$$\begin{aligned}\mathbf{r}_{xd} &= E[\mathbf{x}(i)d(i)] \\ &= \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}(i)d(i) \\ &= \lim_{n\to\infty} \frac{1}{n} \mathbf{X}^T(n)\mathbf{d}(n)\end{aligned} \tag{3.31}$$

where E denotes the statistical expectation operator. Accordingly, we may reformulate the linear least-squares solution of Eq. (3.27) as follows:

$$\begin{aligned}\mathbf{w}_o &= \lim_{n\to\infty} \mathbf{w}(n+1) \\ &= \lim_{n\to\infty} (\mathbf{X}^T(n)\mathbf{X}(n))^{-1}\mathbf{X}^T(n)\mathbf{d}(n) \\ &= \lim_{n\to\infty} \frac{1}{n} (\mathbf{X}^T(n)\mathbf{X}(n))^{-1} \lim_{n\to\infty} \frac{1}{n} \mathbf{X}^T(n)\mathbf{d}(n) \\ &= \mathbf{R}_x^{-1}\mathbf{r}_{xd}\end{aligned} \tag{3.32}$$

where $\mathbf{R}_x^{-1}$ is the inverse of the correlation matrix $\mathbf{R}_x$. The weight vector $\mathbf{w}_o$ is called the *Wiener solution* to the linear optimum filtering problem in recognition of the contributions of Norbert Wiener to this problem (Widrow and Stearns, 1985; Haykin, 1996). Accordingly, we may make the following statement:

> *For an ergodic process, the linear least-squares filter asymptotically approaches the Wiener filter as the number of observations approaches infinity.*

Designing the Wiener filter requires knowledge of the second-order statistics: the correlation matrix $\mathbf{R}_x$ of the input vector $\mathbf{x}(n)$ and the cross-correlation vector $\mathbf{r}_{xd}$ between $\mathbf{x}(n)$ and the desired response $d(n)$. However, this information is not available in many important situations encountered in practice. We may deal with an unknown environment by using a *linear adaptive filter*, adaptive in the sense that the filter is able to adjust its free parameters in response to statistical variations in the environment. A highly popular algorithm for doing this kind of adjustment on a continuing basis is the least-mean-square algorithm, which is intimately related to the Wiener filter.

3.5 LEAST-MEAN-SQUARE ALGORITHM

The *least-mean-square (LMS) algorithm* is based on the use of *instantaneous values* for the cost function, namely,

$$\mathscr{E}(\mathbf{w}) = \frac{1}{2} e^2(n) \tag{3.33}$$

where $e(n)$ is the error signal measured at time n. Differentiating $\mathscr{E}(\mathbf{w})$ with respect to the weight vector $\mathbf{w}$ yields

$$\frac{\partial \mathscr{E}(\mathbf{w})}{\partial \mathbf{w}} = e(n) \frac{\partial e(n)}{\partial \mathbf{w}} \tag{3.34}$$

As with the linear least-squares filter, the LMS algorithm operates with a linear neuron so we may express the error signal as

$$e(n) = d(n) - \mathbf{x}^T(n)\mathbf{w}(n) \tag{3.35}$$

Hence,

$$\frac{\partial e(n)}{\partial \mathbf{w}(n)} = -\mathbf{x}(n)$$

and

$$\frac{\partial \mathscr{E}(\mathbf{w})}{\partial \mathbf{w}(n)} = -\mathbf{x}(n)e(n)$$

Using this latter result as an *estimate* for the gradient vector, we may write

$$\hat{\mathbf{g}}(n) = -\mathbf{x}(n)e(n) \tag{3.36}$$

Finally, using Eq. (3.36) for the gradient vector in Eq. (3.12) for the method of steepest descent, we may formulate the LMS algorithm as follows:

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \eta\mathbf{x}(n)e(n) \tag{3.37}$$

where η is the learning-rate parameter. The feedback loop around the weight vector $\hat{\mathbf{w}}(n)$ in the LMS algorithm behaves like a *low-pass filter,* passing the low frequency components of the error signal and attenuating its high frequency components (Haykin, 1996). The average time constant of this filtering action is inversely proportional to the learning-rate parameter η. Hence, by assigning a small value to η, the adaptive process will progress slowly. More of the past data are then remembered by the LMS algorithm, resulting in a more accurate filtering action. In other words, the inverse of the learning-rate parameter η is a measure of the *memory* of the LMS algorithm.

In Eq. (3.37) we have used $\hat{\mathbf{w}}(n)$ in place of $\mathbf{w}(n)$ to emphasize the fact that the LMS algorithm produces an *estimate* of the weight vector that would result from the use of the method of steepest descent. As a consequence, in using the LMS algorithm we sacrifice a distinctive feature of the steepest descent algorithm. In the steepest descent algorithm the weight vector $\mathbf{w}(n)$ follows a well-defined trajectory in weight space for a prescribed η. In contrast, in the LMS algorithm the weight vector $\hat{\mathbf{w}}(n)$ traces a random trajectory. For this reason, the LMS algorithm is sometimes referred to as a "stochastic gradient algorithm." As the number of iterations in the LMS algorithm approaches infinity, $\hat{\mathbf{w}}(n)$ performs a random walk (Brownian motion) about the Wiener solution $\mathbf{w}_o$. The important point is the fact that, unlike the method of steepest descent, the LMS algorithm does *not* require knowledge of the statistics of the environment.

A summary of the LMS algorithm is presented in Table 3.1, which clearly illustrates the simplicity of the algorithm. As indicated in this table, for the *initialization* of the algorithm, it is customary to set the initial value of the weight vector in the algorithm equal to zero.

TABLE 3.1 Summary of the LMS Algorithm

Training Sample: Input signal vector = $\mathbf{x}(n)$
Desired response = $d(n)$

User-selected parameter: η

Initialization. Set $\hat{\mathbf{w}}(0) = \mathbf{0}$.

Computation. For $n = 1, 2, \ldots$, compute

$$e(n) = d(n) - \hat{\mathbf{w}}^T(n)\mathbf{x}(n)$$
$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \eta\,\mathbf{x}(n)e(n)$$

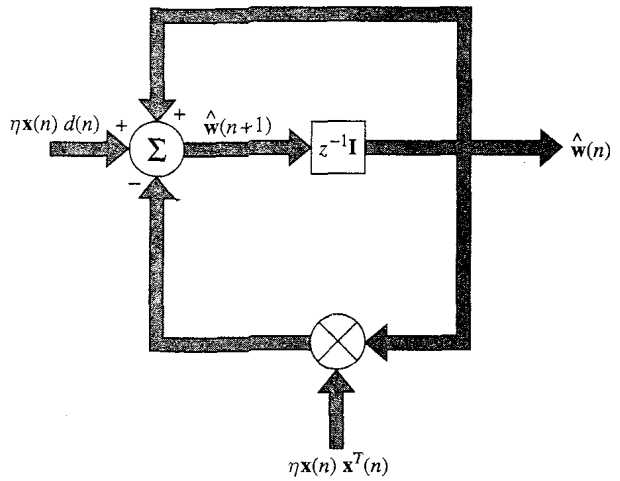

FIGURE 3.3 Signal-flow graph representation of the LMS algorithm.

Signal-Flow Graph Representation of the LMS Algorithm

By combining Eqs. (3.35) and (3.37) we may express the evolution of the weight vector in the LMS algorithm as follows:

$$\begin{aligned}\hat{\mathbf{w}}(n+1) &= \hat{\mathbf{w}}(n) + \eta\mathbf{x}(n)[d(n) - \mathbf{x}^T(n)\hat{\mathbf{w}}(n)] \\ &= [\mathbf{I} - \eta\mathbf{x}(n)\mathbf{x}^T(n)]\hat{\mathbf{w}}(n) + \eta\mathbf{x}(n)d(n)\end{aligned} \tag{3.38}$$

where $\mathbf{I}$ is the identity matrix. In using the LMS algorithm, we recognize that

$$\hat{\mathbf{w}}(n) = z^{-1}[\hat{\mathbf{w}}(n+1)] \tag{3.39}$$

where z^{-1} is the *unit-delay operator*, implying storage. Using Eqs. (3.38) and (3.39), we may thus represent the LMS algorithm by the signal-flow graph depicted in Fig. 3.3. This signal-flow graph reveals that the LMS algorithm is an example of a *stochastic feedback system*. The presence of feedback has a profound impact on the convergence behavior of the LMS algorithm.

Convergence Considerations of the LMS Algorithm

From control theory we know that the stability of a feedback system is determined by the parameters that constitute its feedback loop. From Fig. 3.3 we see that it is the lower feedback loop that adds variability to the behavior of the LMS algorithm. In par-

ticular, there are two distinct quantities, the learning-rate parameter η and the input vector $\mathbf{x}(n)$, that determine the transmittance of this feedback loop. We therefore deduce that the convergence behavior (i.e., stability) of the LMS algorithm is influenced by the statistical characteristics of the input vector $\mathbf{x}(n)$ and the value assigned to the learning-rate parameter η. Casting this observation in a different way, we may state that for a specified environment that supplies the input vector $\mathbf{x}(n)$, we have to exercise care in the selection of the learning-rate parameter η for the LMS algorithm to be convergent.

The first criterion for convergence of the LMS algorithm is *convergence of the mean*, described by

$$E[\hat{\mathbf{w}}(n)] \rightarrow \mathbf{w}_o \qquad \text{as } n \rightarrow \infty \tag{3.40}$$

where $\mathbf{w}_o$ is the Wiener solution. Unfortunately, such a convergence criterion is of little practical value, since a sequence of zero-mean, but otherwise arbitrary, random vectors converges in this sense.

From a practical point of view, the convergence issue that really matters is *convergence in the mean square*, described by

$$E[e^2(n)] \rightarrow \text{constant as } n \rightarrow \infty \tag{3.41}$$

Unfortunately, a detailed convergence analysis of the LMS algorithm in the mean square is rather complicated. To make the analysis mathematically tractable, the following assumptions are usually made:

1. The successive input vectors $\mathbf{x}(1)$, $\mathbf{x}(2)$, ... are statistically independent of each other.
2. At time step n, the input vector $\mathbf{x}(n)$ is statistically independent of all previous samples of the desired response, namely $d(1), d(2), \ldots, d(n-1)$.
3. At time step n, the desired response $d(n)$ is dependent on $\mathbf{x}(n)$, but statistically independent of all previous values of the desired response.
4. The input vector $\mathbf{x}(n)$ and desired response $d(n)$ are drawn from Gaussian-distributed populations.

A statistical analysis of the LMS algorithm so based is called the *independence theory* (Widrow et al., 1976).

By invoking the elements of independence theory and assuming that the learning-rate parameter η is sufficiently small, it is shown in Haykin (1996) that the LMS is convergent in the mean square provided that η satisfies the condition

$$0 < \eta < \frac{2}{\lambda_{\max}} \tag{3.42}$$

where $\lambda_{\max}$ is the *largest eigenvalue* of the correlation matrix $\mathbf{R}_x$. In typical applications of the LMS algorithm, however, knowledge of $\lambda_{\max}$ is not available. To overcome this difficulty, the *trace* of $\mathbf{R}_x$ may be taken as a conservative estimate for $\lambda_{\max}$, in which case the condition of Eq. (3.42) may be reformulated as

$$0 < \eta < \frac{2}{\text{tr}[\mathbf{R}_x]} \tag{3.43}$$

where $\text{tr}[\mathbf{R}_x]$ denotes the trace of matrix $\mathbf{R}_x$. By definition, the trace of a square matrix is equal to the sum of its diagonal elements. Since each diagonal element of the correlation matrix $\mathbf{R}_x$ equals the mean-square value of the corresponding sensor input, we may restate the condition for convergence of the LMS algorithm in the mean square as follows:

$$0 < \eta < \frac{2}{\text{sum of mean-square values of the sensor inputs}} \tag{3.44}$$

Provided the learning-rate parameter satisfies this condition, the LMS algorithm is also assured of convergence of the mean. That is, convergence in the mean square implies convergence of the mean, but the converse is not necessarily true.

Virtues and Limitations of the LMS Algorithm

An important virtue of the LMS algorithm is its simplicity, as exemplified by the summary of the algorithm presented in Table 3.1. Moreover, the LMS algorithm is model independent and therefore *robust*, which means that small model uncertainty and small disturbances (i.e., disturbances with small energy) can only result in small estimation errors (error signals). In precise mathematical terms, the LMS algorithm is optimal in accordance with the H^∞ *(or minimax) criterion* (Hassibi et al., 1993, 1996). The basic philosophy of optimality in the H^∞ sense is to cater to the worst-case scenario[4]:

If you do not know what you are up against, plan for the worst and optimize.

For a long time the LMS algorithm was regarded as an instantaneous approximation to the gradient-descent algorithm. However, the H^∞ optimality of LMS provides this widely used algorithm with a rigorous footing. In particular, it explains its ability to work satisfactorily in a stationary as well as in a nonstationary environment. By a "nonstationary" environment we mean one where the statistics vary with time. In such an environment, the optimum Wiener solution takes on a time-varying form, and the LMS algorithm now has the additional task of *tracking* variations in the parameters of the Wiener filter.

The primary limitations of the LMS algorithm are its slow rate of convergence and sensitivity to variations in the eigenstructure of the input (Haykin, 1996). The LMS algorithm typically requires a number of iterations equal to about 10 times the dimensionality of the input space for it to reach a steady-state condition. The slow rate of convergence becomes particularly serious when the dimensionality of the input space becomes high. As for sensitivity to changes in environmental conditions, the LMS algorithm is particularly sensitive to variations in the *condition number* or *eigenvalue spread* of the correlation matrix $\mathbf{R}_x$ of the input vector $\mathbf{x}$. The condition number of $\mathbf{R}_x$, denoted by $\chi(\mathbf{R}_x)$, is defined by

$$\chi(\mathbf{R}_x) = \frac{\lambda_{\max}}{\lambda_{\min}} \tag{3.45}$$

where $\lambda_{\max}$ and $\lambda_{\min}$ are the maximum and minimum eigenvalues of the matrix $\mathbf{R}_x$, respectively. The sensitivity of the LMS algorithm to variations in the condition number $\chi(\mathbf{R}_x)$ becomes particularly acute when the training sample to which the input vector $\mathbf{x}(n)$ belongs is *ill conditioned*, that is, when the condition number $\chi(\mathbf{R}_x)$ is high.[5]

Note that in the LMS algorithm the *Hessian matrix,* defined as the second derivative of the cost function $\mathscr{E}(\mathbf{w})$ with respect to $\mathbf{w}$, is equal to the correlation matrix $\mathbf{R}_x$; see Problem 3.8. Thus, in the discussion presented here, we could have just as well spoken in terms of the Hessian as the correlation matrix $\mathbf{R}_x$.

3.6 LEARNING CURVES

An informative way of examining the convergence behavior of the LMS algorithm, or an adaptive filter in general, is to plot the *learning curve* of the filter under varying environmental conditions. The learning curve is a *plot of the mean-square value of the estimation error,* $\mathscr{E}_{av}(n)$, *versus the number of iterations,* n.

Imagine an experiment involving an *ensemble* of adaptive filters, with each filter operating under the control of a specific algorithm. It is assumed that the details of the algorithm, including initialization, are the same for all the filters. The differences between the filters arise from the *random* manner in which the input vector $\mathbf{x}(n)$ and the desired response $d(n)$ are drawn from the available training sample. For each filter we plot the squared value of the estimation error (i.e., the difference between the desired response and the actual filter output) versus the number of iterations. A *sample* learning curve so obtained consists of *noisy* exponentials, the noise being due to the inherently stochastic nature of the adaptive filter. To compute the *ensemble-averaged learning curve* (i.e., plot of $\mathscr{E}_{av}(n)$ versus n), we take the average of these sample learning curves over the ensemble of adaptive filters used in the experiment, thereby smoothing out the effects of noise.

Assuming that the adaptive filter is stable, we find that the ensemble-averaged learning curve starts from a large value $\mathscr{E}_{av}(0)$ determined by the initial conditions, then decreases at some rate depending on the type of filter used, and finally converges to a steady-state value $\mathscr{E}_{av}(\infty)$, as illustrated in Fig. 3.4. On the basis of this learning curve we may define the *rate of convergence* of the adaptive filter as the number of iterations, n, required for $\mathscr{E}_{av}(n)$ to decrease to some arbitrarily chosen value, such as 10 percent of the initial value $\mathscr{E}_{av}(0)$.

Another useful characteristic of an adaptive filter that is deduced from the ensemble-averaged learning curve is the *misadjustment*, denoted by $\mathcal{M}$. Let $\mathscr{E}_{min}$ denote the minimum mean-square error produced by the Wiener filter, designed on the basis of known values of the correlation matrix $\mathbf{R}_x$ and cross-correlation vector $\mathbf{r}_{xd}$. We may define the *misadjustment* for the adaptive filter as follows (Widrow and Stearns, 1985; Haykin, 1996):

$$\begin{aligned}\mathcal{M} &= \frac{\mathscr{E}(\infty) - \mathscr{E}_{min}}{\mathscr{E}_{min}} \\ &= \frac{\mathscr{E}(\infty)}{\mathscr{E}_{min}} - 1\end{aligned} \tag{3.46}$$

The misadjustment $\mathcal{M}$ is a dimensionless quantity, providing a measure of how close the adaptive filter 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 of the algorithm. It is customary to express the misadjustment $\mathcal{M}$ as a percentage. Thus, for example, a misadjustment of 10 percent means that the adaptive filter produces a mean-square error

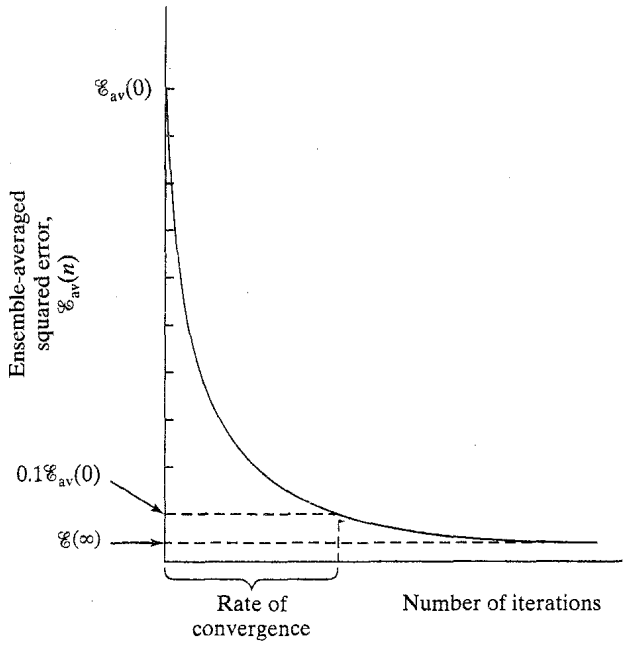

FIGURE 3.4 Idealized learning curve of the LMS algorithm.

(after adaptation is completed) that is 10 percent greater than the minimum mean-square error $\mathscr{E}_{min}$ produced by the corresponding Wiener filter. Such performance is ordinarily considered to be satisfactory in practice.

Another important characteristic of the LMS algorithm is the *settling time*. However, there is no unique definition for the settling time. We may, for example, approximate the learning curve by a single exponential with *average time constant* τ_{av}, and so use τ_{av} as a rough measure of the settling time. The smaller the value of τ_{av} is, the faster the settling time will be (i.e., the faster the LMS algorithm will converge to a "steady-state" condition).

To a good degree of approximation, the misadjustment $\mathcal{M}$ of the LMS algorithm is directly proportional to the learning-rate parameter η, whereas the average time constant τ_{av} is inversely proportional to the learning-rate parameter η (Widrow and Stearns, 1985; Haykin, 1996). We therefore have conflicting results in the sense that if the learning-rate parameter is reduced so as to reduce the misadjustment, then the settling time of the LMS algorithm is increased. Conversely, if the learning-rate parameter is increased so as to accelerate the learning process, then the misadjustment is increased. Careful attention must be given to the choice of the learning parameter η in the design of the LMS algorithm in order to produce a satisfactory overall performance.

3.7 LEARNING-RATE ANNEALING SCHEDULES

The difficulties encountered with the LMS algorithm may be attributed to the fact that the learning-rate parameter is maintained constant throughout the computation, as shown by