PART 3
Linear Adaptive Filtering

Part III, by far the largest portion of the book, consists of Chapters 8 through 17. It is devoted to a detailed treatment of linear finite-duration impulse response (FIR) adaptive filters.

In Chapter 8, we develop the method of steepest descent for computing the tap-weight vector of the Wiener filter in a recursive fashion. In Chapter 9, we use the method of steepest descent to derive the least-mean-square (LMS) algorithm and study its important characteristics. Chapter 10 discusses frequency-domain adaptive filters, designed to extend the utility of the LMS algorithm.

Chapter 11 covers the fundamentals of linear least-squares estimation. In this chapter we also develop the singular value-decomposition (SVD), which provides a powerful tool for solving the linear least-square estimation problem and related ones. Chapter 12 discusses the unitary rotations that are basic to the design of square-root Kalman and least-squares filters.

In Chapter 13, we derive the standard recursive least squares (RLS) algorithm, which may be viewed as a special case of the Kalman filter. In Chapter 14, we study square-root variants of the RLS algorithm. Chapter 15 discusses order-recursive least-squares filters, built around the lattice structure.

In Chapter 16 we study the tracking performance of linear adaptive filters. Part III finishes with Chapter 17 on finite precision (numerical) effects that arise when linear adaptive filters are implemented on a general-purpose or special-purpose digital machine.

CHAPTER

8

Method of Steepest Descent

In this chapter we begin our study of gradient-based adaptation by describing an old optimization technique known as the *method of steepest descent*. This method is basic to the understanding of the various ways in which gradient-based adaptation is implemented in practice. The method of steepest descent is *recursive* in the sense that starting from some initial (arbitrary) value for the tap-weight vector, it improves with the increased number of iterations. The final value so computed for the tap-weight vector converges to the Wiener solution. The important point to note is that the method of steepest descent is descriptive of a *deterministic feedback system* that finds the minimum point of the ensemble-averaged error-performance surface without knowledge of the surface itself. Accordingly, it provides some heuristics for writing the recursions that describe the least-mean-square (LMS) algorithm, an issue that is taken up in the next chapter.

8.1 SOME PRELIMINARIES

Consider a transversal filter with tap inputs $u(n)$, $u(n-1)$, . . . , $u(n-M+1)$ and a corresponding set of *tap weights* $w_0(n)$, $w_1(n)$, . . . , $w_{M-1}(n)$. The tap inputs represent samples drawn from a wide-sense stationary stochastic process of zero mean and correlation matrix $\mathbf{R}$. In addition to these inputs, the filter is supplied with a *desired response* $d(n)$ that provides a frame of reference for the optimum filtering action. Figure 8.1 depicts the filtering action described herein.

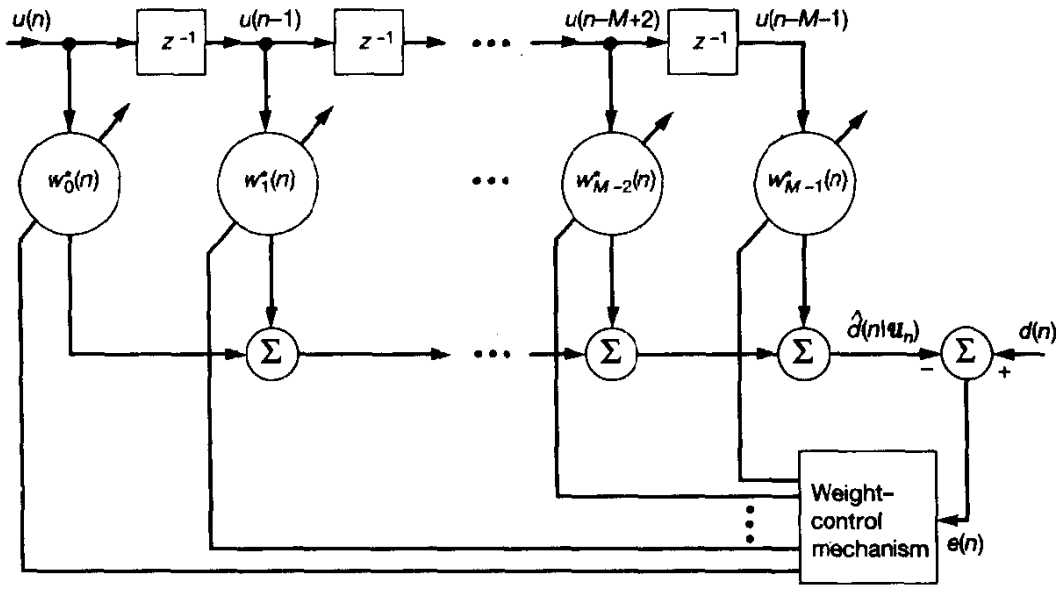


Figure 8.1 *Structure of adaptive transversal filter.*

The vector of tap inputs at time n is denoted by $\mathbf{u}(n)$, and the corresponding *estimate* of the desired response at the filter output is denoted by $\hat{d}(n|\mathcal{U}_n)$, where $\mathcal{U}_n$ is the space spanned by the tap inputs $u(n), u(n-1), \ldots, u(n-M+1)$. By comparing this estimate with the desired response $d(n)$, we produce an *estimation error* denoted by $e(n)$. We may thus write

$$\begin{aligned} e(n) &= d(n) - \hat{d}(n|\mathcal{U}_n) \\ &= d(n) - \mathbf{w}^H(n)\mathbf{u}(n) \end{aligned} \tag{8.1}$$

where the term $\mathbf{w}^H(n)\mathbf{u}(n)$ is the inner product of the tap-weight vector $\mathbf{w}(n)$ and the tap-input vector $\mathbf{u}(n)$. The expanded form of the tap-weight vector is described by

$$\mathbf{w}(n) = [w_0(n), w_1(n), \ldots, w_{M-1}(n)]^T \tag{8.2}$$

and that of the tap-input vector is described by

$$\mathbf{u}(n) = [u(n), u(n-1), \ldots, (n-M+1)]^T \tag{8.3}$$

If the tap-input vector $\mathbf{u}(n)$ and the desired $d(n)$ are jointly stationary, then the *mean-squared error* or *cost function* $J(n)$ at time n is a quadratic function of the tap-weight vector, so we may write [see Eq. (5.50)]

$$J(n) = \sigma_d^2 - \mathbf{w}^H(n)\mathbf{p} - \mathbf{p}^H\mathbf{w}(n) + \mathbf{w}^H(n)\mathbf{R}\mathbf{w}(n) \tag{8.4}$$

where σ_d^2 = variance of the desired response $d(n)$

$\mathbf{p}$ = cross-correlation vector between the tap-input vector $\mathbf{u}(n)$ and the desired response $d(n)$

$\mathbf{R}$ = correlation matrix of the tap-input vector $\mathbf{u}(n)$

Equation (8.4) defines the mean-squared error that would result if the tap-weight vector in the transversal filter were *fixed* at the value $\mathbf{w}(n)$. Since $\mathbf{w}(n)$ varies with time n, it is only natural that the mean-squared error varies with time n in a corresponding fashion, hence the use of $J(n)$ for the mean-squared error in Eq. (8.4). The variation of the mean-squared error $J(n)$ with time n signifies the fact that the estimation error process $e(n)$ is nonstationary.

We may visualize the dependence of the mean-squared error $J(n)$ on the elements of the tap-weight vector $\mathbf{w}(n)$ as a *bowl-shaped surface* with a unique minimum. We refer to this surface as the *error-performance surface* of the adaptive filter. The adaptive process has the task of continually seeking the *bottom* or *minimum point* of this surface. At the minimum point of the error-performance surface, the tap-weight vector takes on the optimum value $\mathbf{w}_o$, which is defined by the Wiener–Hopf equations (5.34), reproduced here for convenience,

$$\mathbf{R}\mathbf{w}_o = \mathbf{p} \tag{8.5}$$

The minimum mean-square error equals [see Eq. (5.49)]

$$J_{\min} = \sigma_d^2 - \mathbf{p}^H\mathbf{w}_o \tag{8.6}$$

8.2 STEEPEST-DESCENT ALGORITHM

The requirement that an adaptive transversal filter has to satisfy is to find a solution for its tap-weight vector that satisfies the Wiener–Hopf equations (8.5). One way of doing this would be to solve this system of equations by some analytical means. In general, this procedure is quite straightforward. However, it presents serious computational difficulties, especially when the filter contains a large number of tap weights and when the input data rate is high.

An alternative procedure is to use the *method of steepest descent*, which is one of the oldest methods of optimization.[1] To find the minimum value of the mean-squared error, $J_{\min}$, by the steepest-descent algorithm, we proceed as follows:

1. We begin with an initial value $\mathbf{w}(0)$ for the tap-weight vector, which provides an initial guess as to where the minimum point of the error-performance surface may be located. Unless some prior knowledge is available, $\mathbf{w}(0)$ is usually set equal to the null vector.
2. Using this initial or present guess, we compute the *gradient vector*, the real and imaginary parts of which are defined as the derivative of the mean-squared error

[1]The steepest-descent algorithm belongs to a family of *iterative methods of optimization* (Luenberger, 1969); it provides a method of searching a multidimensional performance surface. Another method in this family that may be used for this purpose is *Newton's algorithm*, which is primarily a method for finding the zeros of a function. In the context of adaptive filtering, the use of the method of steepest descent results in an algorithm that is much simpler, but slower, than Newton's method (Widrow and Stearns, 1985).

$J(n)$, evaluated with respect to the real and imaginary parts of the tap-weight vector $\mathbf{w}(n)$ at time n (i.e., the nth iteration).

3. We compute the next guess at the tap-weight vector by making a change in the initial or present guess in a direction opposite to that of the gradient vector.
4. We go back to step 2 and repeat the process.

It is intuitively reasonable that successive corrections to the tap-weight vector in the direction of the negative of the gradient vector (i.e., in the direction of the steepest descent of the error-performance surface) should eventually lead to the minimum mean-squared error J_{min}, at which point the tap-weight vector assumes its optimum value $\mathbf{w}_o$.

Let $\nabla J(n)$ denote the value of the *gradient vector* at time n. Let $\mathbf{w}(n)$ denote the value of the tap-weight vector at time n. According to the method of steepest descent, the updated value of the tap-weight vector at time $n + 1$ is computed by using the simple recursive relation

$$\mathbf{w}(n + 1) = \mathbf{w}(n) + \tfrac{1}{2}\mu[-\nabla J(n)] \tag{8.7}$$

where μ is a positive real-valued constant. The factor $\frac{1}{2}$ is used merely for the purpose of canceling a factor 2 that appears in the formula for $\nabla J(n)$; see Eq. (8.8).

From Chapter 4 we find that the gradient vector $\nabla J(n)$ is given by

$$\begin{aligned}\nabla J(n) &= \begin{bmatrix} \dfrac{\partial J(n)}{\partial a_0(n)} + j\dfrac{\partial J(n)}{\partial b_0(n)} \\ \dfrac{\partial J(n)}{\partial a_1(n)} + j\dfrac{\partial J(n)}{\partial b_1(n)} \\ \vdots \\ \dfrac{\partial J(n)}{\partial a_{M-1}(n)} + j\dfrac{\partial J(n)}{\partial b_{M-1}(n)} \end{bmatrix} \\ &= -2\mathbf{p} + 2\mathbf{R}\mathbf{w}(n)\end{aligned} \tag{8.8}$$

where, in the expanded column vector, $\partial J(n)/\partial a_k(n)$ and $\partial J(n)/\partial b_k(n)$ are the partial derivatives of the cost function $J(n)$ with respect to the real part $a_k(n)$ and the imaginary part $b_k(n)$ of the kth tap weight $w_k(n)$, respectively, with $k = 1, 2, \ldots, M - 1$. For the application of the steepest-descent algorithm, we assume that in Eq. (8.8) the correlation matrix $\mathbf{R}$ and the cross-correlation vector $\mathbf{p}$ are known so that we may compute the gradient vector $\nabla J(n)$ for a given value of the tap-weight vector $\mathbf{w}(n)$. Thus, substituting Eq. (8.8) in (8.7), we may compute the updated value of the tap-weight vector $\mathbf{w}(n + 1)$ by using the simple recursive relation

$$\mathbf{w}(n + 1) = \mathbf{w}(n) + \mu[\mathbf{p} - \mathbf{R}\mathbf{w}(n)], \quad n = 0, 1, 2, \ldots \tag{8.9}$$

We observe that the parameter μ controls the size of the incremental correction applied to the tap-weight vector as we proceed from one iteration cycle to the next. We therefore refer

to μ as the *step-size parameter*. Equation (8.9) describes the mathematical formulation of the steepest-descent algorithm.

According to Eq. (8.9), the correction $\delta\mathbf{w}(n)$ applied to the tap-weight vector at time $n + 1$ is equal to $\mu[\mathbf{p} - \mathbf{R}\mathbf{w}(n)]$. This correction may also be expressed as μ times the expectation of the inner product of the tap-input vector $\mathbf{u}(n)$ and the estimation error $e(n)$; see Problem 4. This suggests that we may use a bank of cross-correlators to compute the correction $\delta\mathbf{w}(n)$ applied to the tap-weight vector $\mathbf{w}(n)$ as indicated in Fig. 8.2. In this figure, the elements of the correction vector $\delta\mathbf{w}(n)$ are denoted by $\delta w_0(n)$, $\delta w_1(n)$, . . . , $\delta w_{M-1}(n)$.

Another point of interest is that we may view the steepest-descent algorithm of Eq. (8.9) as a *feedback model*, as illustrated by the *signal-flow graph* shown in Fig. 8.3. This model is multidimensional in the sense that the "signals" at the *nodes* of the graph consist of vectors and that the *transmittance* of each branch of the graph is a scalar or a square matrix. For each branch of the graph, the signal vector flowing out equals the signal vector flowing in multiplied by the transmittance matrix of the branch. For two branches connected in parallel, the overall transmittance matrix equals the sum of the transmittance matrices of the individual branches. For two branches connected in cascade, the overall transmittance matrix equals the product of the individual transmittance matrices arranged in the same order as the pertinent branches. Finally, the symbol z^{-1} is the unit-delay operator, and $z^{-1}\mathbf{I}$ is the transmittance matrix of a unit-delay branch representing a delay of one iteration cycle.

8.3 STABILITY OF THE STEEPEST-DESCENT ALGORITHM

Since the steepest-descent algorithm involves the presence of *feedback*, as exemplified by the model of Fig. 8.3, the algorithm is subject to the possibility of becoming *unstable*. From the feedback model of Fig. 8.3, we observe that the *stability performance* of the steepest-descent algorithm is determined by two factors: (1) the step-size parameter μ, and (2) the correlation matrix $\mathbf{R}$ of the tap-input vector $\mathbf{u}(n)$, as these two parameters completely control the transfer function of the *feedback loop*. To determine *the condition for the stability* of the steepest-descent algorithm, we examine the *natural modes* of the algorithm (Widrow, 1970). In particular, we use the representation of the correlation matrix $\mathbf{R}$ in terms of its eigenvalues and eigenvectors to define a transformed version of the tap-weight vector.

We begin the analysis by defining a *weight-error vector* at time n as

$$\mathbf{c}(n) = \mathbf{w}(n) - \mathbf{w}_o \tag{8.10}$$

where $\mathbf{w}_o$ is the optimum value of the tap-weight vector, as defined by the Wiener–Hopf equations (8.5). Then, eliminating the cross-correlation vector $\mathbf{p}$ between Eqs. (8.5) and (8.9), and rewriting the result in terms of the weight-error vector $\mathbf{c}(n)$, we get

$$\mathbf{c}(n + 1) = (\mathbf{I} - \mu\mathbf{R})\mathbf{c}(n) \tag{8.11}$$

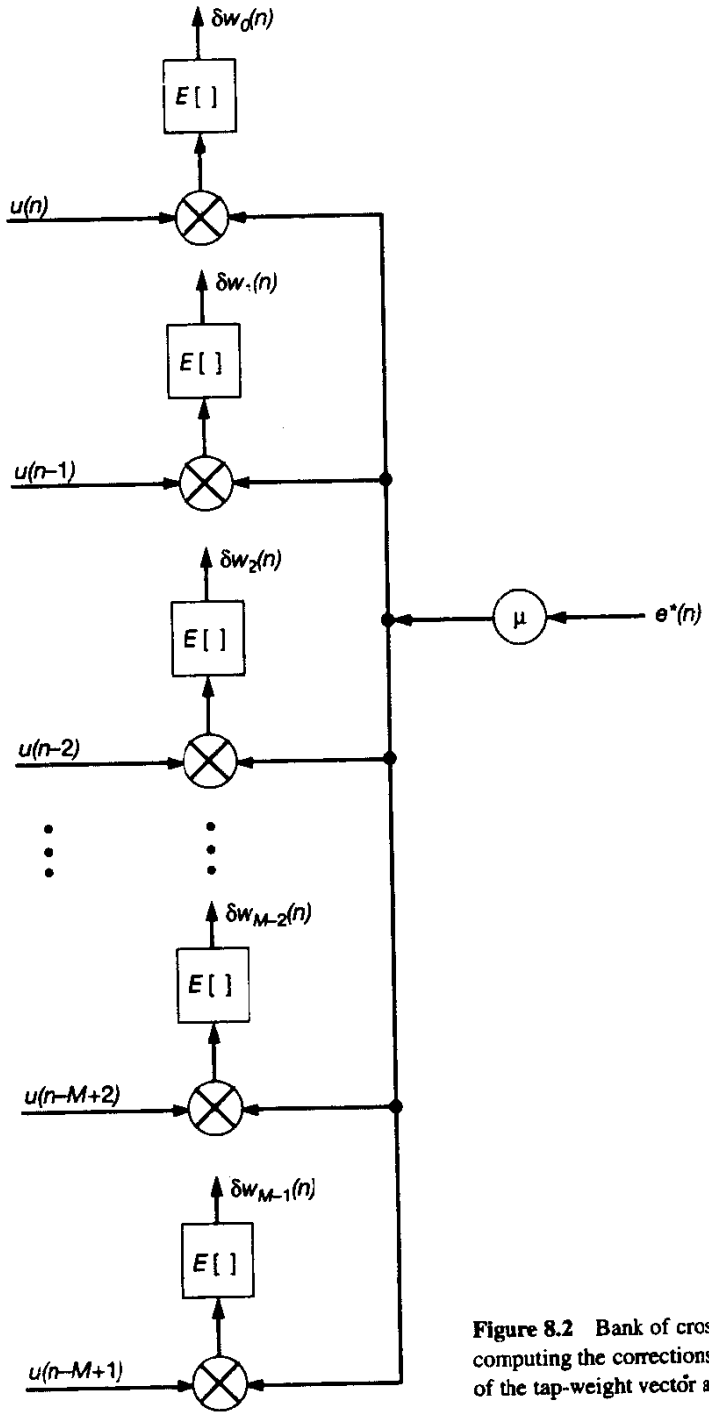


Figure 8.2 Bank of cross-correlators for computing the corrections to the elements of the tap-weight vector at time $n + 1$.

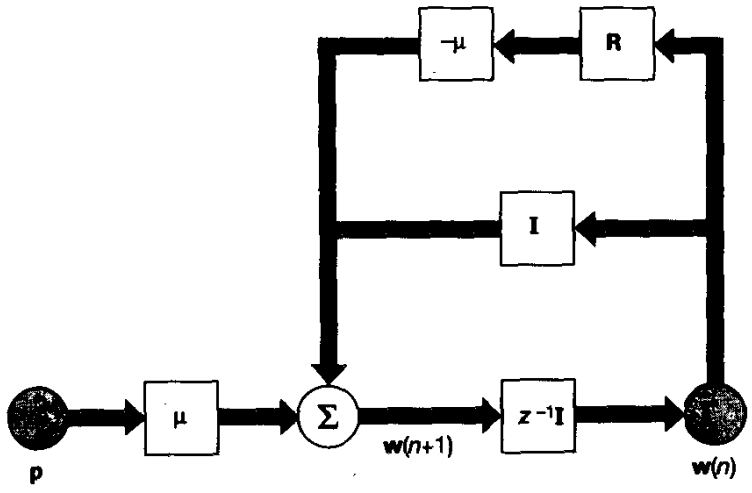


Figure 8.3 Signal-flow-graph representation of the steepest-descent algorithm.

where **I** is the identity matrix. Equation (8.11) is represented by the feedback model shown in Fig. 8.4. This diagram further emphasizes the fact that the stability performance of the steepest-descent algorithm is dependent exclusively on μ and **R**.

Using the unitary similarity transformation, we may express the correlation matrix **R** as follows (see Chapter 4):

$$\mathbf{R} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^H \tag{8.12}$$

The matrix **Q** has as its columns an orthogonal set of *eigenvectors* associated with the eigenvalues of the matrix **R**. The matrix **Q** is called the *unitary matrix* of the transformation. The matrix $\mathbf{\Lambda}$ is a *diagonal* matrix and has as its diagonal elements the eigenvalues of the correlation matrix **R**. These eigenvalues, denoted by $\lambda_1, \lambda_2, \ldots, \lambda_M$, are all positive and real. Each eigenvalue is associated with a corresponding eigenvector or column of matrix **Q**. Substituting Eq. (8.12) in (8.11), we get

$$\mathbf{c}(n+1) = (\mathbf{I} - \mu\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^H)\mathbf{c}(n) \tag{8.13}$$

Premultiplying both sides of this equation by $\mathbf{Q}^H$ and using the property of the unitary matrix **Q** that $\mathbf{Q}^H$ equals the inverse $\mathbf{Q}^{-1}$ (see Chapter 4), we get

$$\mathbf{Q}^H\mathbf{c}(n+1) = (\mathbf{I} - \mu\mathbf{\Lambda})\mathbf{Q}^H\mathbf{c}(n) \tag{8.14}$$

We now define a new set of coordinates as follows:

$$\begin{aligned}\boldsymbol{\nu}(n) &= \mathbf{Q}^H\mathbf{c}(n) \\ &= \mathbf{Q}^H[\mathbf{w}(n) - \mathbf{w}_o]\end{aligned} \tag{8.15}$$

Accordingly, we may rewrite Eq. (8.13) in the transformed form:

$$\boldsymbol{\nu}(n+1) = (\mathbf{I} - \mu\mathbf{\Lambda})\boldsymbol{\nu}(n) \tag{8.16}$$

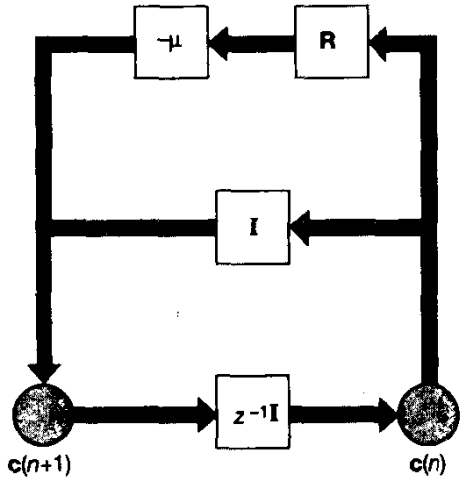


Figure 8.4 Signal-flow-graph representation of the steepest-descent algorithm based on the weight-error vector.

The initial value of $\boldsymbol{\nu}(n)$ equals

$$\boldsymbol{\nu}(0) = \mathbf{Q}^H[\mathbf{w}(0) - \mathbf{w}_o] \tag{8.17}$$

Assuming that the initial tap-weight vector is zero, Eq. (8.17) reduces to

$$\boldsymbol{\nu}(0) = -\mathbf{Q}^H\mathbf{w}_o \tag{8.18}$$

For the kth *natural mode* of the steepest descent algorithm, we thus have

$$\nu_k(n + 1) = (1 - \mu\lambda_k)\nu_k(n), \quad k = 1, 2, \ldots, M \tag{8.19}$$

where λ_k is the kth eignevalue of the correlation matrix $\mathbf{R}$. This equation is represented by the scalar-valued feedback model of Fig. 8.5, where z^{-1} is the unit-delay operator. Clearly, the structure of this model is much simpler than that of the original matrix-valued feedback model of Fig. 8.3. These two models represent different and yet equivalent ways of viewing the steepest-descent algorithm.

Equation (8.19) is a homogeneous *difference equation of the first order*. Assuming that $\nu_k(n)$ has the initial value $\nu_k(0)$, we readily obtain the solution

$$\nu_k(n) = (1 - \mu\lambda_k)^n\nu_k(0), \quad k = 1, 2, \ldots, M \tag{8.20}$$

Since all eignevalues of the correlation matrix $\mathbf{R}$ are positive and real, the response $\nu_k(n)$ will exhibit no oscillations. Furthermore, as illustrated in Fig. 8.6, the numbers generated by Eq. (8.20) represent a *geometric series* with a geometric ratio equal to $1 - \mu\lambda_k$. For *stability* or *convergence* of the steepest-descent algorithm, the magnitude of this geometric ratio must be less than 1 for all k. That is, provided we have

$$-1 < 1 - \mu\lambda_k < 1 \quad \text{for all } k$$

then as the number of iterations, n, approaches infinity, all the natural modes of the steepest-descent algorithm die out, irrespective of the initial conditions. This is equivalent to saying that the tap-weight vector $\mathbf{w}(n)$ approaches the optimum solution $\mathbf{w}_o$ as n approaches infinity.

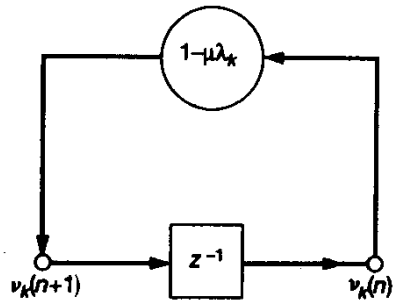


Figure 8.5 Signal-flow-graph representation of the kth natural mode of the steepest-descent algorithm.

Since the eigenvalues of the correlation matrix $\mathbf{R}$ are all real and positive, it therefore follows that the necessary and sufficient condition for the convergence or stability of the steepest-descent algorithm is to require the step-size parameter μ satisfy the following condition:

$$0 < \mu < \frac{2}{\lambda_{max}} \tag{8.21}$$

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

Referring to Fig. 8.6, we see that an exponential envelope of *time constant* τ_k can be fitted to the geometric series by assuming the unit of time to be the duration of one iteration cycle and by choosing the time constant τ_k such that

$$1 - \mu\lambda_k = \exp\left(-\frac{1}{\tau_k}\right)$$

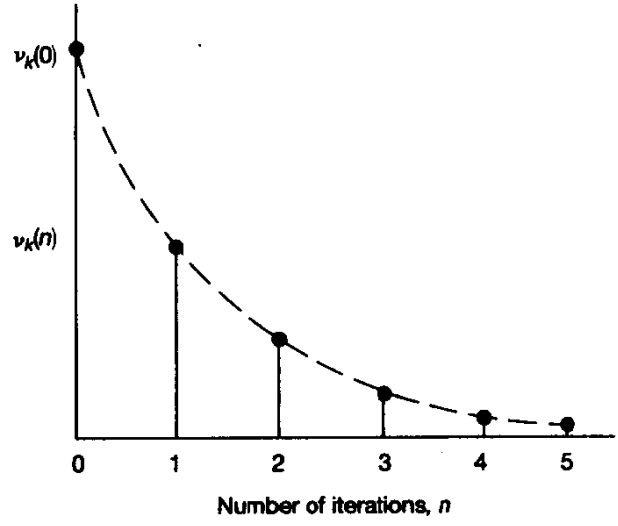


Figure 8.6 Variation of the kth natural mode of the steepest-descent algorithm with time, assuming that the magnitude of $1 - \mu\lambda_k$ is less than 1.

Hence, the kth time constant τ_k can be expressed in terms of the step-size parameter μ and the kth eigenvalue as follows:

$$\tau_k = \frac{-1}{\ln(1 - \mu\lambda_k)} \tag{8.22}$$

The time constant τ_k defines the number of iterations required for the amplitude of the kth natural mode $v_k(n)$ to decay to $1/e$ of its initial value $v_k(0)$, where e is the base of the natural logarithm. For the special case of slow adaptation, for which the step-size parameter μ is small, we may approximate the time constant τ_k as

$$\tau_k \simeq \frac{1}{\mu\lambda_k}, \qquad \mu << 1 \tag{8.23}$$

We may now formulate the transient behavior of the original tap-weight vector $\mathbf{w}(n)$. In particular, premultiplying both sides of Eq. (8.15) by $\mathbf{Q}$, using the fact that $\mathbf{QQ}^H = \mathbf{I}$, and solving for $\mathbf{w}(n)$, we get

$$\begin{aligned} \mathbf{w}(n) &= \mathbf{w}_o + \mathbf{Q}\mathbf{v}(n) \\ &= \mathbf{w}_o + [\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_M] \begin{bmatrix} v_1(n) \\ v_2(n) \\ \cdot \\ \cdot \\ \cdot \\ v_M(n) \end{bmatrix} \\ &= \mathbf{w}_o + \sum_{k=1}^{M} \mathbf{q}_k v_k(n) \end{aligned} \tag{8.24}$$

where $\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_M$ are the eigenvectors associated with the eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_M$ of the correlation matrix $\mathbf{R}$, respectively, and the kth natural mode $v_k(n)$ is defined by Eq. (8.20). Thus, substituting Eq. (8.20) in (8.24), we find that the transient behavior of the ith tap weight is described by (Griffiths, 1975)

$$w_i(n) = w_{oi} + \sum_{k=1}^{N} q_{ki} v_k(0)(1 - \mu\lambda_k)^n, \qquad i = 1, 2, \ldots, M \tag{8.25}$$

where w_{oi} is the optimum value of the ith tap weight, and q_{ki} is the ith element of the kth eigenvector $\mathbf{q}_k$.

Equation (8.25) shows that each tap weight in the steepest-descent algorithm converges as the weighted sum of exponentials of the form $(1 - \mu\lambda_k)^n$. The time τ_k required for each term to reach $1/e$ of its initial value is given by Eq. (8.22). However, the *overall time constant*, τ_a, defined as the time required for the summation term in Eq. (8.25) to decay to $1/e$ of its initial value, cannot be expressed in a simple closed form similar to Eq. (8.22). Nevertheless, the *slowest rate of convergence* is attained when $q_{ki}v_k(0)$ is zero for all k except for that mode corresponding to the smallest eigenvalue $\lambda_{\min}$ of matrix $\mathbf{R}$, so the upper bound on τ_a is defined by $-1/\ln(1 - \mu\lambda_{\min})$. The *fastest rate of convergence* is attained when all the $q_{ki}v_k(0)$ are zero except for that mode corresponding to the largest

eigenvalue λ_{max}, and so the lower bound on τ_a is defined by $-1/\ln(1 - \mu\lambda_{max})$. Accordingly, the overall time constant τ_a for any tap weight of the steepest-descent algorithm is bounded as follows (Griffiths, 1975):

$$\frac{-1}{\ln(1 - \mu\lambda_{max})} \leq \tau_a \leq \frac{-1}{\ln(1 - \mu\lambda_{min})} \tag{8.26}$$

We see therefore that, when the eigenvalues of the correlation matrix $\mathbf{R}$ are widely spread (i.e., the correlation matrix of the tap inputs is ill conditioned), the settling time of the steepest-descent algorithm is limited by the smallest eigenvalues or the slowest modes.

Transient Behavior of the Mean-Squared Error

We may develop further insight into the operation of the steepest-descent algorithm by examining the transient behavior of the mean-squared error $J(n)$. From Eq. (5.56) we have

$$J(n) = J_{min} + \sum_{k=1}^{M} \lambda_k |v_k(n)|^2 \tag{8.27}$$

where J_{min} is the minimum mean-squared error. The transient behavior of the kth natural mode, $v_k(n)$, is defined by Eq. (8.20). Hence substituting Eq. (8.20) into (8.27), we get

$$J(n) = J_{min} + \sum_{k=1}^{M} \lambda_k (1 - \mu\lambda_k)^{2n} |v_k(0)|^2 \tag{8.28}$$

where $v_k(0)$ is the initial value of $v_k(n)$. When the steepest-descent algorithm is convergent, that is, the step-size parameter μ is chosen within the bounds defined by Eq. (8.21), we see that, irrespective of the initial conditions,

$$\lim_{n \to \infty} J(n) = J_{min} \tag{8.29}$$

The curve obtained by plotting the mean-squared error $J(n)$ versus the number of iterations, n, is called a *learning curve*. From Eq. (8.28), we see that *the learning curve of the steepest-descent algorithm consists of a sum of exponentials, each of which corresponds to a natural mode of the algorithm.* In general, the number of natural modes equals the number of tap weights. In going from the initial value $J(0)$ to the final value J_{min}, the exponential decay for the kth natural mode has a time constant equal to

$$\tau_{k,mse} \simeq \frac{-1}{2\ln(1 - \mu\lambda_k)} \tag{8.30}$$

For small values of the step-size parameter μ, we may approximate this time constant as

$$\tau_{k,mse} \simeq \frac{1}{2\mu\lambda_k} \tag{8.31}$$

Equation (8.31) shows that the smaller the step-size parameter μ, the slower will be the rate of decay of each natural mode of the LMS algorithm.

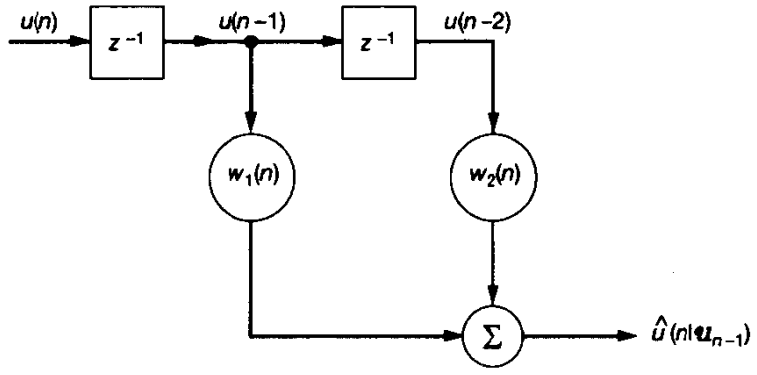


Figure 8.7 Two-tap predictor for real-valued input.

8.4 EXAMPLE

In this example, we examine the transient behavior of the steepest-descent algorithm applied to a predictor that operates on a real-valued autoregressive (AR) process. Figure 8.7 shows the structure of the predictor, assumed to contain two tap weights that are denoted by $w_1(n)$ and $w_2(n)$; the dependence of these tap weights on the number of iterations n emphasizes the transient condition of the predictor. The AR process $u(n)$ is described by the second-order difference equation

$$u(n) + a_1 u(n-1) + a_2 u(n-2) = v(n) \tag{8.32}$$

where the sample $v(n)$ is drawn from a white-noise process of zero mean and variance σ_v^2. The AR parameters a_1 and a_2 are chosen so that the roots of the characteristic equation

$$1 + a_1 z^{-1} + a_2 z^{-2} = 0$$

are complex; that is, $a_1^2 < 4a_2$. The particular values assigned to a_1 and a_2 are determined by the desired eigenvalue spread $\chi(\mathbf{R})$. For specified values of a_1 and a_2, the variance σ_v^2 of the white-noise process is chosen to make the process $u(n)$ have variance $\sigma_u^2 = 1$.

The requirement is to evaluate the transient behavior of the steepest-descent algorithm for the following conditions:

- Varying eigenvalue spread $\chi(\mathbf{R})$ and fixed step-size parameter μ.
- Varying step-size parameter μ and fixed eigenvalue spread $\chi(\mathbf{R})$.

Characterization of the AR Process

Since the predictor of Fig. 8.7 has two tap weights and the AR process $u(n)$ is real valued, it follows that the correlation matrix $\mathbf{R}$ of the tap inputs is a 2-by-2 symmetric matrix:

$$\mathbf{R} = \begin{bmatrix} r(0) & r(1) \\ r(1) & r(0) \end{bmatrix}$$

where (see Chapter 2)

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

$$r(1) = -\frac{a_1}{1 + a_2}\sigma_u^2$$

$$\sigma_u^2 = \left(\frac{1 + a_2}{1 - a_2}\right)\frac{\sigma_v^2}{(1 + a_2)^2 - a_1^2}$$

The two eigenvalues of $\mathbf{R}$ are

$$\lambda_1 = \left(1 - \frac{a_1}{1 + a_2}\right)\sigma_u^2$$

$$\lambda_2 = \left(1 + \frac{a_1}{1 + a_2}\right)\sigma_u^2$$

Hence, the eigenvalue spread equals (assuming that a_1 is negative)

$$\chi(\mathbf{R}) = \frac{\lambda_1}{\lambda_2} = \frac{1 - a_1 + a_2}{1 + a_1 + a_2}$$

The eigenvectors $\mathbf{q}_1$ and $\mathbf{q}_2$ associated with the respective eigenvalues λ_1 and λ_2 are

$$\mathbf{q}_1 = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$\mathbf{q}_2 = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

both of which are normalized to unit length.

Experiment 1: Varying Eigenvalue Spread

In this experiment, the step-size parameter μ is fixed at 0.3, and the evaluations are made for the four sets of AR parameters given in Table 8.1.

TABLE 8.1 SUMMARY OF PARAMETER VALUES CHARACTERIZING THE SECOND-ORDER AR MODELING PROBLEM

	AR parameters		Eigenvalues		Eigenvalue spread,	Minimum mean squared error,
Case	a_1	a_2	λ_1	λ_2	$\chi = \lambda_1/\lambda_2$	$J_{\min} = \sigma_v^2$
1	−0.1950	0.95	1.1	0.9	1.22	0.0965
2	−0.9750	0.95	1.5	0.5	3	0.0731
3	−1.5955	0.95	1.818	0.182	10	0.0322
4	−1.9114	0.95	1.957	0.0198	100	0.0038

For a given set of parameters, we use a two-dimensional plot of the transformed tap-weight error $v_1(n)$ versus $v_2(n)$ to display the transient behavior of the steepest-descent algorithm. In particular, the use of Eq. (8.20) yields

$$\begin{aligned} \mathbf{v}(n) &= \begin{bmatrix} v_1(n) \\ v_2(n) \end{bmatrix} \\ &= \begin{bmatrix} (1 - \mu\lambda_1)^n v_1(0) \\ (1 - \mu\lambda_2)^n v_2(0) \end{bmatrix}, \qquad n = 1, 2, \ldots \end{aligned} \tag{8.33}$$

To calculate the initial value $\mathbf{v}(0)$, we use Eq. (8.18), assuming that the initial value $\mathbf{w}(0)$ of the tap-weight vector $\mathbf{w}(n)$ is zero. This equation requires knowledge of the optimum tap-weight vector $\mathbf{w}_o$. Now when the two-tap predictor of Fig. 8.7 is optimized, with the second-order AR process of Eq. (8.32) supplying the tap inputs, we find that the optimum tap-weight vector equals

$$\mathbf{w}_o = \begin{bmatrix} -a_1 \\ -a_2 \end{bmatrix}$$

and the minimum mean-squared error equals

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

Accordingly, the use of Eq. (8.18) yields the initial value:

$$\begin{aligned} \mathbf{v}(0) &= \begin{bmatrix} v_1(0) \\ v_2(0) \end{bmatrix} \\ &= \frac{-1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} -a_1 \\ -a_2 \end{bmatrix} \\ &= \frac{1}{\sqrt{2}} \begin{bmatrix} a_1 + a_2 \\ a_1 - a_2 \end{bmatrix} \end{aligned} \tag{8.34}$$

Thus, for specified parameters, we use Eq. (8.34) to compute the initial value $\mathbf{v}(0)$, and then use Eq. (8.33) to compute $\mathbf{v}(1)$, $\mathbf{v}(2)$, ... By joining the points defined by these values of $\mathbf{v}(n)$ for varying n, we obtain a *trajectory* that describes the transient behavior of the steepest-descent algorithm for the particular set of parameters.

It is informative to include in the two-dimensional plot of $v_1(n)$ versus $v_2(n)$ loci representing Eq. (8.27) for fixed values of n. For our example, Eq. (8.27) yields

$$J(n) - J_{\min} = \lambda_1 v_1^2(n) + \lambda_2 v_2^2(n) \tag{8.35}$$

When $\lambda_1 = \lambda_2$ and n is fixed, Eq. (8.35) represents a circle with center at the origin and radius equal to the square root of $[J(n) - J_{\min}]/\lambda$, where λ is the common value of the two eigenvalues. When, on the other hand, $\lambda_1 \neq \lambda_2$, Eq. (8.35) represents (for fixed n) an ellipse with major axis equal to the square root of $[J(n) - J_{\min}]/\lambda_2$ and minor axis equal to the square root of $[J(n) - J_{\min}]/\lambda_1$.

Case 1: Eigenvalue Spread $\chi(\mathbf{R}) = 1.22$. For the parameter values given for Case 1 in Table 8.1, the eigenvalue spread $\chi(\mathbf{R})$ equals 1.22; that is, the eigenvalues λ_1 and λ_2 are approximately equal. The use of these parameter values in Eqs. (8.33) and (8.34) yields the trajectory of $[v_1(n), v_2(n)]$ shown in Fig. 8.8(a), with n as running parameter, and their use in Eq. (8.35) yields the (approximately) circular loci shown for fixed values of $J(n)$, corresponding to $n = 0, 1, 2, 3, 4, 5$.

We may also display the transient behavior of the steepest-descent algorithm by plotting the tap weight $w_1(n)$ versus $w_2(n)$. In particular, for our example the use of Eq. (8.24) yields the tap-weight vector

$$\begin{aligned}\mathbf{w}(n) &= \begin{bmatrix} w_1(n) \\ w_2(n) \end{bmatrix} \\ &= \begin{bmatrix} -a_1 + (v_1(n) + v_2(n))/\sqrt{2} \\ -a_2 + (v_1(n) - v_2(n))/\sqrt{2} \end{bmatrix}\end{aligned} \tag{8.36}$$

The corresponding trajectory of $[w_1(n), w_2(n)]$, with n as a running parameter, obtained by using Eq. (8.36), is shown plotted in Fig. 8.9(a). Here again we have included the loci of $[w_1(n), w_2(n)]$ for fixed values of $J(n)$ corresponding to $n = 0, 1, 2, 3, 4, 5$. Note that these loci, unlike Fig. 8.8(a), are ellipsoidal.

Case 2: Eigenvalue Spread $\chi(\mathbf{R}) = 3$. The use of the parameter values for Case 2 in Eqs. (8.33) and (8.34) yields the trajectory of $[v_1(n), v_2(n)]$ shown in Fig. 8.8(b), with n as running parameter, and their use in Eq. (8.35) yields the ellipsoidal loci shown for the fixed values of $J(n)$ for $n = 0, 1, 2, 3, 4, 5$. Note that for this set of parameter values the initial value $v_2(0)$ is approximately zero, so the initial value $\mathbf{v}(0)$ lies practically on the v_1-axis.

The corresponding trajectory of $[w_1(n), w_2(n)]$, with n as running parameter, is shown in Fig. 8.9(b).

Case 3: Eigenvalue Spread $\chi(\mathbf{R}) = 10$. For this case, the application of Eqs. (8.33) and (8.34) yields the trajectory of $[v_1(n), v_2(n)]$ shown in Fig. 8.8(c), with n as running parameter, and the application of Eq. (8.35) yields the ellipsoidal loci included in this figure for fixed values of $J(n)$ for $n = 0, 1, 2, 3, 4, 5$. The corresponding trajectory of $[w_1(n), w_2(n)]$, with n as running parameter, is shown in Fig. 8.9(c).

Case 4: Eigenvalue Spread $\chi(\mathbf{R}) = 100$. For this case, application of the preceding equations yields the results shown in Fig. 8.8(d) for the trajectory of $[v_1(n), v_2(n)]$ and the ellipsoidal loci for fixed values of $J(n)$. The corresponding trajectory of $[w_1(n), w_2(n)]$ is shown in Fig. 8.9(d).

In Fig. 8.10 we have plotted the mean-squared error $J(n)$ versus n for the four eigenvalue spreads 1.22, 3, 10, and 100. We see that as the eigenvalue spread increases (and the input process becomes more correlated), the minimum mean-squared error $J_{\min}$ decreases.

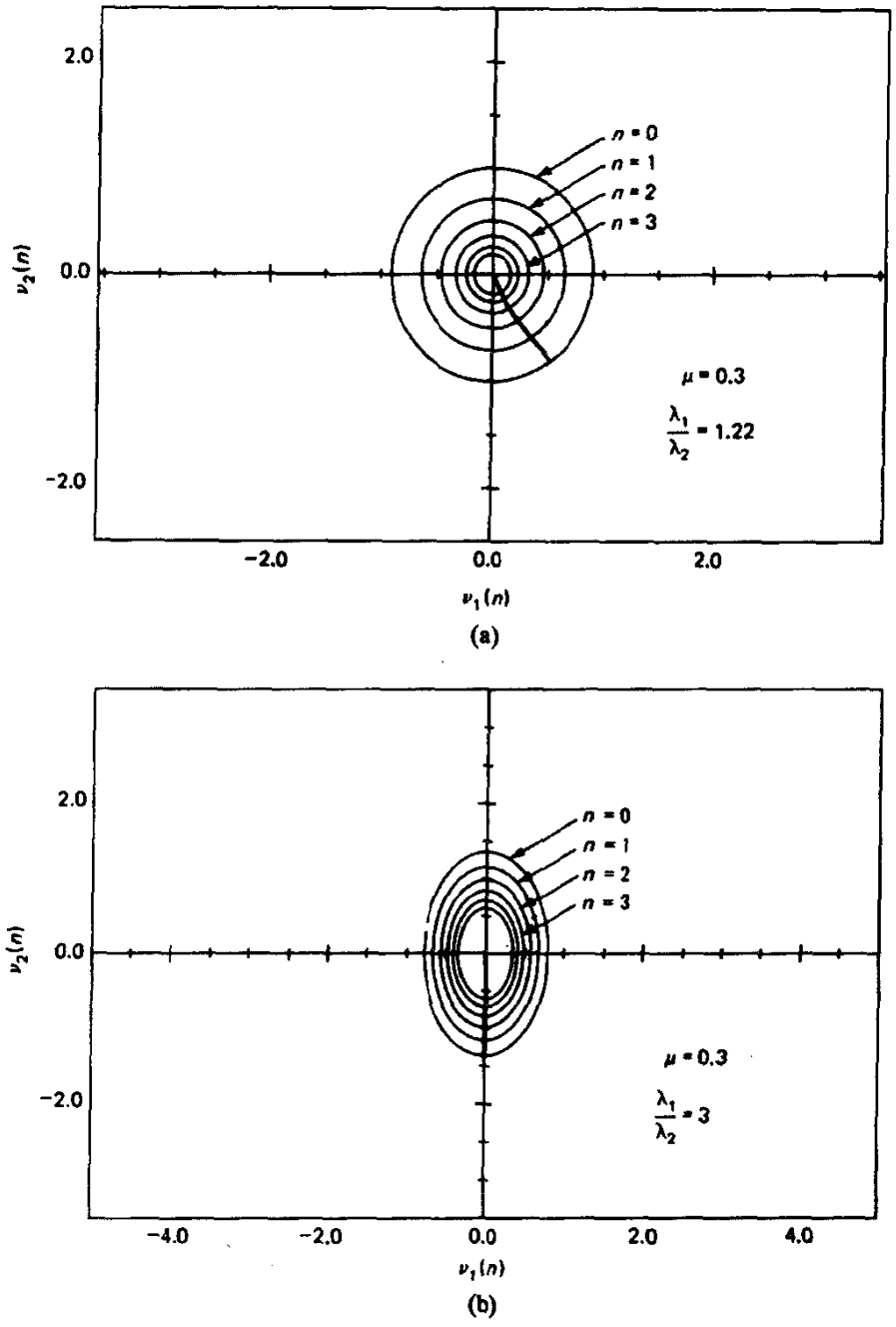
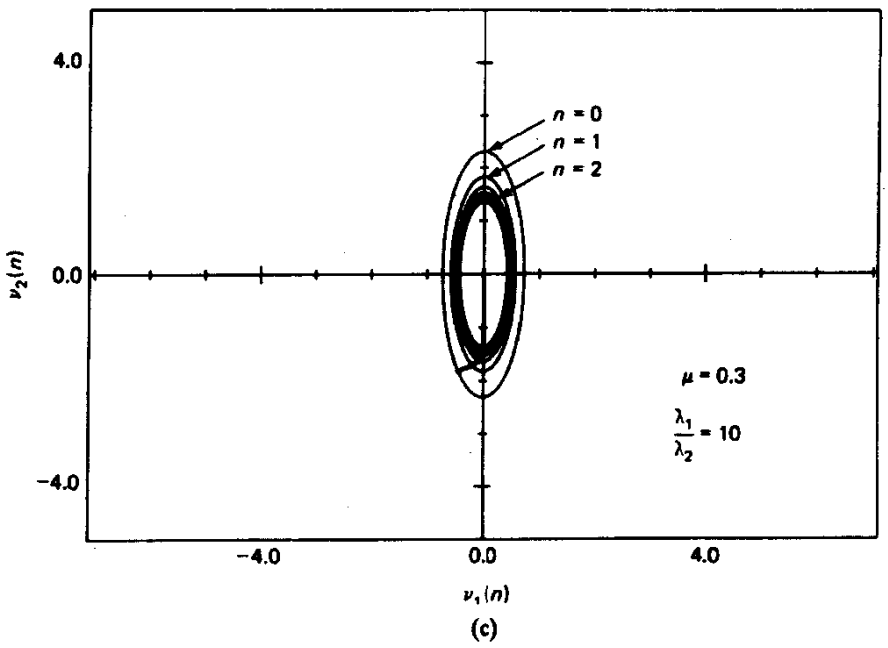
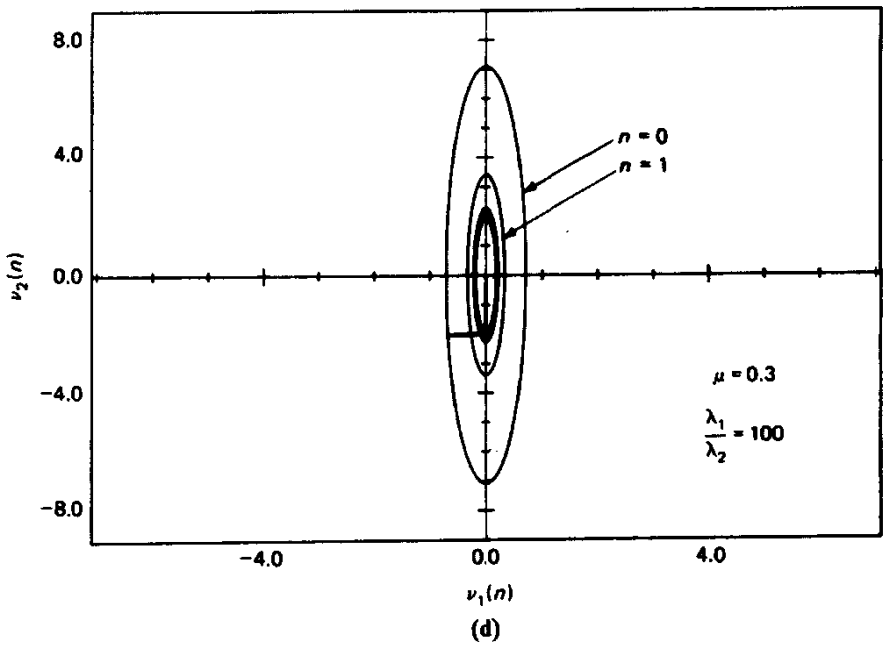


Figure 8.8 Loci of $\nu_1(n)$ versus $\nu_2(n)$ for the steepest-descent algorithm with step-size parameter $\mu = 0.3$ and varying eigenvalue spread: (a) $\chi(\mathbf{R}) = 1.22$; (b) $\chi(\mathbf{R}) = 3$; (c) $\chi(\mathbf{R}) = 10$; (d) $\chi(\mathbf{R}) = 100$.



(c)



(d)

Figure 8.8 (*Cont.*)

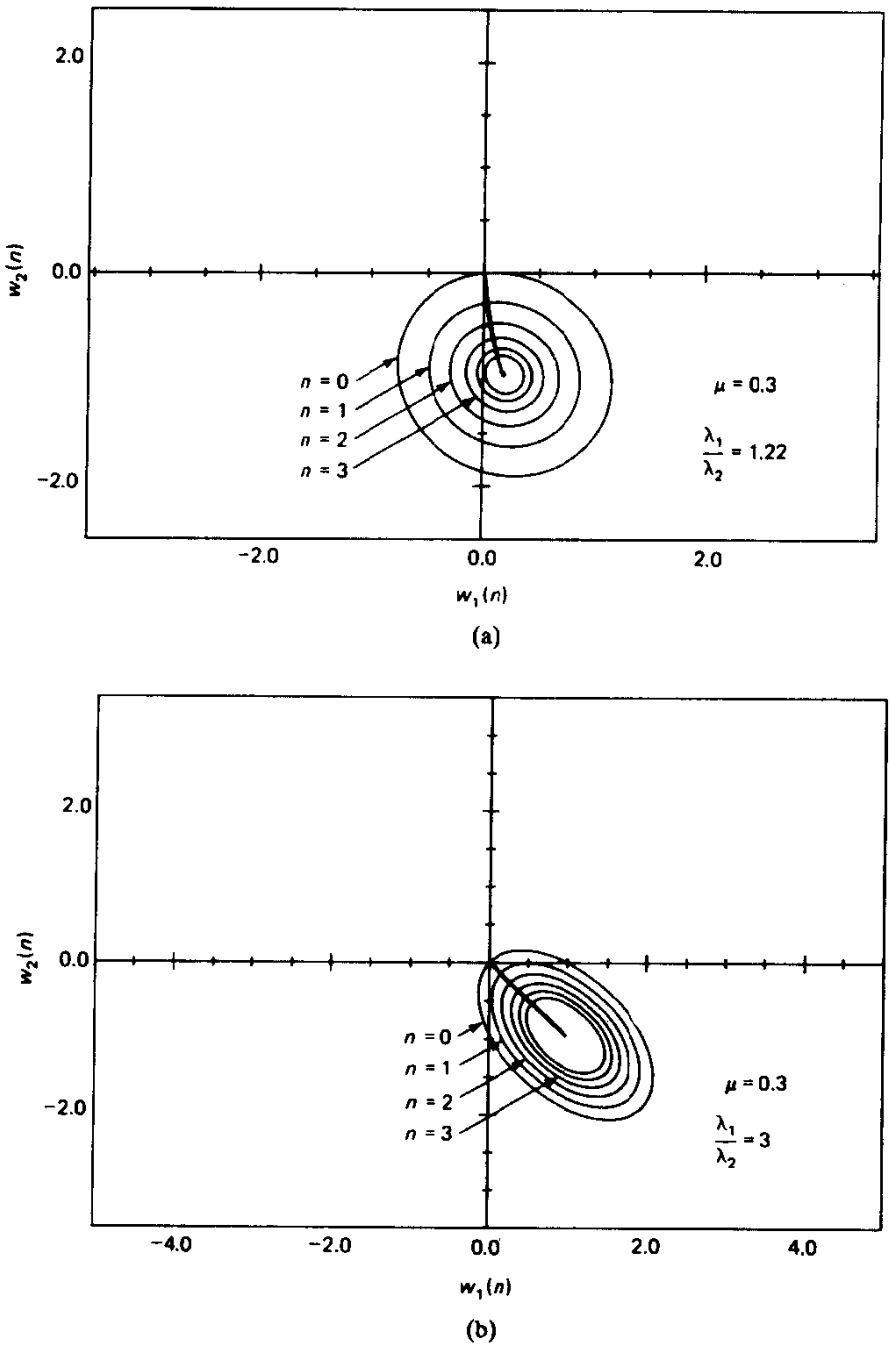
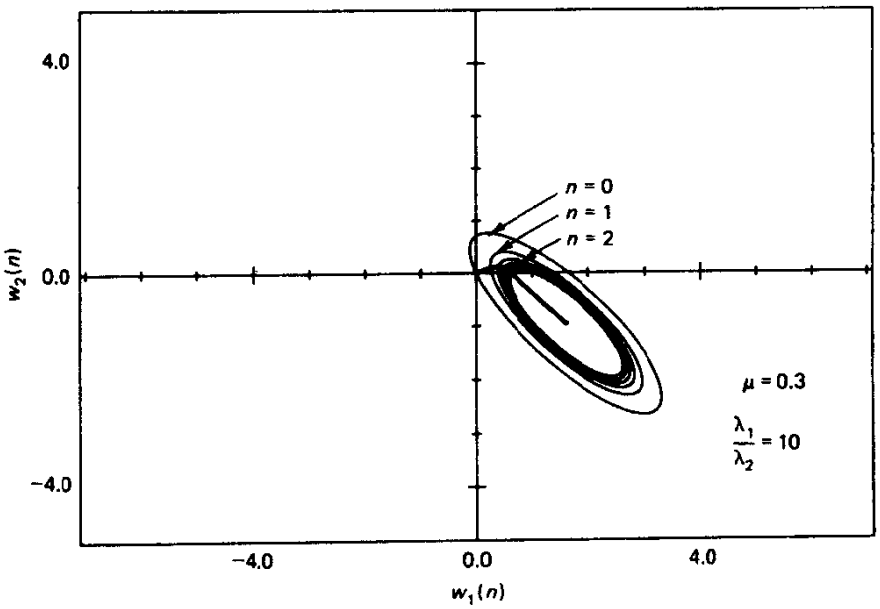
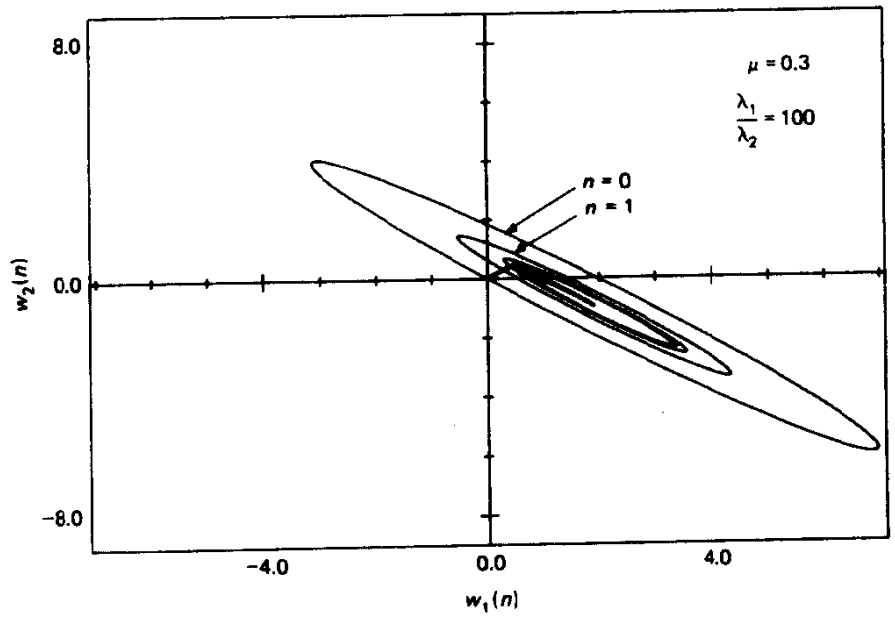


Figure 8.9 Loci of $w_1(n)$ versus $w_2(n)$ for the steepest-descent algorithm with step-size parameter $\mu = 0.3$ and varying eigenvalue spread: (a) $\chi(\mathbf{R}) = 1.22$; (b) $\chi(\mathbf{R}) = 3$; (c) $\chi(\mathbf{R}) = 10$; (d) $\chi(\mathbf{R}) = 100$.



(c)



(d)

Figure 8.9 *(Cont.)*

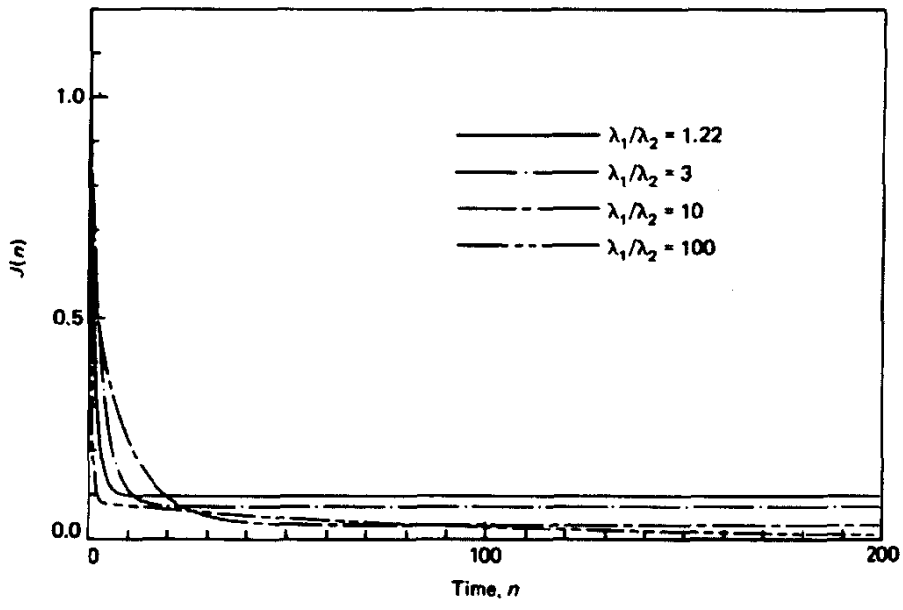


Figure 8.10 Learning curves of steepest-descent algorithm with step-size parameter $\mu = 0.3$ and varying eigenvalue spread.

This makes intuitive sense: The predictor should do a better job tracking a highly correlated input process than a weakly correlated one.

Experiment 2: Varying Step-size Parameter

In this experiment the eigenvalue spread is fixed at $\chi(\mathbf{R}) = 10$, and the step-size parameter μ is varied. In particular, we examine the transient behavior of the steepest-descent algorithm for $\mu = 0.3$ and 1.0. The corresponding results in terms of the transformed tap-weight errors $v_1(n)$ and $v_2(n)$ are shown in parts (a) and (b) of Fig. 8.11, respectively. The results included in part (a) of this figure are the same as those in Fig. 8.8(c). Note also that in accordance with Eq. (8.21), the critical value of the step-size parameter equals $\mu_{max} = 2/\lambda_{max} = 1.1$, which is slightly in excess of the actual value $\mu = 1$ used in Fig. 8.11(b).

The results for $\mu = 0.3$ and 1.0 in terms of the tap weights $w_1(n)$ and $w_2(n)$ are shown in parts (a) and (b) of Fig. 8.12, respectively. Here again, the results included in part (a) of the figure are the same as those in Fig. 8.9(c).

Observations

Based on the results presented for Experiments 1 and 2, we may make the following observations:

1. The trajectory of $[v_1(n), v_2(n)]$, with the number of iterations n as running parameter, is normal to the locus of $[v_1(n), v_2(n)]$ for fixed $J(n)$. This also applies to the trajectory of $[w_1(n), w_2(n)]$ for fixed $J(n)$.

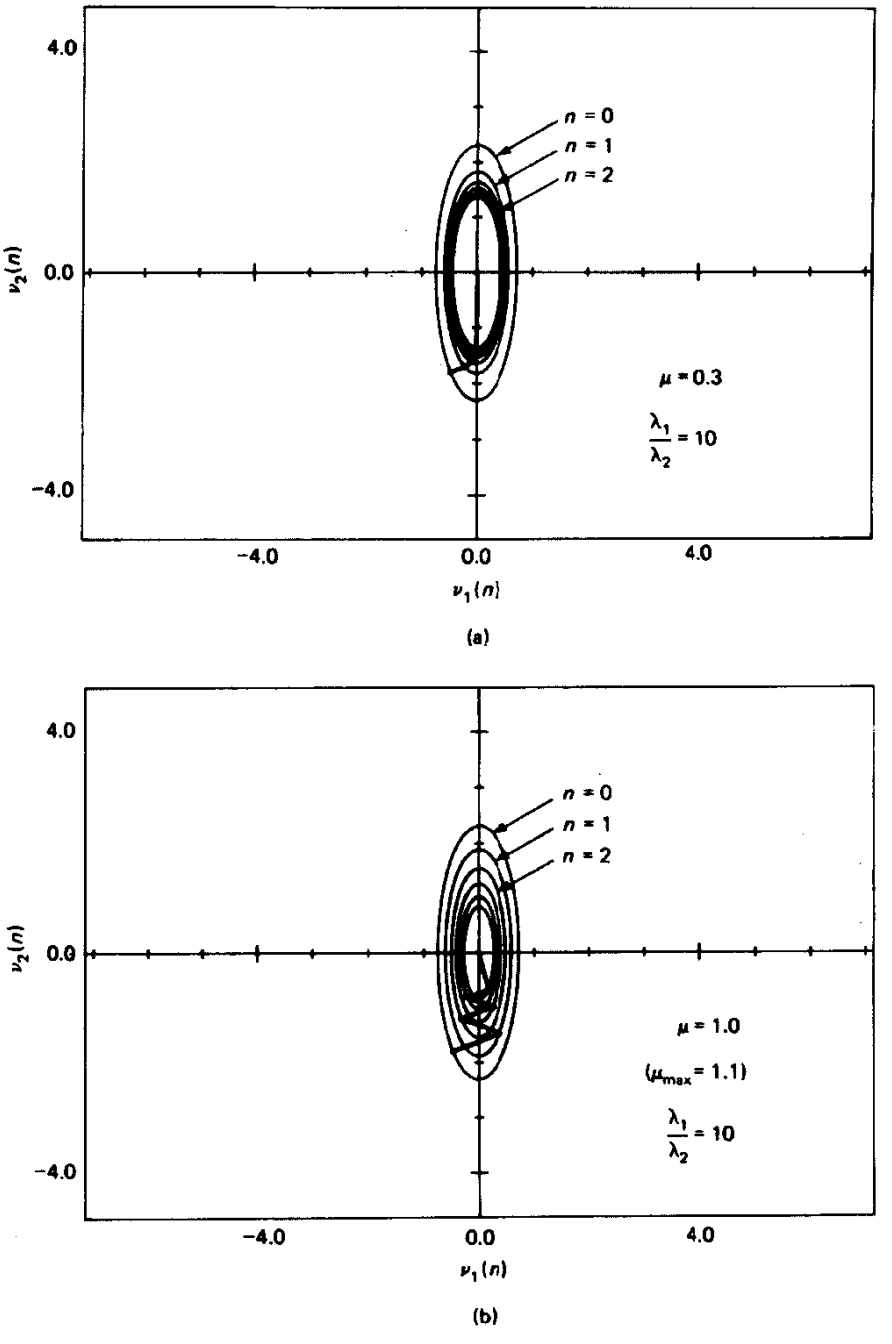


Figure 8.11 Loci of $\nu_1(n)$ versus $\nu_2(n)$ for the steepest-descent algorithm with eigenvalue spread $\chi(\mathbf{R}) = 10$ and varying step-size parameters: (a) overdamped, $\mu = 0.3$; (b) underdamped, $\mu = 1.0$.

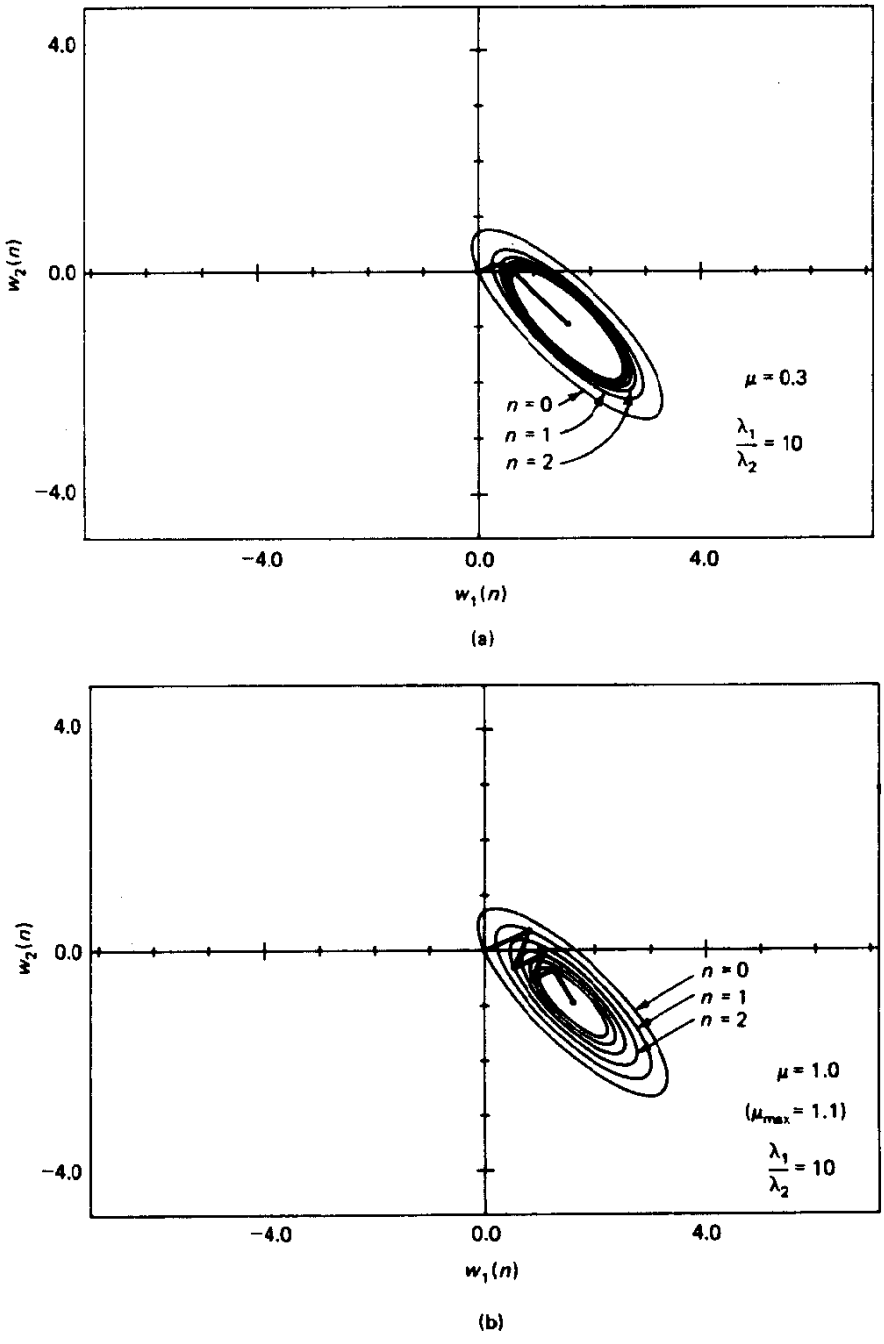


Figure 8.12 Loci of $w_1(n)$ versus $w_2(n)$ for the steepest-descent algorithm with eigenvalue spread $\chi(\mathbf{R}) = 10$ and varying step-size parameters: (a) overdamped, $\mu = 0.3$; underdamped, $\mu = 1.0$.

2. When the eigenvalues λ_1 and λ_2 are equal, the trajectory of $[v_1(n), v_2(n)]$ or that of $[w_1(n), w_2(n)]$, with n as running parameter, is a straight line. This is illustrated in Fig. 8.8(a) or 8.9(a) for which the eigenvalues λ_1 and λ_2 are approximately equal.
3. When the conditions are right for the initial value $\mathbf{v}(0)$ of the transformed tap-weight error vector $\mathbf{v}(n)$ to lie on the v_1-axis or v_2-axis, the trajectory of $[v_1(n), v_2(n)]$, with n as running parameter, is a straight line. This is illustrated in Fig. 8.8(b), where $v_1(0)$ is approximately zero. Correspondingly, the trajectory of $[w_1(n), w_2(n)]$, with n as running parameter, is also a straight line, as illustrated in Fig. 8.9(b).
4. Except for two special cases—(1) equal eigenvalues, and (2) the right choice of initial conditions—the trajectory of $[v_1(n), v_2(n)]$, with n as running parameter, follows a curved path, as illustrated in Fig. 8.8(c). Correspondingly, the trajectory of $[w_1(n), w_2(n)]$, with n as running parameter, also follows a curved path as illustrated in Fig. 8.9(c). When the eigenvalue spread is very high (i.e., the input data are very highly correlated), two things happen:
   - The error performance surface assumes the shape of a deep valley.
   - The trajectories of $[v_1(n), v_2(n)]$ and $[w_1(n), w_2(n)]$ develop distinct bends. Both of these points are well illustrated in Figs. 8.8(d) and 8.9(d), respectively, for the case of $\chi(\mathbf{R}) = 100$.
5. The steepest-descent algorithm converges fastest when the eigenvalues λ_1 and λ_2 are equal or the starting point of the algorithm is chosen right, for which cases the trajectory formed by joining the points $v(0)$, $v(1)$, $v(2)$, . . . is a straight line, the shortest possible path.
6. For fixed step-size parameter μ, as the eigenvalue spread $\chi(\mathbf{R})$ increases (i.e., the correlation matrix $\mathbf{R}$ of the tap inputs becomes more ill conditioned), the ellipsoidal loci of $[v_1(n), v_2(n)]$ for fixed values of $J(n)$, for $n = 0, 1, 2, \ldots$, become increasingly narrower (i.e., the minor axis becomes smaller) and more crowded.
7. When the step-size parameter μ is small, the transient behavior of the steepest-descent algorithm is *overdamped*, in that the trajectory formed by joining the points $\mathbf{v}(0)$, $\mathbf{v}(1)$, $\mathbf{v}(2)$, . . . follows a continuous path. When, on the other hand μ approaches the maximum allowable value, $\mu_{max} = 2/\lambda_{max}$, the transient behavior of the steepest-descent algorithm is *underdamped*, in that this trajectory exhibits oscillations. These two different forms of transient behavior are illustrated in parts (a) and (b) of Fig. 8.11 in terms of $v_1(n)$ and $v_2(n)$. The corresponding results in terms of $w_1(n)$ and $w_2(n)$ are presented in parts (a) and (b) of Fig. 8.12.

The conclusion to be drawn from these observations is that the transient behavior of the steepest-descent algorithm is highly sensitive to variations in both the step-size parameter μ and the eigenvalue spread of the correlation matrix of the tap inputs.

8.5 SUMMARY AND DISCUSSION

The method of steepest descent provides a simple procedure for computing the tap-weight vector of a Wiener filter, given knowledge of two ensemble-averaged quantities:

- The correlation matrix of the tap-input vector
- The cross-correlation vector between the tap-input vector and the desired response.

A critical feature of the method of steepest descent is the presence of feedback, which is another way of saying that the underlying algorithm is recursive in nature. As such, we have to pay particular attention to the issue of stability, which is governed by two parameters in the feedback loop of the algorithm:

- The step-size parameter μ
- The correlation matrix $\mathbf{R}$ of the tap-input vector.

Specifically, the necessary and sufficient condition for stability of the algorithm is embodied in the following:

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

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

Moreover, depending on the value assigned to the step-size parameter μ, the transient response of the steepest descent algorithm may exhibit one of three forms of behavior:

- *Underdamped response*, in which case the trajectory followed by the tap-weight vector toward the optimum Wiener solution exhibits oscillations; this response arises when the step-size parameter μ is large.
- *Overdamped response*, which is a nonoscillatory behavior that arises when μ is small.
- *Critically damped response*, which is the fine dividing line between the underdamped and overdamped conditions.

Unfortunately, in general, these conditions do not lend themselves to an exact mathematical analysis; they are usually evaluated by experimentation.

PROBLEMS

1. Consider a Wiener filtering problem characterized by the following values for the correlation matrix $\mathbf{R}$ of the tap-input vector $\mathbf{u}(n)$ and the cross-correlation vector $\mathbf{p}$ between $\mathbf{u}(n)$ and the desired response $d(n)$:

$$\mathbf{R} = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}$$

$$\mathbf{p} = \begin{bmatrix} 0.5 \\ 0.25 \end{bmatrix}$$

(a) Suggest a suitable value for the step-size parameter μ that would ensure convergence of the method of steepest descent, based on the given value for matrix $\mathbf{R}$.

(b) Using the value proposed in part (a), determine the recursions for computing the elements $w_1(n)$ and $w_2(n)$ of the tap-weight vector $\mathbf{w}(n)$. For this computation, you may assume the initial values

$$w_1(0) = w_2(0) = 0$$

(c) Investigate the effect of varying the step-size parameter μ on the trajectory of the tap-weight vector $\mathbf{w}(n)$ as n varies from zero to infinity.

2. Start with the formula for the estimation error:

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

where $d(n)$ is the desired response, $\mathbf{u}(n)$ is the tap-input vector, and $\mathbf{w}(n)$ is the tap-weight vector in the transversal filter. Hence, show that the gradient of the instantaneous squared error $|e(n)|^2$ equals

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

3. In this problem we explore another way of deriving the steepest-descent algorithm of Eq. (8.9) used to adjust the tap-weight vector in a transversal filter. The inverse of a positive-definite matrix may be expanded in a series as follows:

$$\mathbf{R}^{-1} = \mu \sum_{k=0}^{\infty} (\mathbf{I} - \mu\mathbf{R})^k$$

where $\mathbf{I}$ is the identity matrix, and μ is a positive constant. To ensure convergence of the series, the constant μ must lie inside the range

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

where λ_{max} is the largest eigenvalue of the matrix $\mathbf{R}$. By using this series expansion for the inverse of the correlation matrix in the Wiener–Hopf equations, develop the recursion

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu[\mathbf{p} - \mathbf{R}\mathbf{w}(n)]$$

where $\mathbf{w}(n)$ is the approximation to the Wiener solution for the tap-weight vector:

$$\mathbf{w}(n) = \mu \sum_{k=0}^{n-1} (\mathbf{I} - \mu\mathbf{R})^k \mathbf{p}$$

4. In the method of steepest descent, show that the correction applied to the tap-weight vector after $n+1$ iterations may be expressed as follows:

$$\delta\mathbf{w}(n+1) = \mu E[\mathbf{u}(n)e^*(n)]$$

where $\mathbf{u}(n)$ is the tap-input vector and $e(n)$ is the estimation error. What happens to this correction at the minimum point of the error performance surface? Discuss your answer in light of the principle of orthogonality.

5. Consider the method of steepest descent involving a single weight $w(n)$. Do the following:
   (a) Determine the mean-squared error $J(n)$ as a function of $w(n)$.
   (b) Find the Wiener solution w_o, and the minimum mean-squared error $J_{\min}$.
   (c) Sketch a plot of $J(n)$ versus $w(n)$.

6. Equation (8.28) defines the transient behavior of the mean-squared error $J(n)$ for varying n that is produced by the steepest-descent algorithm. Let $J(0)$ and $J(\infty)$ denote the initial and final values of $J(n)$. Suppose that we approximate this transient behavior with a single exponential, as follows:

$$J_{\text{approx}}(n) = [J(0) - J(\infty)]e^{-n/\tau} + J(\infty)$$

where τ is termed the *effective time constant*. Let τ be chosen such that

$$J_{\text{approx}}(1) = J(1)$$

Hence, show that *the initial rate of convergence* of the steepest-descent algorithm, defined as the inverse of τ, is given by

$$\frac{1}{\tau} = \ln\left[\frac{J(0) - J(\infty)}{J(1) - J(\infty)}\right]$$

Using Eq. (8.28), find the value of $1/\tau$. Assume that the initial value $\mathbf{w}(0)$ is zero, and that the step-size parameter μ is small.

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

$$u(n) = -au(n-1) + v(n)$$

where a is the AR parameter of the process, and $v(n)$ is a zero-mean white-noise process of variance σ_v^2.
   (a) Set up a linear predictor of order 1 to compute the parameter a. To be specific, use the method of steepest descent for the recursive computation of the Wiener solution for the parameter a.
   (b) Plot the error-performance curve for this problem, identifying the minimum point of the curve in terms of known parameters.
   (c) What is the condition on the step-size parameter μ to ensure stability? Justify your answer.