf{x}(n)$, $\mathbf{c}_n$, and $d(n)$. Although $\xi_{ex}(n)$ is not easy to evaluate, by invoking the independence assumption, the following property may be established with a fair amount of effort [20].

Property 3. The mean-square error $\xi(n)$ converges to a steady-state value of

$$\xi(\infty) = \xi_{\min} + \xi_{\mathrm{ex}}(\infty) = \xi_{\min} \frac{1}{1 - \mu \sum_{k=0}^{p} \frac{\lambda_k}{2 - \mu\lambda_k}} \tag{9.43}$$

and the LMS algorithm is said to *converge in the mean-square* if and only if the step-size μ satisfies the following two conditions:

$$0 < \mu < \frac{2}{\lambda_{\max}} \tag{9.44}$$

$$\mu \sum_{k=0}^{p} \frac{\lambda_k}{2 - \mu\lambda_k} < 1 \tag{9.45}$$

Note that Eq. (9.44) is the condition that is required for the LMS algorithm to converge in the mean, and Eq. (9.45) guarantees that $\xi(\infty)$ is positive. Solving Eq. (9.43) for $\xi_{ex}(\infty)$ we find

$$\xi_{\mathrm{ex}}(\infty) = \mu\, \xi_{\min} \frac{\sum_{k=0}^{p} \frac{\lambda_k}{2 - \mu\lambda_k}}{1 - \mu \sum_{k=0}^{p} \frac{\lambda_k}{2 - \mu\lambda_k}} \tag{9.46}$$

If $\mu \ll 2/\lambda_{\max}$, as is typically the case, then $\mu\lambda_k \ll 2$ and Eq. (9.45) may be simplified to

$$\frac{1}{2}\mu \sum_{k=0}^{p} \lambda_k < 1$$

or,

$$\mu < \frac{2}{\mathrm{tr}(\mathbf{R}_x)}$$

When $\mu \ll 2/\lambda_{\max}$ it also follows that

$$\xi(\infty) \approx \xi_{\min} \frac{1}{1 - \frac{1}{2}\mu\, \mathrm{tr}(\mathbf{R}_x)}$$

and the excess mean-square error given in Eq. (9.46) is approximately

$$\xi_{\mathrm{ex}}(\infty) \approx \mu\, \xi_{\min} \frac{\frac{1}{2}\mathrm{tr}(\mathbf{R}_x)}{1 - \frac{1}{2}\mu\, \mathrm{tr}(\mathbf{R}_x)} \approx \frac{1}{2}\mu\, \xi_{\min}\, \mathrm{tr}(\mathbf{R}_x) \tag{9.47}$$

Thus, for small μ, the excess mean-square error is proportional to the step size μ. Adaptive filters may be described in terms of their *misadjustment*, which is a normalized mean-square error that is defined as follows.

Definition. The misadjustment $\mathcal{M}$ is the ratio of the steady-state excess mean-square error to the minimum mean-square error,

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

From Eq. (9.47) we see that if the step size is small, $\mu \ll 2/\lambda_{\max}$, then the misadjustment is approximately

$$\mathcal{M} \approx \mu \frac{\frac{1}{2}\text{tr}(\mathbf{R}_x)}{1 - \frac{1}{2}\mu\,\text{tr}(\mathbf{R}_x)} \approx \frac{1}{2}\mu\,\text{tr}(\mathbf{R}_x)$$

Example 9.2.2 *LMS Misadjustment*

In this example, we look at the learning curves for the adaptive linear predictor considered in Example 9.2.1, and evaluate the excess mean-square error and the misadjustment for different step sizes. Since the learning curve is a plot of $\xi(n) = E\{|e(n)|^2\}$ versus n, we may approximate the learning curve by averaging plots of $|e(n)|^2$ that are obtained by repeatedly implementing the adaptive predictor. For example, implementing the adaptive predictor K times, and denoting the squared error at time n on the kth trial by $|e_k(n)|^2$, we have

$$\hat{\xi}(n) = \hat{E}\{|e(n)|^2\} = \frac{1}{K}\sum_{k=1}^{K}|e_k(n)|^2$$

With $K = 200$, an initial weight vector of zero, and step sizes of $\mu = 0.02$ and $\mu = 0.004$, these estimates of the learning curves are shown in Fig. 9.11. One property of the LMS algorithm that we are able to observe from these plots is that, when the step size is decreased, the convergence of the adaptive filter to its steady-state value is slower, but the average steady-state squared error is smaller.

We may estimate the steady-state mean-square error from these plots by averaging $\hat{\xi}(n)$ over n after the LMS algorithm has reached steady-state. For example, with

$$\hat{\xi}(\infty) = \frac{1}{100}\sum_{n=901}^{1000}\hat{E}\{|e(n)|^2\}$$

we find

$$\hat{\xi}(\infty) = \begin{cases} 1.1942 & ; \quad \text{for } \mu = 0.02 \\ 1.0155 & ; \quad \text{for } \mu = 0.004 \end{cases}$$

We may compare these results to the theoretical steady-state mean-square error using Eq. (9.43). With $\xi_{\min} = 1$, and eigenvalues $\lambda_1 = 9.7924$ and $\lambda_2 = 1.7073$ (see Example 9.2.1), it follows that

$$\xi(\infty) = \begin{cases} 1.1441 & ; \quad \text{for } \mu = 0.02 \\ 1.0240 & ; \quad \text{for } \mu = 0.004 \end{cases}$$

which is in fairly close agreement with the estimated values given above.

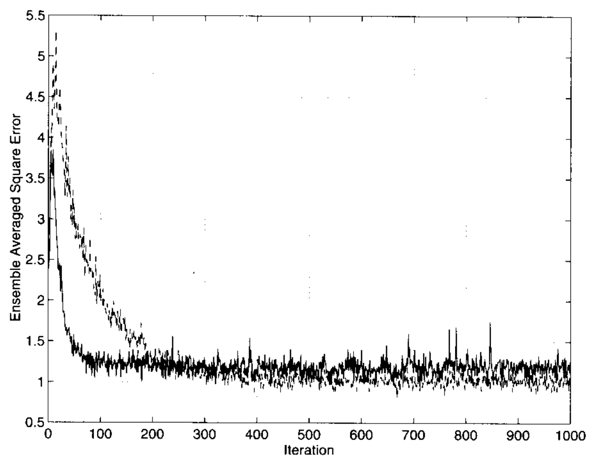


Figure 9.11 *Approximations to the learning curves for a second-order LMS adaptive linear predictor using step sizes of* $\mu = 0.02$ *(solid line) and* $\mu = 0.004$ *(dotted line).*

9.2.4 Normalized LMS

As we have seen, one of the difficulties in the design and implementation of the LMS adaptive filter is the selection of the step size μ. For stationary processes, the LMS algorithm converges in the mean if $0 < \mu < 2/\lambda_{\max}$, and converges in the mean-square if $0 < \mu < 2/\text{tr}(\mathbf{R}_x)$. However, since $\mathbf{R}_x$ is generally unknown, then either $\lambda_{\max}$ or $\mathbf{R}_x$ must be estimated in order to use these bounds. One way around this difficulty is to use the fact that, for stationary processes, $\text{tr}(\mathbf{R}_x) = (p+1)E\{|x(n)|^2\}$. Therefore, the condition for mean-square convergence may be replaced with

$$0 < \mu < \frac{2}{(p+1)E\{|x(n)|^2\}}$$

where $E\{|x(n)|^2\}$ is the power in the process $x(n)$. This power may be estimated using a time average such as[4]

$$\hat{E}\{|x(n)|^2\} = \frac{1}{p+1}\sum_{k=0}^{p} |x(n-k)|^2$$

which leads to the following bound on the step size for mean-square convergence:

$$0 < \mu < \frac{2}{\mathbf{x}^H(n)\mathbf{x}(n)}$$

A convenient way to incorporate this bound into the LMS adaptive filter is to use a (time-varying) step size of the form

$$\mu(n) = \frac{\beta}{\mathbf{x}^H(n)\mathbf{x}(n)} = \frac{\beta}{\|\mathbf{x}(n)\|^2} \tag{9.48}$$

where β is a *normalized step size* with $0 < \beta < 2$. Replacing μ in the LMS weight vector update equation with $\mu(n)$ leads to the Normalized LMS algorithm (NLMS), which is given

[4]Since this estimate uses only those values of $x(n)$ that are within the tapped delay line at time n, no extra memory is required to evaluate this sum.

The Normalized LMS Algorithm

```
function [A,E] = nlms(x,d,beta,nord,a0)
%
X=convm(x,nord);
[M,N] = size(X);
if nargin < 5,    a0 = zeros(1,N);    end
a0 = a0(:).';
E(1) = d(1) - a0*X(1,:).';
DEN=X(1,:)*X(1,:)' + 0.0001;
A(1,:) = a0 + beta/DEN*E(1)*conj(X(1,:));
if M>1
for k=2:M-nord+1;
    E(k) = d(k) - A(k-1,:)*X(k,:).';
    DEN=X(k,:)*X(k,:)' + 0.0001;
    A(k,:) = A(k-1,:) + beta/DEN*E(k)*conj(X(k,:));
    end;
end;
```

Figure 9.12 *A* MATLAB *program for the Normalized LMS algorithm.*

by

$$\boxed{\mathbf{w}_{n+1} = \mathbf{w}_n + \beta \frac{\mathbf{x}^*(n)}{\|\mathbf{x}(n)\|^2} e(n)} \tag{9.49}$$

Note that the effect of the normalization by $\|\mathbf{x}(n)\|^2$ is to alter the magnitude, but not the *direction*, of the estimated gradient vector. Therefore, with the appropriate set of statistical assumptions it may be shown that the normalized LMS algorithm converges in the mean-square if $0 < \beta < 2$ [4,34].

In the LMS algorithm, the correction that is applied to $\mathbf{w}_n$ is proportional to the input vector $\mathbf{x}(n)$. Therefore, when $\mathbf{x}(n)$ is large, the LMS algorithm experiences a problem with *gradient noise amplification*. With the normalization of the LMS step size by $\|\mathbf{x}(n)\|^2$ in the NLMS algorithm, however, this noise amplification problem is diminished. Although the NLMS algorithm bypasses the problem of noise amplification, we are now faced with a similar problem that occurs when $\|\mathbf{x}(n)\|$ becomes too small. An alternative, therefore, is to use the following modification to the NLMS algorithm:

$$\mathbf{w}_{n+1} = \mathbf{w}_n + \beta \frac{\mathbf{x}^*(n)}{\epsilon + \|\mathbf{x}(n)\|^2} e(n) \tag{9.50}$$

where ϵ is some small positive number. A MATLAB program for the normalized LMS algorithm is given in Fig. 9.12.

Compared with the LMS algorithm, the normalized LMS algorithm requires additional computation to evaluate the normalization term $\|\mathbf{x}(n)\|^2$. However, if this term is evaluated recursively as follows

$$\|\mathbf{x}(n+1)\|^2 = \|\mathbf{x}(n)\|^2 + |x(n+1)|^2 - |x(n-p)|^2 \tag{9.51}$$

then the extra computation involves only two squaring operations, one addition, and one substraction.

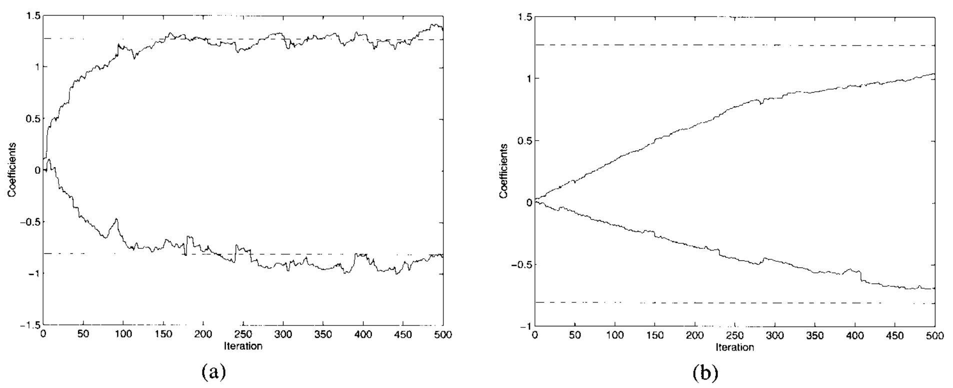


Figure 9.13 *A two-coefficient normalized LMS adaptive linear predictor. The trajectories of the predictor coefficients are shown for step sizes of (a) $\beta = 0.05$ and (b) $\beta = 0.01$ with the correct values for the coefficients indicated by the dashed lines.*

Example 9.2.3 *Adaptive Linear Prediction Using the NLMS Algorithm*

In this example, we consider once again the problem of adaptive linear prediction. This time, however, we will use the normalized LMS algorithm. As in Example 9.2.1, the process that is to be predicted is the AR(2) process that is generated by the difference equation

$$x(n) = 1.2728x(n-1) - 0.81x(n-2) + v(n)$$

where $v(n)$ is unit variance white noise. With a two-coefficient LMS adaptive predictor we have

$$\hat{x}(n) = w_n(1)x(n-1) + w_n(2)x(n-2)$$

where the predictor coefficients are updated according to

$$w_{n+1}(k) = w_n(k) + \beta \frac{x(n-k)}{\epsilon + x^2(n-1) + x^2(n-2)} e(n) \quad ; \quad k = 1, 2$$

Using normalized step sizes of $\beta = 0.05$ and $\beta = 0.01$, with $\epsilon = 0.0001$ and $w_0(1) = w_0(2) = 0$ we obtain the sequence of predictor coefficients shown in Fig. 9.13*a* and *b*, respectively. Comparing these results to those shown in Fig. 9.10 we see that the trajectories of the coefficients are similar. The difference, however, is that for the NLMS algorithm it is not necessary to estimate $\lambda_{\max}$ in order to select a step size.

9.2.5 Application: Noise Cancellation

In Section 7.2.3, we looked at the problem of *noise cancellation* in which a process $d(n)$ is to be estimated from a noise corrupted observation

$$x(n) = d(n) + v_1(n)$$

Clearly, without any information about $d(n)$ or $v_1(n)$ it is not possible to separate the signal from the noise. However, given a *reference signal*, $v_2(n)$, that is correlated with $v_1(n)$, then this reference signal may be used to estimate the noise $v_1(n)$, and this estimate may then be subtracted from $x(n)$ to form an estimate of $d(n)$,

$$\hat{d}(n) = x(n) - \hat{v}_1(n)$$

For example, if $d(n)$, $v_1(n)$, and $v_2(n)$ are jointly wide-sense stationary processes, and if the autocorrelation $r_{v_2}(k)$ and the cross-correlation $r_{v_1v_2}(k)$ are known, then a Wiener filter may be designed to find the minimum mean-square estimate of $v_1(n)$ as illustrated in Fig. 9.14*a*. In practice, however, a stationarity assumption is not generally appropriate and, even if it were, the required statistics of $v_2(n)$ and $v_1(n)$ are generally unknown. Therefore, as an alternative to the Wiener filter, let us consider the adaptive noise canceller shown in Fig. 9.14*b*. If the reference signal $v_2(n)$ is uncorrelated with $d(n)$, then it follows from the

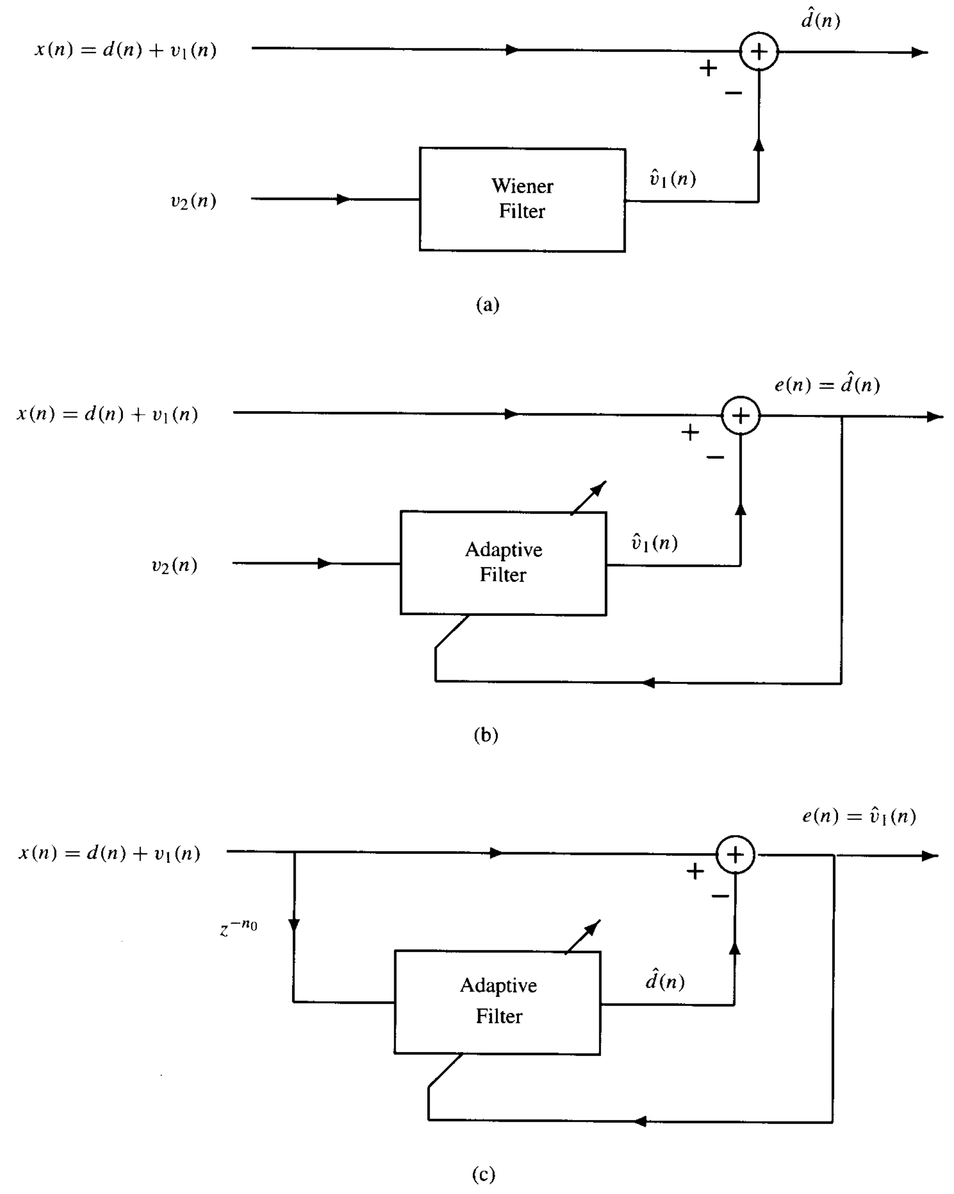


Figure 9.14 *Noise cancellation using (a) a Wiener filter, (b) an adaptive noise canceller with a reference signal $v(n)$, and (c) adaptive noise cancellation without a reference.*

discussion in Section 9.1 that minimizing the mean-square error $E\{|e(n)|^2\}$ is equivalent to minimizing $E\{|v_1(n) - \hat{v}_1(n)|^2\}$. In other words, the output of the adaptive filter is the minimum mean-square estimate of $v_1(n)$. Basically, if there is no information about $d(n)$ in the reference signal $v_2(n)$, then the best that the adaptive filter can do to minimize $E\{|e(n)|^2\}$ is to remove the part of $e(n)$ that may be estimated from $v_2(n)$, which is $v_1(n)$. Since the output of the adaptive filter is the minimum mean-square estimate of $v_1(n)$, then it follows that $e(n)$ is the minimum mean-square estimate of $d(n)$.

As a specific example, let us reconsider the problem presented in Example 7.2.6 in which the signal to be estimated is a sinusoid,

$$d(n) = \sin(n\omega_0 + \phi)$$

with $\omega_0 = 0.05\pi$, and the noise sequences $v_1(n)$ and $v_2(n)$ are generated by the first-order difference equations

$$\begin{aligned} v_1(n) &= 0.8v_1(n-1) + g(n) \\ v_2(n) &= -0.6v_2(n-1) + g(n) \end{aligned} \tag{9.52}$$

where $g(n)$ is a zero-mean, unit variance white noise process that is uncorrelated with $d(n)$. Shown in Fig. 9.15*a* is a plot of 1000 samples of the sinusoid and in Fig. 9.15*b* is the noisy signal $x(n) = d(n) + v_1(n)$. The reference signal $v_2(n)$ that is used to estimate $v_1(n)$ is shown in Fig. 9.15*c*. Using a 12th-order adaptive noise canceller with coefficients that are updated using the normalized LMS algorithm, the estimate of $d(n)$ that is produced with a step size $\beta = 0.25$ is shown in Fig. 9.15*d*. As we see from this figure, after about 100 iterations the adaptive filter is producing a fairly accurate estimate of $d(n)$ and, after about 200 iterations the adaptive filter appears to have settled down into its steady-state behavior. Although not quite as good as the estimate that is produced with a 6th-order Wiener filter as shown in Fig. 7.9*d* on p. 351, the adaptive noise canceller, unlike the Wiener filter, does not require any statistical information.

One of the advantages of this adaptive noise canceller over a Wiener filter is that it may be used when the processes are nonstationary. For example, let $v_1(n)$ and $v_2(n)$ be nonstationary processes that are generated by the first-order difference equations given in Eq. (9.52), where $g(n)$ is nonstationary white noise with a variance that increases linearly from $\sigma_g^2(0) = 0.25$ to $\sigma_g^2(1000) = 6.25$. As before, $d(n)$ is estimated using a 12th-order adaptive noise canceller with coefficients that are updated according to the normalized LMS algorithm. Shown in Fig. 9.16*a* is the desired signal $d(n)$ and in Fig. 9.16*b* is the noisy signal $x(n)$. The increasing variance in the additive noise is clearly evident. The nonstationary reference signal $v_2(n)$ that is used to estimate $v_1(n)$ is shown in Fig. 9.16*c* and in Fig. 9.16*d* is the estimate of $d(n)$ that is produced by the adaptive filter. As we see in this figure, the performance of the adaptive noise canceller is not significantly affected by the nonstationarity of the noise (note that for $n > 250$ the variance of the nonstationary noise is larger than the variance of the noise in Fig. 9.15).

The key to the successful operation of the adaptive noise canceller in Fig. 9.14*b* is the availability of a reference signal, $v_2(n)$, that may be used to estimate the additive noise $v_1(n)$. Unfortunately, in many applications a reference signal is not available and another approach must be considered. In some cases, however, it is possible to derive a reference signal by simply delaying the process $x(n) = d(n) + v_1(n)$. For example, suppose that $d(n)$ is a narrowband process and that $v_1(n)$ is a broadband process with

$$E\{v_1(n)v_1(n-k)\} = 0 \quad ; \quad |k| > k_0$$

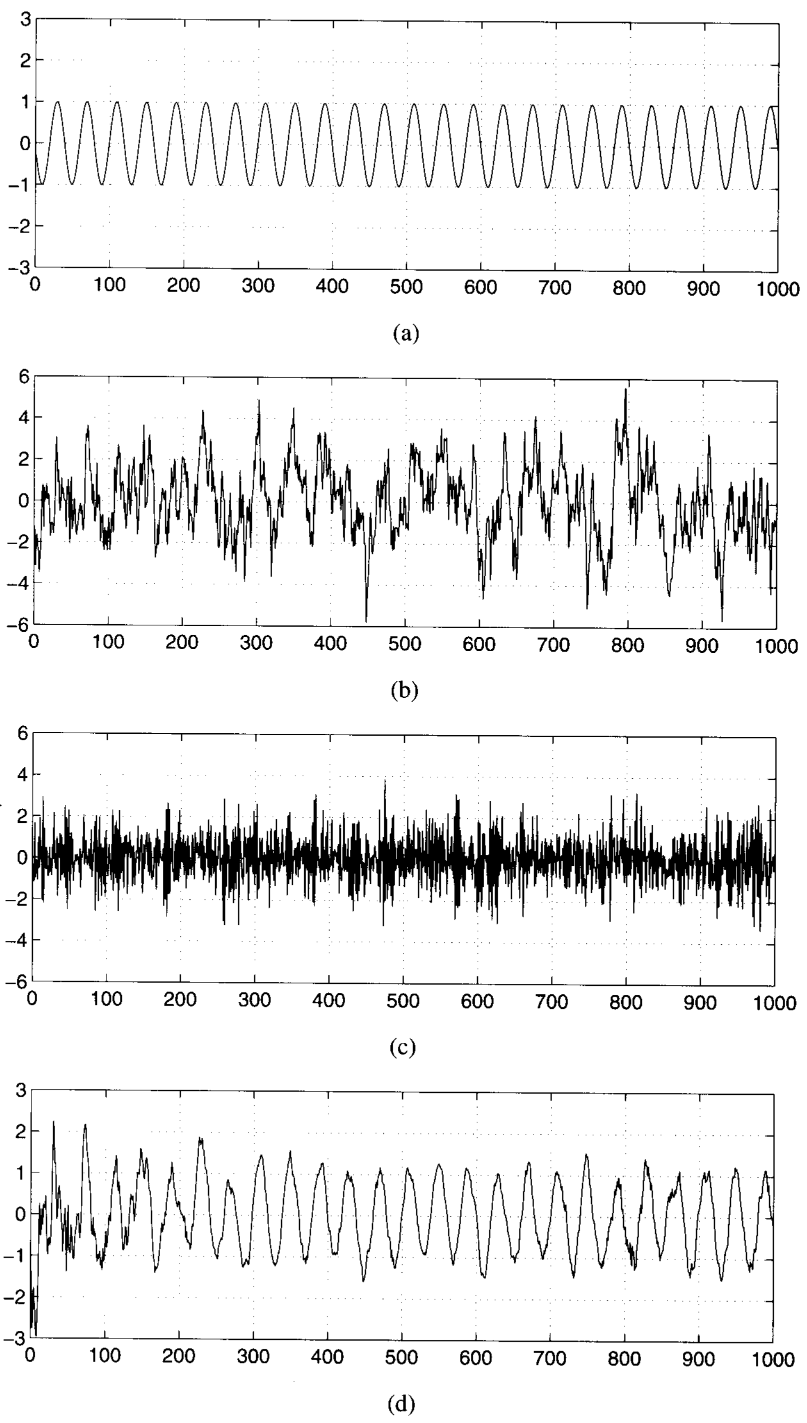


Figure 9.15 *Noise cancellation example. (a) A sinusoid that is to be estimated. (b) Noise corrupted sinusoid. (c) The reference signal used by the secondary sensor. (d) The output of a sixth-order NLMS adaptive noise canceller with a normalized step size $\beta = 0.25$.*

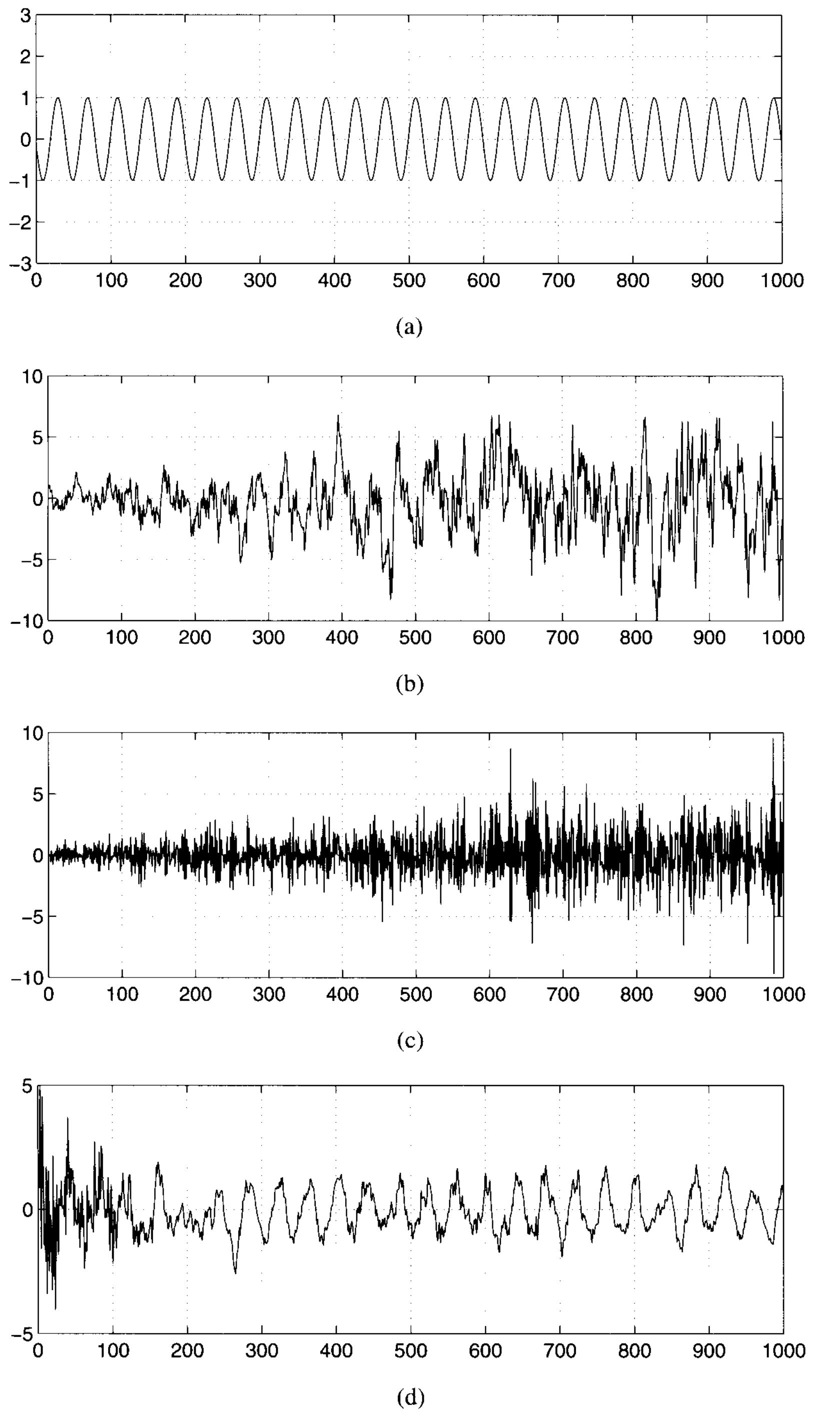
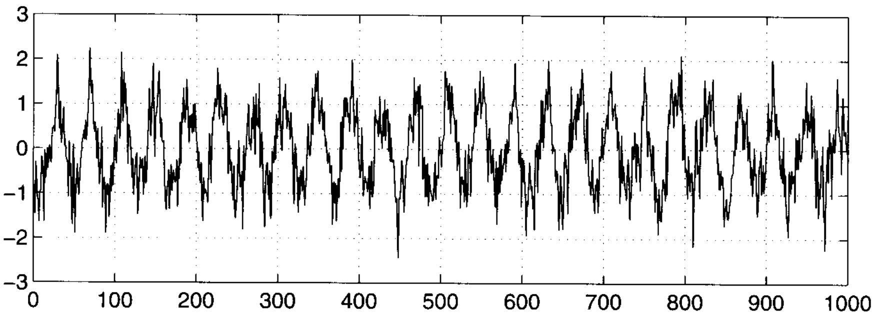
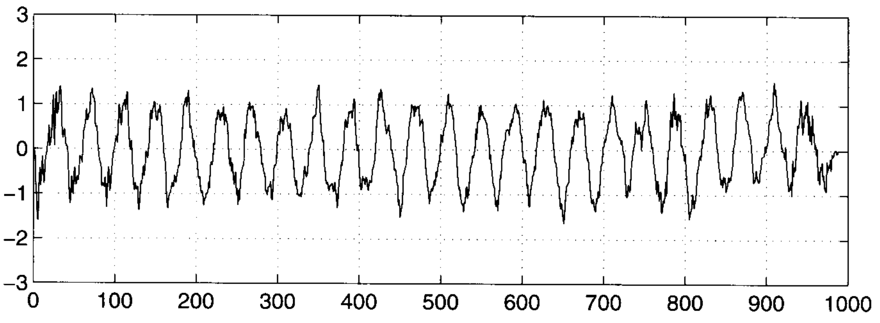


Figure 9.16 *Noise cancellation of nonstationary noise. (a) Desired signal that is to be estimated. (b) Noise corrupted sinusoid. (c) The reference signal used by secondary sensor. (d) The output of a 12th-order NLMS adaptive noise canceller with a normalized step size $\beta = 0.25$.*



(a)



(b)

Figure 9.17 *Noise cancellation without a reference signal. (a) The noisy process $x(n)$ and (b) the output of a 12th-order NLMS adaptive noise canceller with $\beta = 0.25$ and $n_0 = 25$.*

If $d(n)$ and $v_1(n)$ are uncorrelated, then

$$E\{v_1(n)x(n-k)\} = E\{v_1(n)d(n-k)\} + E\{v_1(n)v_1(n-k)\} = 0 \quad ; \quad |k| > k_0$$

Therefore, if $n_0 > k_0$ then the delayed process $x(n - n_0)$ will be uncorrelated with the noise $v_1(n)$, and correlated with $d(n)$ (with the assumption that $d(n)$ is a broadband process). Thus, $x(n - n_0)$ may be used as a reference signal to estimate $d(n)$ as illustrated in Fig. 9.14*c*. In contrast to the adaptive noise canceller in Fig. 9.14*b*, note that the adaptive filter in Fig. 9.14*c* produces an estimate of the broadband process, $d(n)$, and the error $e(n)$ corresponds to an estimate of the noise $v_1(n)$. As an example of using an adaptive noise canceller without a reference, let $x(n) = d(n)+v_1(n)$ where $d(n)$ is a sinusoid of frequency $\omega_0 = 0.05\pi$, and let $v_1(n)$ be the AR(1) process defined in Eq. (9.52) where $g(n)$ white noise with a variance of $\sigma_g^2 = 0.25$. Shown in Fig. 9.17*a* is the noisy process $x(n) = d(n) + v_1(n)$ and in Fig. 9.17*b* is the output of the adaptive filter using the NLMS algorithm with a normalized step size of $\beta = 0.25$, and a reference signal that is obtained by delaying $x(n)$ by $n_0 = 25$ samples.

9.2.6 Other LMS-Based Adaptive Filters

In addition to the NLMS algorithm, a number of other modifications to the LMS algorithm have been proposed and studied. Each of these modifications attempts to improve one or more of the properties of the LMS algorithm. In this section, we provide a brief overview of some of these modifications, including the leaky LMS algorithm for dealing with the problems that occur when the autocorrelation matrix is singular, the block LMS algorithm and the sign algorithms for increasing the computational efficiency of the LMS algorithm, and the variable step-size (VS) algorithms for improving the speed of convergence.

The Leaky LMS Algorithm. When the input process to an adaptive filter has an autocorrelation matrix with zero eigenvalues, the LMS adaptive filter has one or more modes that are undriven and undamped. For example, it follows from Eq. (9.39) that if $\lambda_k = 0$, then

$$E\{u_n(k)\} = u_0(k)$$

which does not decay to zero with n. Since it is possible for these undamped modes to become unstable, it is important to stabilize the LMS adaptive filter by forcing these modes to zero [33]. One way to accomplish this is to introduce a leakage coefficient γ into the LMS algorithm as follows:

$$\boxed{\mathbf{w}_{n+1} = (1 - \mu\gamma)\mathbf{w}_n + \mu e(n)\mathbf{x}^*(n)} \tag{9.53}$$

where $0 < \gamma \ll 1$. The effect of this *leakage coefficient* is to force the filter coefficients to zero if either the error $e(n)$ or the input $x(n)$ become zero, and to force any undamped modes of the system to zero.

The properties of the leaky LMS algorithm may be derived by examining the behavior of $E\{\mathbf{w}_n\}$. If we substitute Eq. (9.7) for $e(n)$ into Eq. (9.53), then we have

$$\mathbf{w}_{n+1} = \left[\mathbf{I} - \mu\left\{\mathbf{x}^*(n)\mathbf{x}^T(n) + \gamma\mathbf{I}\right\}\right]\mathbf{w}_n + \mu d(n)\mathbf{x}^*(n)$$

Taking the expected value and using the independence assumption yields

$$E\{\mathbf{w}_{n+1}\} = \left[\mathbf{I} - \mu\left\{\mathbf{R}_x + \gamma\mathbf{I}\right\}\right]E\{\mathbf{w}_n\} + \mu\mathbf{r}_{dx} \tag{9.54}$$

Comparing Eq. (9.54) to Eq. (9.38) we see that the autocorrelation matrix $\mathbf{R}_x$ in the LMS algorithm has been replaced with $\mathbf{R}_x + \gamma\mathbf{I}$. Therefore, the coefficient leakage term effectively adds white noise to $x(n)$ by adding γ to the main diagonal of the autocorrelation matrix. Since the eigenvalues of $\mathbf{R}_x + \gamma\mathbf{I}$ are $\lambda_k + \gamma$ and since $\lambda_k \geq 0$, then none of the modes of the leaky LMS algorithm will be undamped. In addition, the constraint on the step size μ for convergence in the mean becomes

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

The drawback to the leaky LMS algorithm is that, for stationary processes, the steady-state solution will be biased. Specifically, note that if $\mathbf{w}_n$ converges in the mean then

$$\lim_{n\to\infty} E\{\mathbf{w}_n\} = \left(\mathbf{R}_x + \gamma\mathbf{I}\right)^{-1}\mathbf{r}_{dx}$$

Thus, the leakage coefficient introduces a bias into the steady-state solution.

It is interesting to note that the leaky LMS algorithm may also be derived by using the LMS gradient descent algorithm to minimize the error

$$\xi(n) = |e(n)|^2 + \gamma\|\mathbf{w}_n\|^2$$

Specifically, since the gradient of $\xi(n)$ is

$$\nabla\xi(n) = -e(n)\mathbf{x}^*(n) + \gamma\mathbf{w}_n$$

then the gradient descent algorithm becomes

$$\mathbf{w}_{n+1} = \mathbf{w}_n - \mu\nabla\xi(n) = \mathbf{w}_n - \mu\gamma\mathbf{w}_n + \mu e(n)\mathbf{x}^*(n)$$

which is the same as Eq. (9.53).

LMS Algorithms With Reduced Complexity. In spite of the computational efficiency of the LMS algorithm, additional simplifications may be necessary in some applications, such as high speed digital communications. There are several approaches that may be used to reduce the computational requirements of the LMS algorithm. One of these is the block LMS algorithm, which is identical to the LMS algorithm except that the filter coefficients are updated only once for each block of L samples [9]. In other words, the filter coefficients are held constant over each block of L samples, and the filter output $y(n)$ and the error $e(n)$ for each value of n within the block are calculated using the filter coefficients for that block. Then, at the end of each block, the coefficients are updated using an average of the L gradient estimates over the block. Thus, the update equation for the coefficients in the kth block is

$$\mathbf{w}_{(k+1)L} = \mathbf{w}_{kL} + \mu\frac{1}{L}\sum_{l=0}^{L-1} e(kL+l)\mathbf{x}^*(kL+l)$$

where the output for the kth block is

$$y(kL+l) = \mathbf{w}_{kL}^T\mathbf{x}(kL+l) \quad ; \quad l = 0, 1, \ldots, L-1$$

and the error is

$$e(kL+l) = d(kL+l) - \mathbf{w}_{kL}^T\mathbf{x}(kL+l) \quad ; \quad l = 0, 1, \ldots, L-1$$

Since $y(n)$ over each block of L values is the convolution of the weight vector $\mathbf{w}_{kL}$ with a block of input samples, the efficiency of the block LMS algorithm comes from using an FFT to perform this block convolution [12].

Another set of simplifications to the LMS algorithm are found in the *sign algorithms*. In these algorithms, the LMS coefficient update equation is modified by applying the sign operator to either the error $e(n)$, the data $x(n)$, or both the error and the data. For example, assuming that $x(n)$ and $d(n)$ are real-valued processes, the *sign-error* algorithm is

$$\boxed{\mathbf{w}_{n+1} = \mathbf{w}_n + \mu\,\mathrm{sgn}\{e(n)\}\mathbf{x}(n)} \qquad (9.55)$$

where

$$\mathrm{sgn}\{e(n)\} = \begin{cases} 1 & ; \quad e(n) > 0 \\ 0 & ; \quad e(n) = 0 \\ -1 & ; \quad e(n) < 0 \end{cases}$$

Note that the sign-error algorithm may be viewed as the result of applying a two-level quantizer to the error. The simplification in the sign-error algorithm comes when the step size is chosen to be a power of two, $\mu = 2^{-l}$. In this case, the coefficient update equation may be implemented using $p+1$ data shifts instead of $p+1$ multiplies. Compared to the LMS algorithm, the sign-error algorithm uses a noisy estimate of the gradient. Since replacing $e(n)$ with the sign of the error changes only the magnitude of the correction that is used to update $\mathbf{w}_n$ and does not alter the direction, the sign-error algorithm is equivalent to the LMS algorithm with a step size that is inversely proportional to the magnitude of the error.

It is interesting to note that the sign error algorithm may also be derived by using the LMS gradient descent algorithm to minimize the error

$$\xi(n) = |e(n)|$$

To show this, note that

$$\frac{\partial |e(n)|}{\partial w_n(k)} = \text{sgn}\{e(n)\}\frac{\partial e(n)}{\partial w_n(k)} = \text{sgn}\{e(n)\}x(n-k)$$

Therefore, the gradient vector is

$$\nabla\xi(n) = \text{sgn}\{e(n)\}\mathbf{x}(n)$$

and the result follows. Thus, the sign-error algorithm is sometimes referred to as the Least Mean Absolute Value (LMAV) algorithm [3].

Along the same lines, instead of using the sign of the error, the computational requirements of the LMS algorithm may be simplified by using the sign of the data as follows:

$$\boxed{\mathbf{w}_{n+1} = \mathbf{w}_n + \mu\, e(n)\text{sgn}\{\mathbf{x}(n)\}} \tag{9.56}$$

which is the *sign-data algorithm.* Note that, unlike the sign-error algorithm, the sign-data algorithm alters the direction of the update vector. As a result, the sign-data algorithm is generally less robust than the sign-error algorithm. In fact, examples have been found in which the coefficients diverge using the sign-data algorithm while converging using the LMS algorithm [28]. Note that the kth coefficient in the sign of the data vector may be written as follows:

$$\text{sgn}\{x(n-k)\} = \frac{x(n-k)}{|x(n-k)|}$$

Therefore, as in the normalized LMS algorithm where the weight vector is normalized by $\|\mathbf{x}(n)\|^2$, the sign-data algorithm individually normalizes each coefficient of the weight vector. Thus, the sign-data algorithm may be written as

$$w_{n+1}(k) = w_n(k) + \frac{\mu}{|x(n-k)|}\, e(n)x(n-k)$$

which is an LMS algorithm that has a different (time-varying) step size for each coefficient in the weight vector.

Finally, quantizing both the error and the data leads to the *sign-sign* algorithm, which has a coefficient update equation given by

$$\boxed{\mathbf{w}_{n+1} = \mathbf{w}_n + \mu\, \text{sgn}\{e(n)\}\text{sgn}\{\mathbf{x}(n)\}} \tag{9.57}$$

In this algorithm, the coefficients $w_n(k)$ are updated by either adding or subtracting a constant μ. For stability, a leakage term is often introduced into the sign-sign algorithm [3] giving an update equation of the form

$$\mathbf{w}_{n+1} = (1 - \mu\gamma)\mathbf{w}_n + \mu \operatorname{sgn}\{e(n)\}\operatorname{sgn}\{\mathbf{x}(n)\}$$

Generally, the sign-sign algorithm is slower to converge than the LMS adaptive filter and has a larger excess mean-square error [5,11]. Nevertheless, the extreme simplicity of the update algorithm has made it a popular algorithm and has, in fact, been adopted in the international CCITT standard for 32kbps ADPCM [10].[5]

Variable Step-Size Algorithms. In selecting the step size μ in the LMS algorithm, there is a tradeoff between the rate of convergence, the amount of excess mean-square error, and the ability of the filter to track signals as their statistics change. Ideally, when the adaptation begins and $\mathbf{w}_n$ is far from the optimum solution, the step size should be large in order to move the weight vector rapidly toward the desired solution. Then, as the filter begins to converge in the mean to the steady-state solution, the step size should be decreased in order to reduce the excess mean-square error. This suggests the possibility of using a variable step size in the LMS adaptive filter. The difficulty, however, is in specifying a set of rules for changing the step size in such a way that the adaptive filter has a small excess mean-square error while, at the same time, maintaining the ability of the filter to respond quickly to changes in the signal statistics.

Assuming that $x(n)$ and $d(n)$ are real-valued processes, one approach that has been proposed for changing the step size, referred to as the *Variable Step* (VS) algorithm [19], is to use a coefficient update equation of the form

$$w_{n+1}(k) = w_n(k) + \mu_n(k)e(n)x(n-k)$$

where $\mu_n(k)$ are step sizes that are adjusted independently for each coefficient. The rules for adjusting $\mu_n(k)$ are tied to the rate at which the gradient estimate changes sign. With an estimated gradient given by

$$\nabla e^2(n) = -2e(n)x(n-k)$$

these rules are based on the premise that if the sign of $e(n)x(n-k)$ is changing frequently, then the coefficient $w_n(k)$ should be close to its optimum value where the gradient is equal to zero. On the other hand, if the sign is not changing very often, then the coefficient $w_n(k)$ is probably not close to its optimum value. Therefore, $\mu_n(k)$ is decreased by a constant, c_1, if m_1 successive sign changes are observed in $e(n)x(n-k)$, whereas $\mu_n(k)$ is increased by a constant, c_2, if $e(n)x(n-k)$ has the same sign for m_2 successive updates. In addition, hard limits are placed on the step size,

$$\mu_{\min} < \mu_n(k) < \mu_{\max}$$

in order to ensure that the VS algorithm converges in the mean with only a modest increase in computation, the VS algorithm may result in a considerable improvement in the convergence rate [19].

[5]See "32 kbit/s Adaptive differential pulse code modulation (ADPCM)," in the CCITT standard recommendation G.721, Melbourne 1988.