LINEAR FIR ADAPTIVE FILTERING (I)
Gradient-based Algorithms (Method of Steepest Descent)

Dr. Yogananda Isukapalli

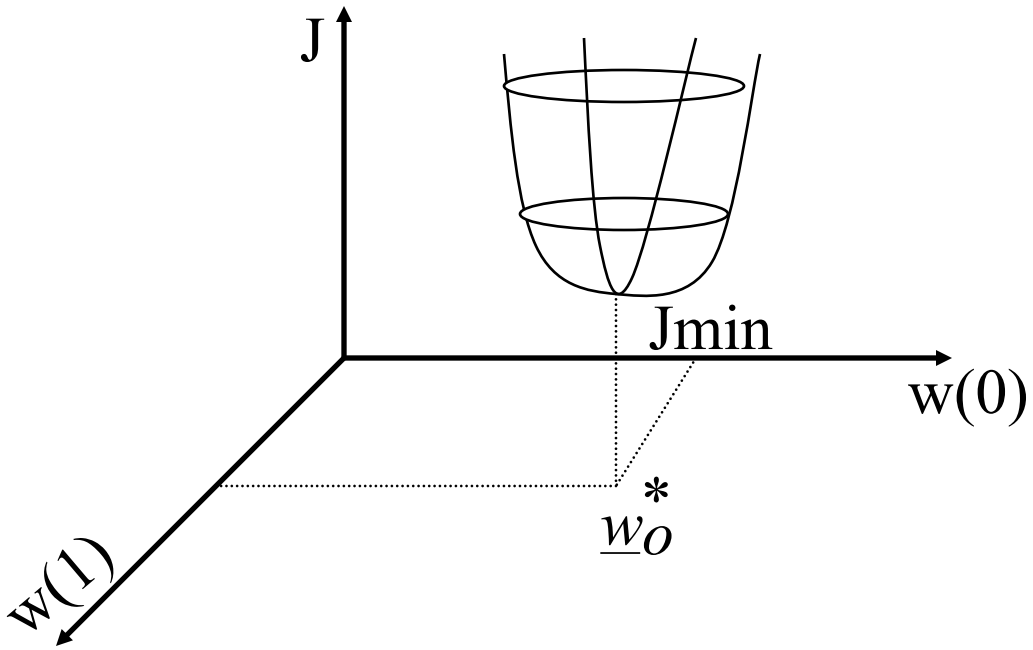
Motivation

- Optimal digital filters designed with fixed filter tap weights have limitations when nonstationary input signals with unknown statistical characteristics are encountered in many applications .

- A more complete solution to nonstationary environment is provided by a continuously adaptive filter .

- In an attempt to build a adaptive filter , we are seeking to find an algorithm which is optimal in some sense (in the least -square sense) but which is responsive to changes in the optimal solution arising as each new data point becomes available .

- We want a filter which will iterate to track changes in the optimal solution .



- Adaptive filters have been successfully applied in several areas including :

(a) adaptive antenna systems in which adaptive filters are used for beam steering and providing nulls in the beam pattern to remove undesired interference .

(b) digital communication receivers in which adaptive filters are used to provide equalization of intersymbol interference and for channel identification .

(c) Adaptive Noise cancellation (ANC) is another useful application of adaptive filtering. ANC is concerned with the enhancement of noise corrupted signals where no a priori knowledge of signal or noise is required .

(d) Adaptive filters are used to estimate the time-delay between two measured signals, an application useful in radar, geophysics and biomedical signal analysis.

(e) System modelling, in which an adaptive filter is used as a model to estimate the characteristics of an unknown system.

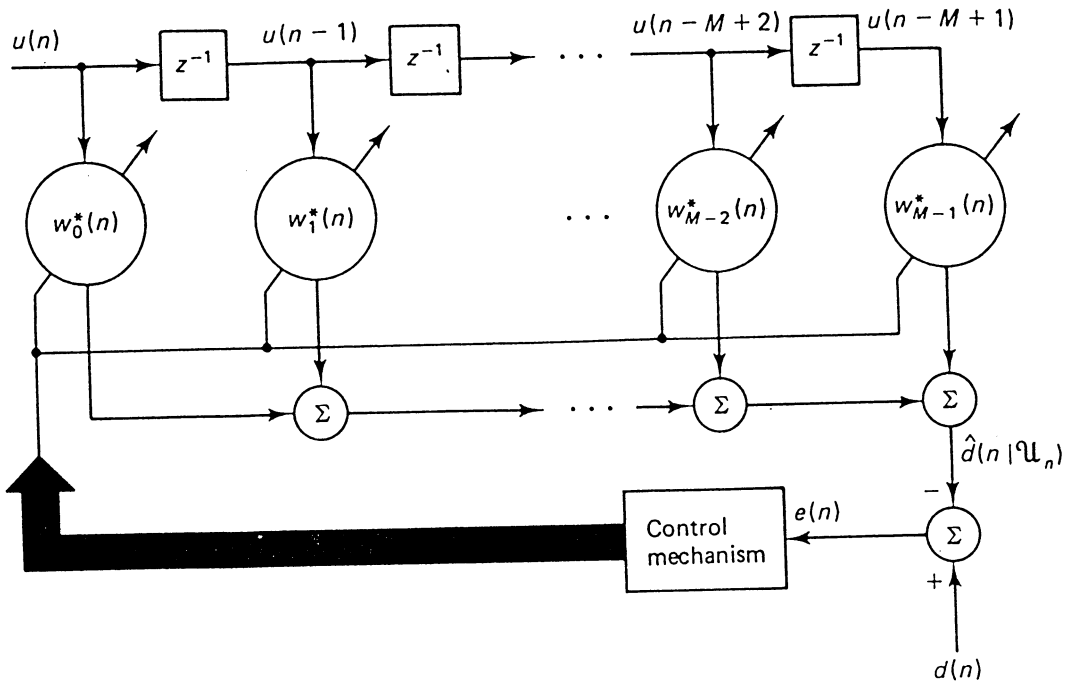


Figure 8.1 Structure of adaptive transversal filter.

- Problem Formulation

From the figure :

Estimation error :

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

where

$$\underline{w}(n) = [w_o(n), w_1(n), \ldots . w_{M-1}(n)]^T$$
$$\underline{u}(n) = [u(n), u(n-1), \ldots ., u(n-M+1)]^T$$

Mean-Squared error :

$$J(n) = \sigma_d^2 - \underline{w}^H(n)\underline{p}\underline{w}(n) + \underline{w}^H(n)\underset{\sim}{R}\underline{w}(n)$$

σ_d^2 = variance of the desired response

$\underline{p}$ = cross-correlation vector between $\underline{u}(n)$ and $d(n)$

$\underset{\sim}{R}$ = correlation matrix of the vector $\underline{u}(n)$

- Note that the variation of the mean-squared error J(n) with time n signifies that error process {e(n)} is nonstationary .

- J(n) depends on $\underline{w}(n)$ and this functional relation-ship generates a bowl-shaped surface with a unique minimum. This surface is known as the error-performance surface of the adaptive filter.

- At the minimum point of the error-performance, $\underline{w}$ takes on the optimum value $\underline{w}_o$ which satisfies Weiner-Hopf equation .

$$\underset{\sim}{R}\,\underline{w}_o = \underline{p}$$

$$J_{\min} = \sigma_d^2 - \underline{p}^H \underline{w}_o$$

- Steepest Descent Algorithm

Find the J_{min} by the following four steps :

1. Choose $\underline{w}(0)$. Typically, $\underline{w}(0)$ is set equal to the null vector .

2. Compare the gradient vector $\nabla(J(n))$. Partials are taken with respect to real and imaginary values of $\underline{w}(n)$. This is done using initial guess.

3. Compute the next guess of $\underline{w}(n)$ by making a change in the initial or present guess in a direction opposite to that of the gradient vector .

4. Go back to Step 2 and repeat the process .

- Updated tap-weight vector at $n+1$:

$$\underline{w}(n+1) = \underline{w}(n) + \frac{1}{2}\mu[-\nabla(J(n))]$$

Substitute :

$$\nabla(J(n)) = -2\underline{p} + 2\underset{\sim}{R}\,\underline{w}(n)$$

where

$$\nabla(J(n)) = \begin{bmatrix} \frac{\partial J(n)}{\partial a_o(n)} + j\frac{\partial J(n)}{\partial b_o(n)} \\ \frac{\partial J(n)}{\partial a_1(n)} + j\frac{\partial J(n)}{\partial b_1(n)} \\ . \\ . \\ \frac{\partial J(n)}{\partial a_{M-1}(n)} + j\frac{\partial J(n)}{\partial b_{M-1}(n)} \end{bmatrix}$$

$$\underline{w}(n) = \underline{a}(n) + j\underline{b}(n)$$

Finally, the updated tap-weight vector $\underline{w}(n+1)$:

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

where μ controls the size of the incremental correction to $\underline{w}(n)$ from one iteration cycle to next :

Interpretation : If we express :

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

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

$$\underline{p} = E[\underline{u}(n)d^*(n)] \qquad \underline{\underline{R}} = E[\underline{u}(n)\underline{u}^H(n)]$$

$$\delta\underline{w}(n+1) = \mu[\, E[\underline{u}(n)d^*(n)] - E[\underline{u}(n)\underline{u}^H(n)]\underline{w}(n)\,]$$

$$= \mu E[\underline{u}(n)\{\underbrace{d^*(n) - \underline{u}^H(n)\underline{w}(n)}_{e(n)}\}]$$

$$= \mu E[\underline{u}(n)e^*(n)]$$

then we may use a bank of cross-correlators to compute the correction vector $\delta\underline{w}(n)$.

Steepest-descent algorithm when represented as a signal flow graph leads to a feedback model.

- Stability

Define a weight-error vector at time n as :

$$\underline{c}(n) = \underline{w}(n) - \underline{w}_o$$

where $\underline{w}_o$, the optimum value satisfies the Weiner-Hopf equation :

$$\underset{\sim}{R}\,\underline{w}_o = \underline{p}$$

Eliminate $\underline{p}$: $\underline{w}(n+1) = \underline{w}(n) + \mu[\underline{p} - \underset{\sim}{R}\,w(n)]$

Now : $\boxed{\underline{c}(n+1) = (\underset{\sim}{I} - \mu\,\underset{\sim}{R})\underline{c}(n)}$

From Unitary transformation , we can represent :

$$\underset{\sim}{R} = \underset{\sim}{Q}\,\underset{\sim}{\Lambda}\,\underset{\sim}{Q}^{H}$$

$\underset{\sim}{Q}$: Unitary matrix $\quad \underset{\sim}{\Lambda} = \mathrm{diag}(\lambda_1, \lambda_2, \ldots \lambda_M)$

Then

$$\underline{c}(n+1) = (I - \mu \underset{\sim}{Q} \underset{\sim}{\Lambda} \underset{\sim}{Q}^H)\underline{c}(n)$$

Now $(Q^H = Q^{-1})$

$$\underset{\sim}{Q}^H \underline{c}(n+1) = (\underset{\sim}{I} - \mu \underset{\sim}{\Lambda})\underset{\sim}{Q}^H \underline{c}(n)$$

Define a new set of coordinates :

$$\underline{v}(n) = \underset{\sim}{Q}^H \underline{c}(n)$$

$$= \underset{\sim}{Q}^H [\underline{w}(n) - \underline{w}_o]$$

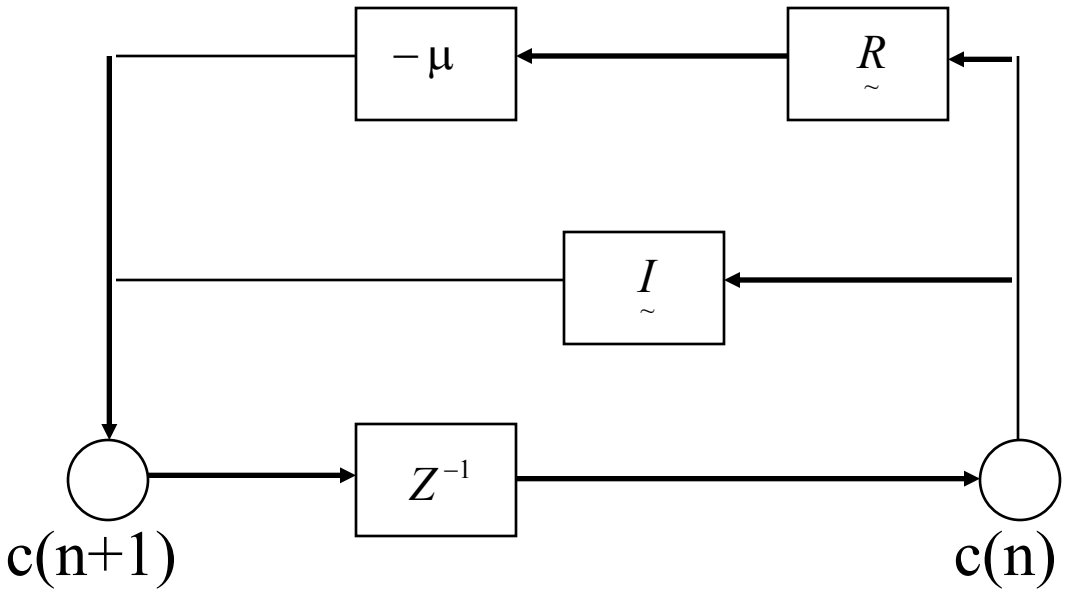
then $\boxed{\underline{v}(n+1) = (\underset{\sim}{I} - \mu \underset{\sim}{\Lambda})\underline{v}(n)}$

with : $\underline{v}(0) = \underset{\sim}{Q}^H [\underline{w}(0) - w_o]$

$$= -\underset{\sim}{Q}^H \underline{w}_o \qquad (\underline{w}(0) = 0)$$

- Two representations

$$\underline{c}(n+1) = (\underset{\sim}{I} - \mu \underset{\sim}{R})\underline{c}(n)$$

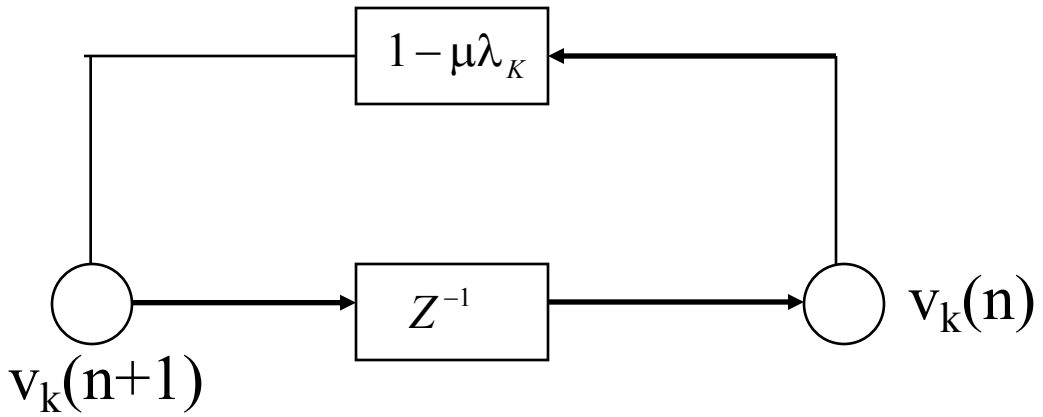


- Eigenvalue representation：

$$\underline{v}(n+1) = (\underset{\sim}{I} - \mu \underset{\sim}{\Lambda})\underline{v}(n)$$

kth natural mode：

$$v_k(n+1) = (1 - \mu\lambda_k)v_k(n)$$



Solving the scalar difference equation :

$$v_k(n) = (1 - \mu\lambda_k)^n v_k(0) \qquad k = 1,2,\ldots M$$

no oscillations : eigenvalues of $\underset{\sim}{R}$ are positive and real

Stability : $-1 < 1 - \mu\lambda_k < 1$ all k

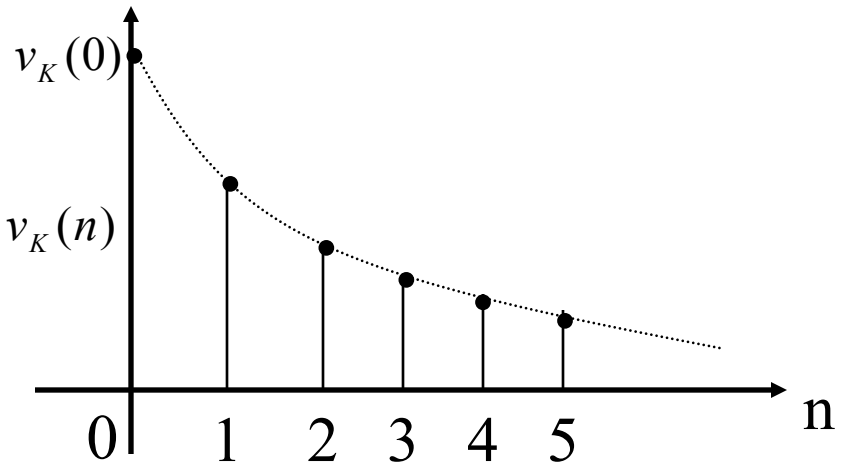
implies : $\underline{w}(n) \to \underline{w}_0$ as $n \to \infty$

the necessary and sufficient condition for the stability of the Steepest-descent algorithm :

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

where $\lambda_{\max}$ is the largest eigenvalue of $\underset{\sim}{R}$

- Time constant equation



Assume the unit of time to be the duration of one iteration cycle.

$$1-\mu\lambda_k = \exp(-1/\tau_k)$$

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

τ_k defines the time required for the amplitude of the kth natural mode $v_k(n)$ to decay to $1/e$ of its initial value $v_k(0)$, e is the base of the natural logarithm. For small μ

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

- Transient behaviour of $\underline{w}(n)$

  Start from :

$$\underline{v}(n) = \underset{\sim}{Q}^{H}[\underline{w}(n) - \underline{w}_o]$$

$$\underset{\sim}{Q}\underline{v}(n) = \underbrace{\underset{\sim}{Q}\underset{\sim}{Q}^{H}}_{\underset{\sim}{I}}[\underline{w}(n) - \underline{w}_o]$$

$$\therefore \quad \underline{w}(n) = \underline{w}_o + \underset{\sim}{Q}\underline{v}(n)$$

$$= \underline{w}_o + [\underline{q}_1\ \underline{q}_2 \cdot \cdot \cdot \cdot \underline{q}_M] \begin{bmatrix} v_1(n) \\ v_2(n) \\ \vdots \\ v_M(n) \end{bmatrix}$$

$$= \underline{w}_o + \sum_{k=1}^{M} \underline{q}_k v_k(n)$$

where

$$v_k(n) = (1 - \mu\lambda_k)^n v_k(0) \qquad k = 1,2,\ldots M$$

$$\text{Now } w_i(n) = w_{0i} + \sum_{k=1}^{M} q_{ki} v_k(0)(1-\mu\lambda_k)^n$$

$$i = 1,2,. \ . \ . \ . \ .M$$

It can be shown :

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

$\tau_a = \text{overall time constant}$.

- Transient behaviour of $J(n)$:
  Start from :

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

By substituting $v_k(n) = (1-\mu\lambda_k)^n v_k(0)$

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

$$\lim_{n\to\infty} J(n) = J_{\min} \quad \text{for any} \quad v_k(0)$$

$J(n)$ vs n is called the learning curve
the exponential decay for kth natural mode has time constant :

$$\tau_{k,\text{mse}} \approx \frac{-1}{2\ln(1-\mu\lambda_k)}$$

For small μ $\quad \tau_{k,\text{mse}} \approx \frac{1}{2\mu\lambda_k}$

4.4 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 4.7 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

$$u(n) + a_1 u(n-1) + a_2 u(n-2) = v(n), \tag{4.33}$$

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

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

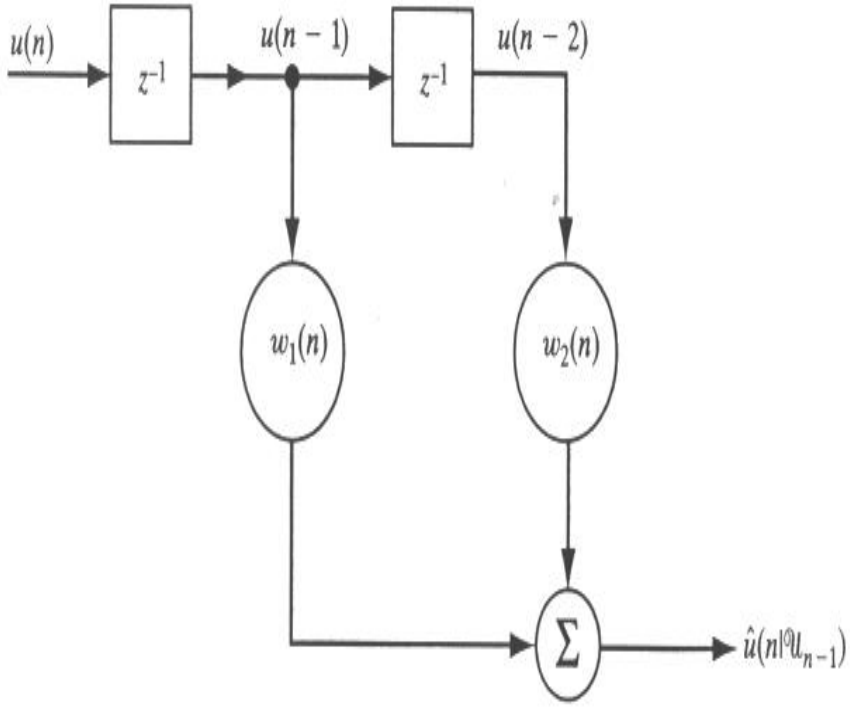


FIGURE 4.7 Two-tap predictor for real-valued input.

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 $v(n)$ 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. 4.7 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 two-by-two symmetric matrix. That is,

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

where (see Chapter 1)

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

and

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

in each of which

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

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

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

and

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

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 associated with the eigenvalues λ_1 and λ_2 are, respectively,

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

and

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

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 4.1.

TABLE 4.1 Summary of Parameter Values Characterizing the Second-Order AR Modeling Problem

Case	AR parameters		Eigenvalues		Eigenvalue spread, $\chi = \lambda_1/\lambda_2$	Minimum mean-square error, $J_{\min} = \sigma_\nu^2$
	a_1	a_2	λ_1	λ_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. (4.21) yields

$$\mathbf{v}(n) = \begin{bmatrix} v_1(n) \\ v_2(n) \end{bmatrix} \tag{4.34}$$

$$= \begin{bmatrix} (1 - \mu\lambda_1)^n v_1(0) \\ (1 - \mu\lambda_2)^n v_2(0) \end{bmatrix}, \quad n = 1, 2, \ldots .$$

To calculate the initial value $\mathbf{v}(0)$, we use Eq. (4.19), 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. 4.7 is optimized, with the second-order AR process of Eq. (4.33) supplying the tap inputs, we find that the optimum tap-weight vector is

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

and the minimum mean-square error is

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

Accordingly, the use of Eq. (4.19) yields the initial value

$$\begin{aligned}\mathbf{v}(0) &= \begin{bmatrix} v_1(0) \\ v_2(0) \end{bmatrix} \\ &= \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} -a_1 \\ -a_2 \end{bmatrix} \\ &= \frac{-1}{\sqrt{2}} \begin{bmatrix} a_1 + a_2 \\ a_1 - a_2 \end{bmatrix}.\end{aligned} \tag{4.35}$$

Thus, for specified parameters, we use Eq. (4.35) to compute the initial value $\mathbf{v}(0)$ and then use Eq. (4.34) to compute $\mathbf{v}(1), \mathbf{v}(2), \ldots$. 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. (4.28) for fixed values of n. For our example, Eq. (4.28) yields

$$J(n) - J_{\min} = \lambda_1 v_1^2(n) + \lambda_2 v_2^2(n). \tag{4.36}$$

When $\lambda_1 = \lambda_2$ and n is fixed, Eq. (4.36) 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. (4.36) 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 $[J(n) - J_{\min}]/\lambda_1$, with $\lambda_1 > \lambda_2$.

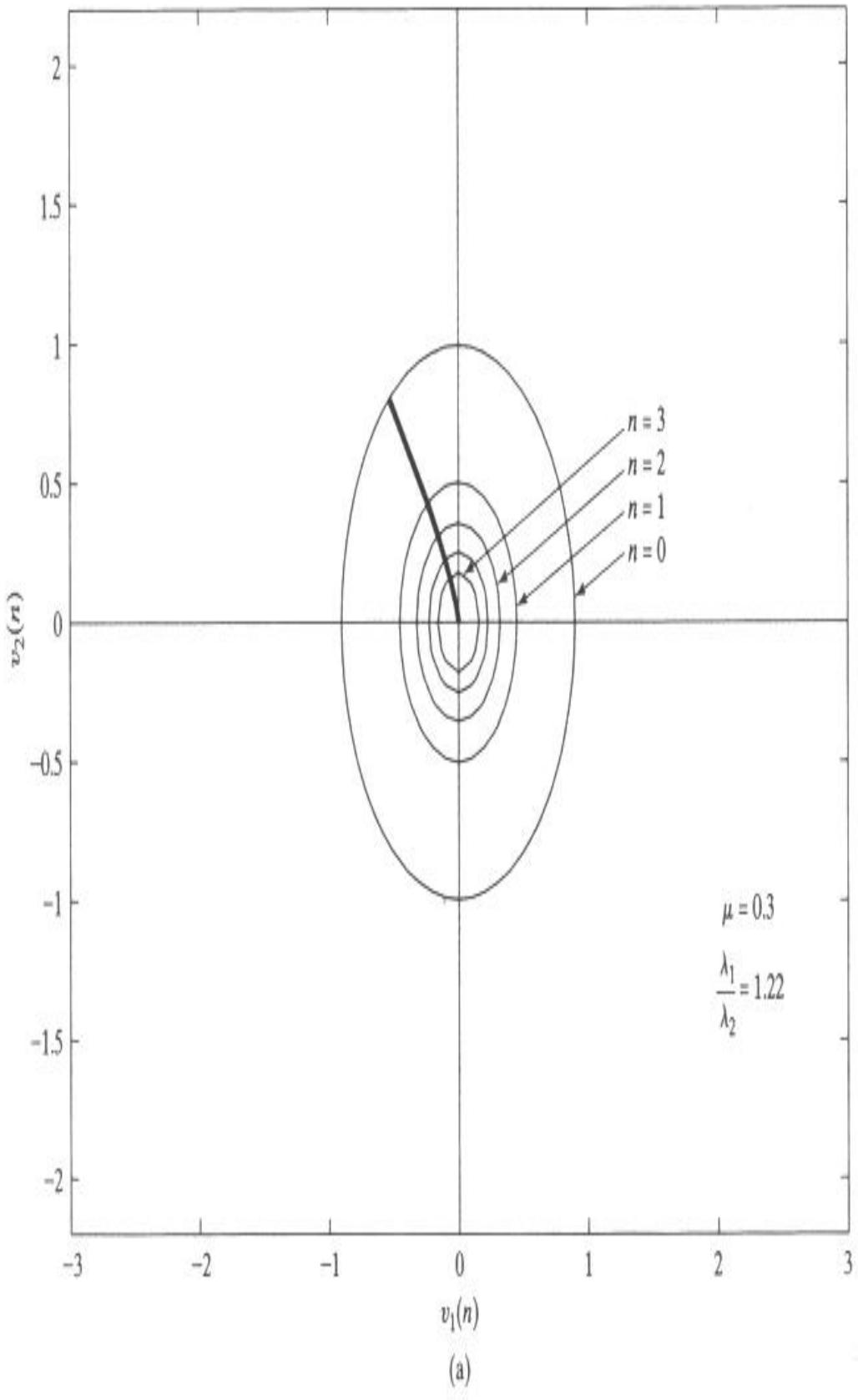
Case 1: Eigenvalue Spread $\chi(\mathbf{R}) = 1.22$. For the parameter values given for Case 1 in Table 4.1, the eigenvalue spread $\chi(\mathbf{R})$ equals 1.22; that is, the eigenvalues λ_1 and λ_2 are approximately equal. The use of these parameter values in Eqs. (4.34) and (4.35) yields the trajectory of $[v_1(n), v_2(n)]$ shown in Fig. 4.8(a), with n as running parameter. The use of same parameter values in Eq. (4.36) 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. (4.25) yields the tap-weight vector

$$\begin{aligned}\mathbf{w}(n) &= \begin{bmatrix} w_1(n) \\ w_2(n) \end{bmatrix} \\ &= \begin{bmatrix} -a_1 - (v_1(n) + v_2(n))/\sqrt{2} \\ -a_2 - (v_1(n) - v_2(n))/\sqrt{2} \end{bmatrix}.\end{aligned} \tag{4.37}$$

The corresponding trajectory of $[w_1(n), w_2(n)]$, with n as a running parameter, obtained by using Eq. (4.37), is shown in Fig. 4.9(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 those of Fig. 4.8(a), are ellipsoidal.

Case 2: Eigenvalue Spread $\chi(\mathbf{R}) = 3$. The use of the parameter values for Case 2 of Table 4.1 in Eqs. (4.34) and (4.35) yields the trajectory of $[v_1(n), v_2(n)]$ shown in


2
1.5
1
0.5
0
−0.5
−1
−1.5
−2
−3
−2
−1
0
1
2
3
$v_2(n)$
$v_1(n)$
$n = 3$
$n = 2$
$n = 1$
$n = 0$
$\mu = 0.3$
$\frac{\lambda_1}{\lambda_2} = 1.22$

(a)

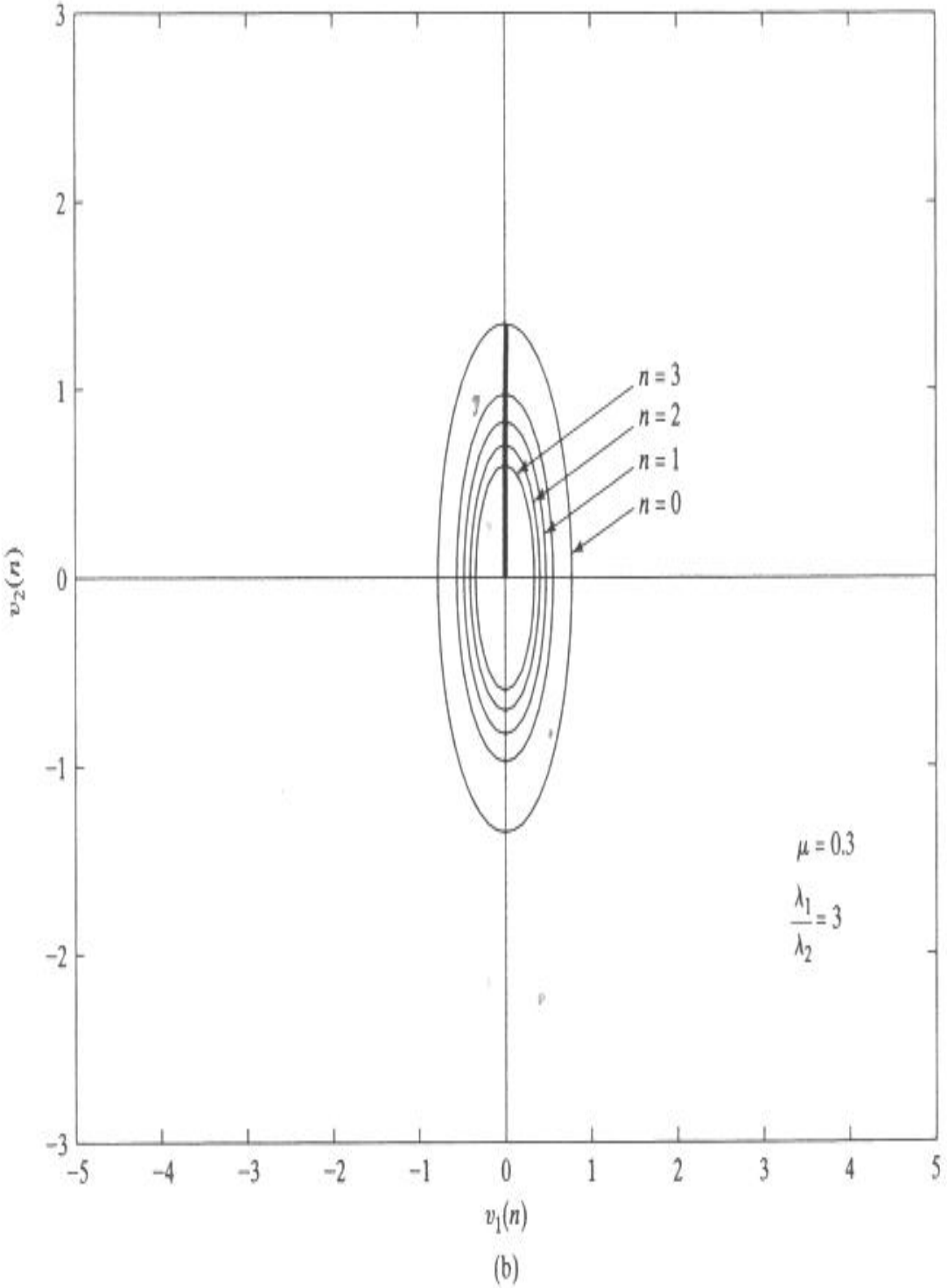
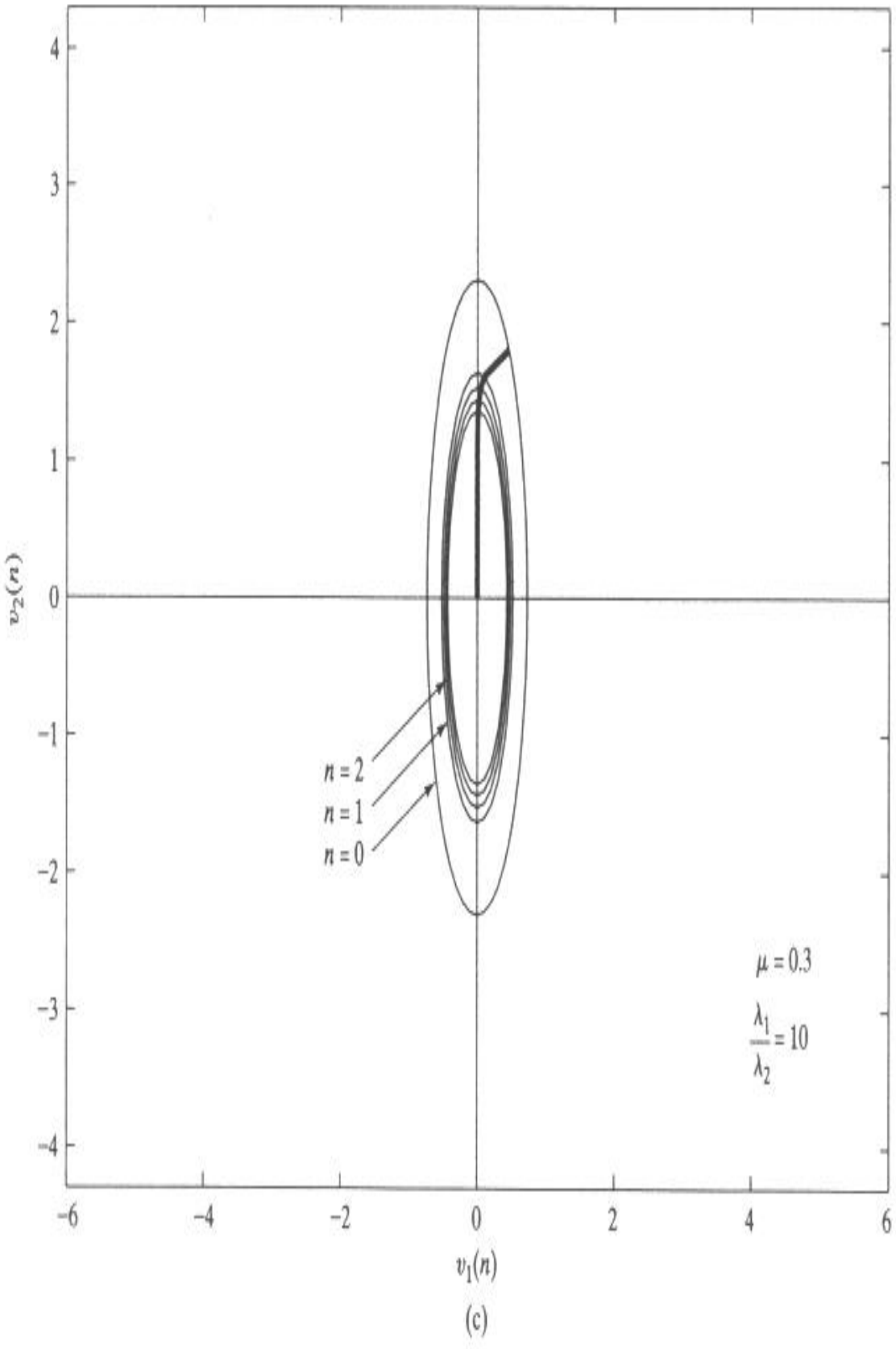


FIGURE 4.8 Loci of $v_1(n)$ versus $v_2(n)$ for the steepest-descent algorithm with step-size parameter $\mu = 0.3$ and varying eigenvalue spread: (a) $\chi(\mathbf{R}) = 1.22$; (b) $\chi(\mathbf{R}) = 3$; (c) $\chi(\mathbf{R}) = 10$; (d) $\chi(\mathbf{R}) = 100$.


4
3
2
1
0
−1
−2
−3
−4
−6
−4
−2
0
2
4
6
$v_2(n)$
$v_1(n)$
$n = 2$
$n = 1$
$n = 0$
$\mu = 0.3$
$\frac{\lambda_1}{\lambda_2} = 10$

(c)

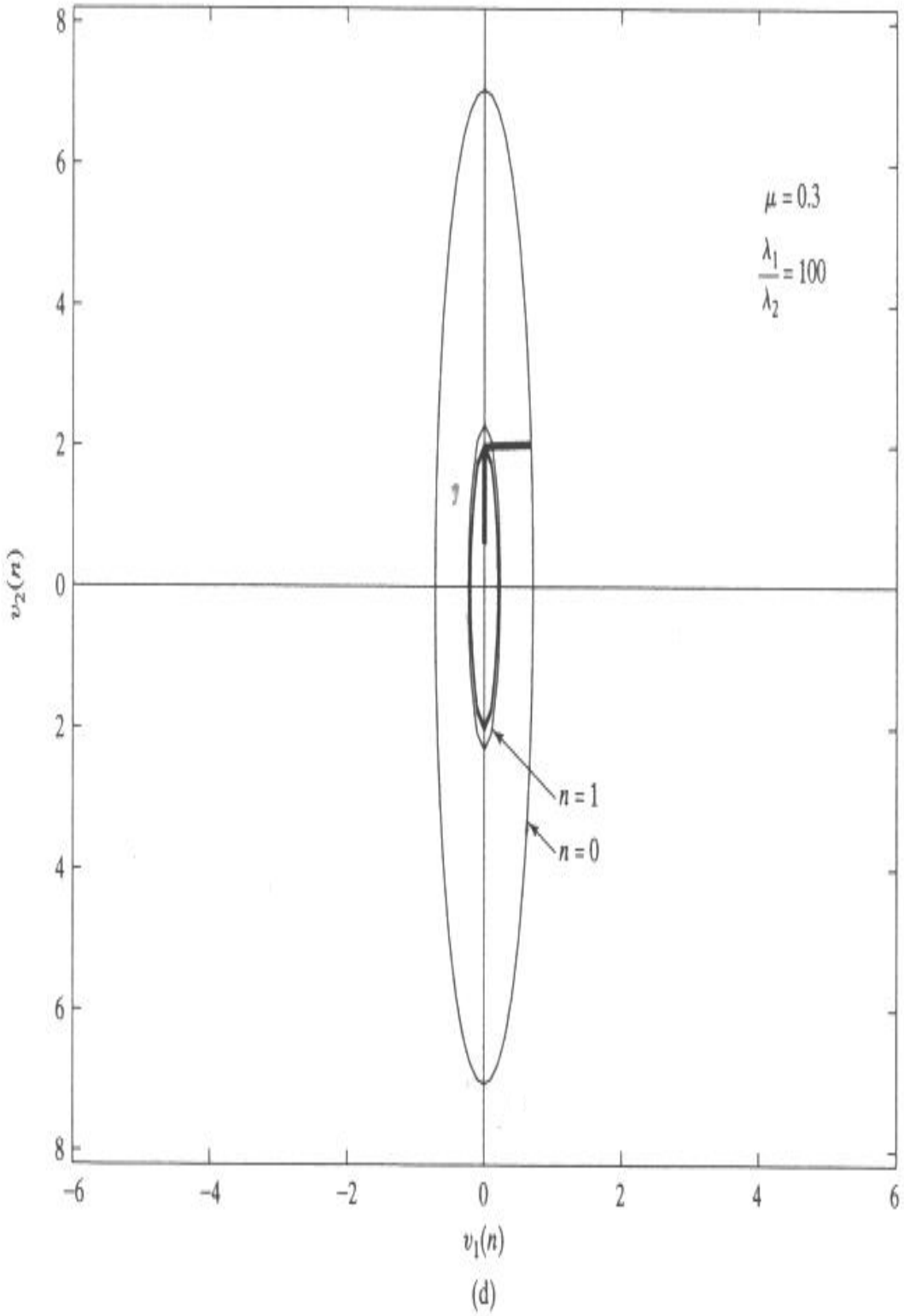
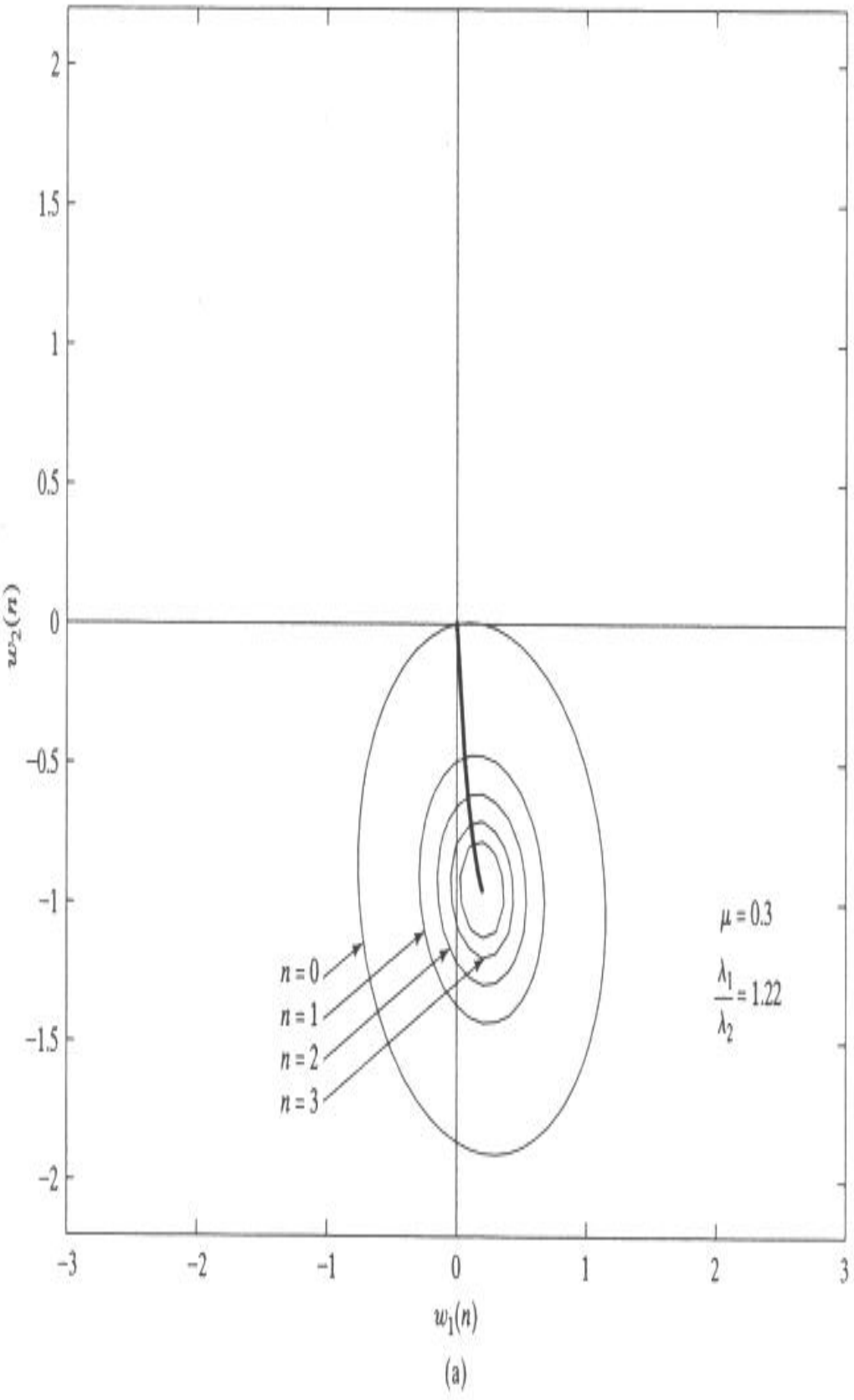


FIGURE 4.8 *(continued)*


2
1.5
1
0.5
0
−0.5
−1
−1.5
−2
$w_2(n)$
−3
−2
−1
0
1
2
3
$w_1(n)$
$n = 0$
$n = 1$
$n = 2$
$n = 3$
$\mu = 0.3$
$\frac{\lambda_1}{\lambda_2} = 1.22$

(a)

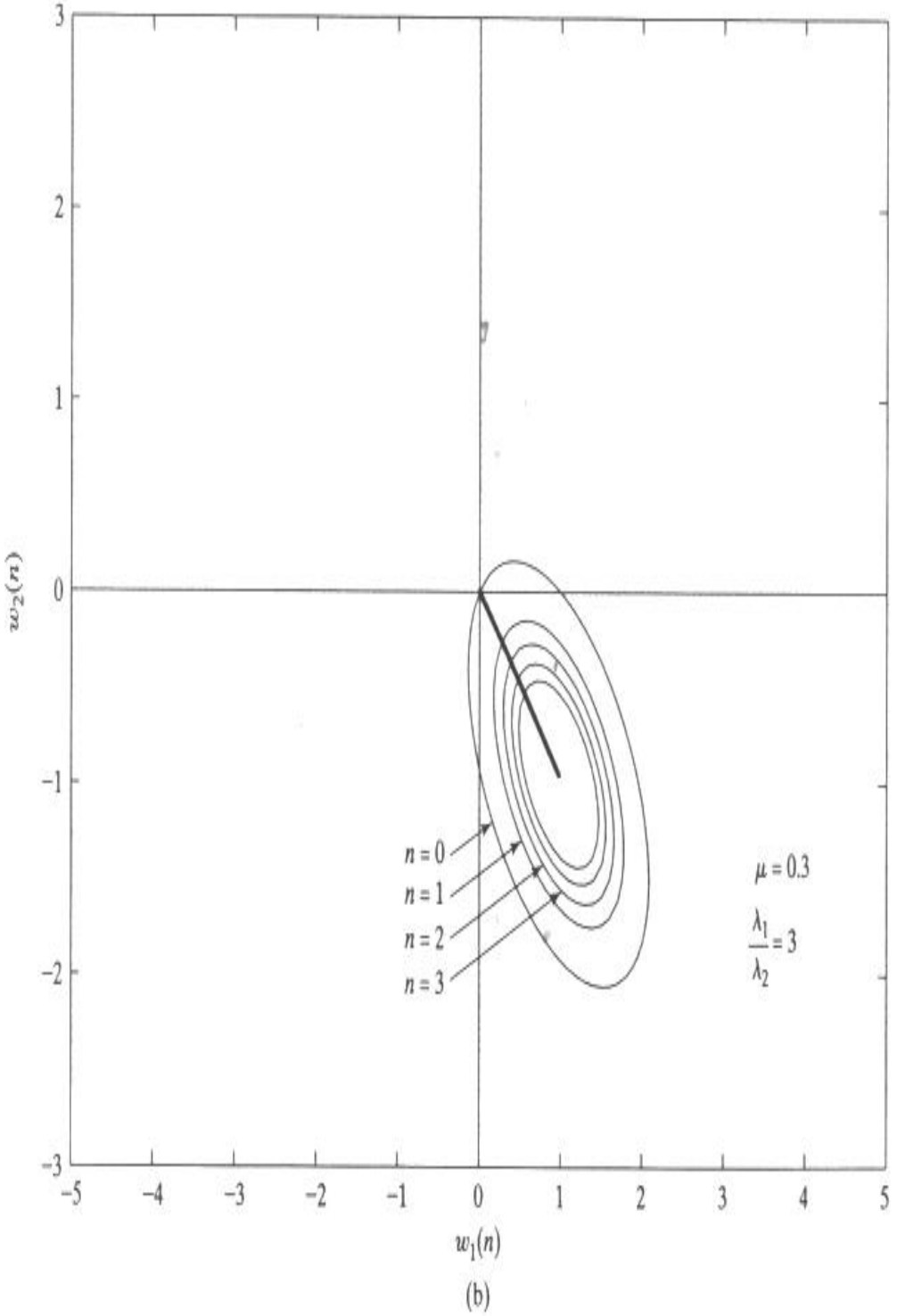
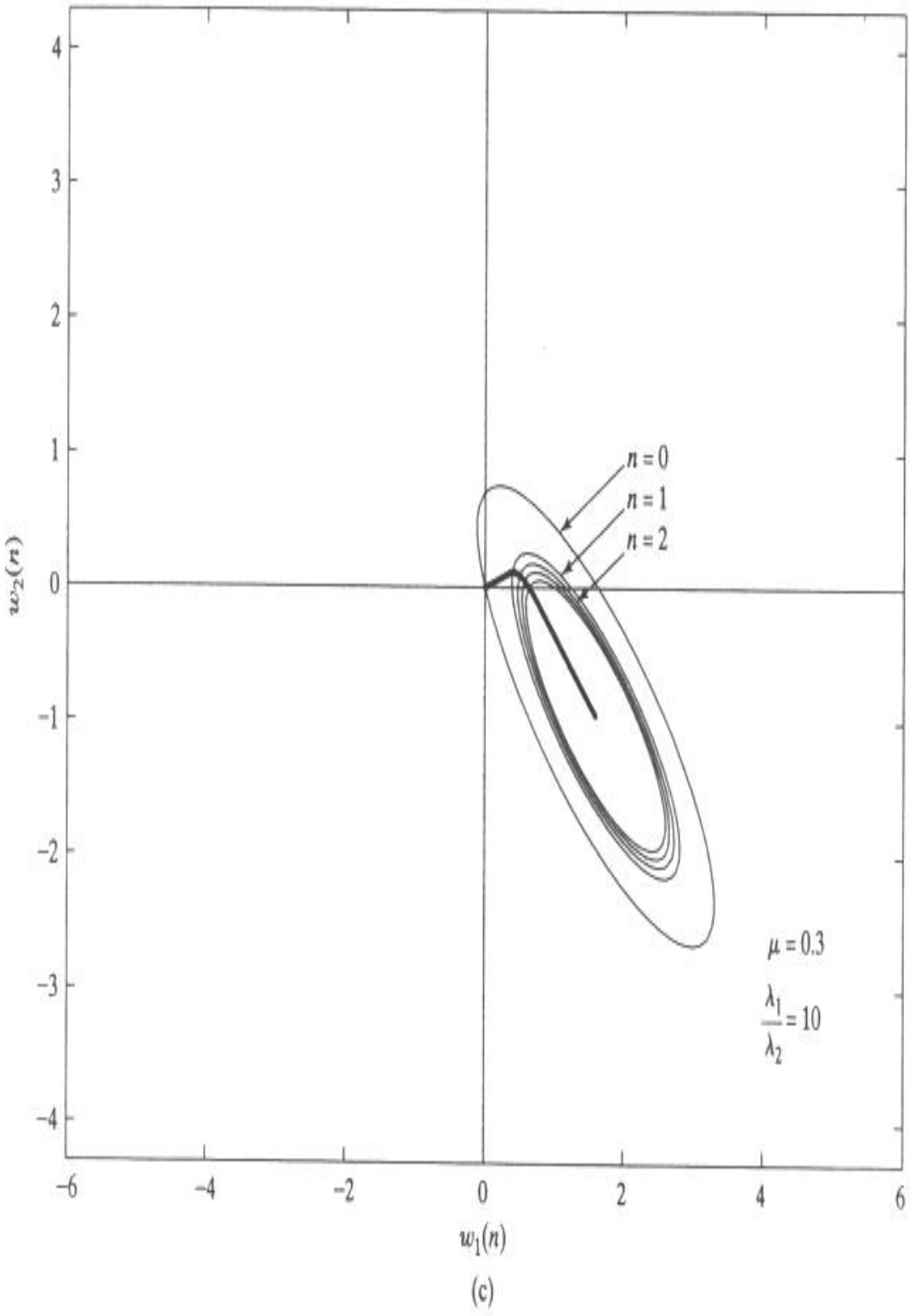
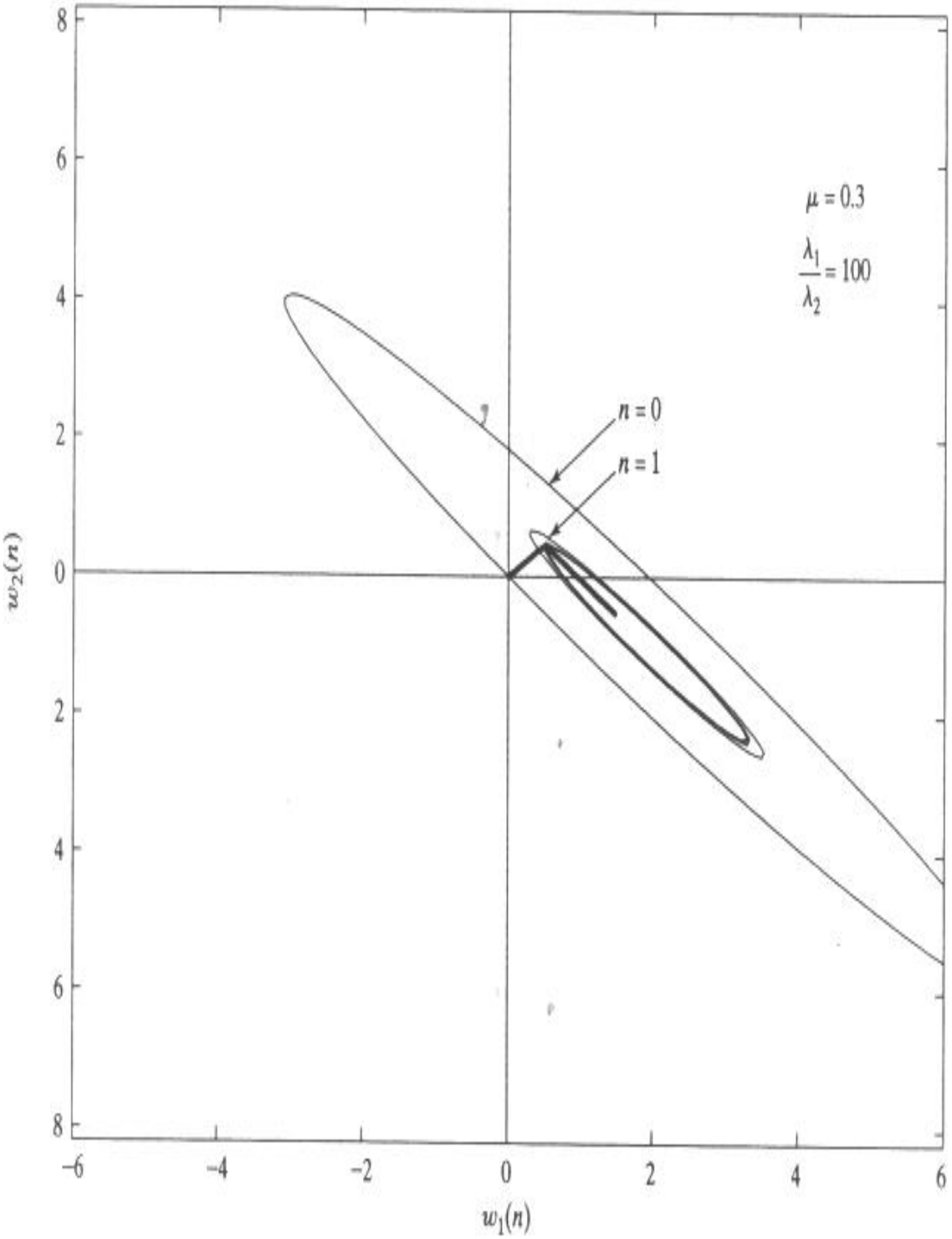


FIGURE 4.9 Loci of $w_1(n)$ versus $w_2(n)$ for the steepest-descent algorithm with step-size parameter $\mu = 0.3$ and varying eigenvalue spread: (a) $\chi(\mathbf{R}) = 1.22$; (b) $\chi(\mathbf{R}) = 3$; (c) $\chi(\mathbf{R}) = 10$; (d) $\chi(\mathbf{R}) = 100$.


4
3
2
1
0
−1
−2
−3
−4
−6
−4
−2
0
2
4
6
$w_2(n)$
$w_1(n)$
$n = 0$
$n = 1$
$n = 2$
$\mu = 0.3$
$\frac{\lambda_1}{\lambda_2} = 10$

(c)



(d)

FIGURE 4.9 *(continued)*

Fig. 4.8(b), with n as running parameter, and the use of the same parameter values in Eq. (4.36) 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. 4.9(b).

Case 3: Eigenvalue Spread $\chi(\mathbf{R}) = 10$. For this case, the application of Eqs. (4.34) and (4.35) yields the trajectory of $[v_1(n), v_2(n)]$ shown in Fig. 4.8(c), with n as running parameter, and the application of Eq. (4.36) yields the ellipsoidal loci included in the 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. 4.9(c).

Case 4: Eigenvalue Spread $\chi(\mathbf{R}) = 100$. For this case, the application of the preceding equations yields the results shown in Fig. 4.8(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. 4.9(d).

In Fig. 4.10, we have plotted the mean-square 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-square error $J_{\min}$ decreases. This observation makes intuitive sense: The predictor should do a better job tracking a strongly correlated input process than a weakly correlated one.

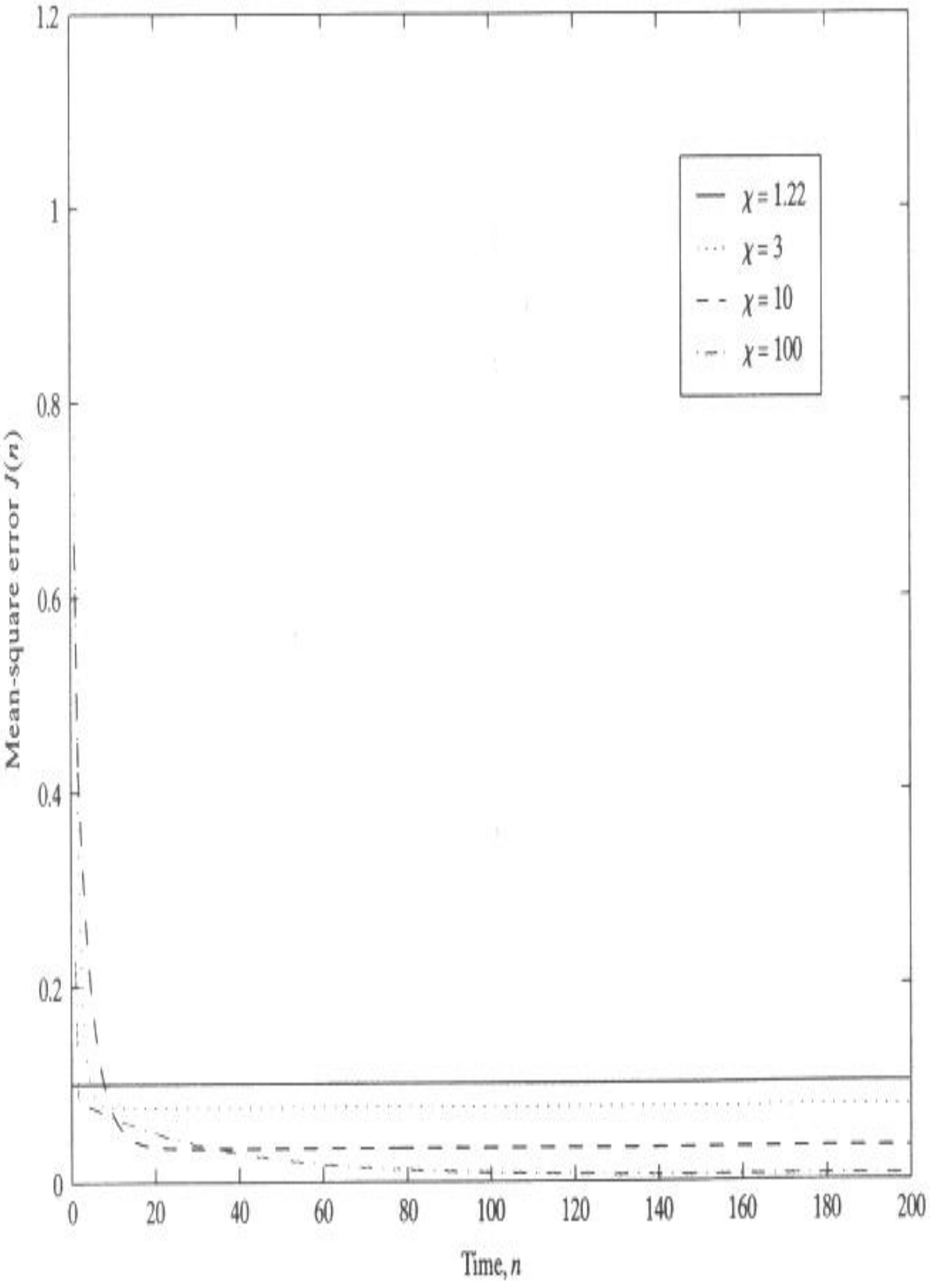


FIGURE 4.10 Learning curves of steepest-descent algorithm with step-size parameter $\mu = 0.3$ and varying eigenvalue spread.

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. 4.11, respectively. The results included in part (a) of this figure are the same as those in Fig. 4.8(c). Note that, in accordance with Eq. (4.22), 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. 4.11(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. 4.12, respectively. Here again, the results included in part (a) of the figure are the same as those in Fig. 4.9(c).

Observations

On the basis of the results presented for Experiments 1 and 2, we may make the following observations:

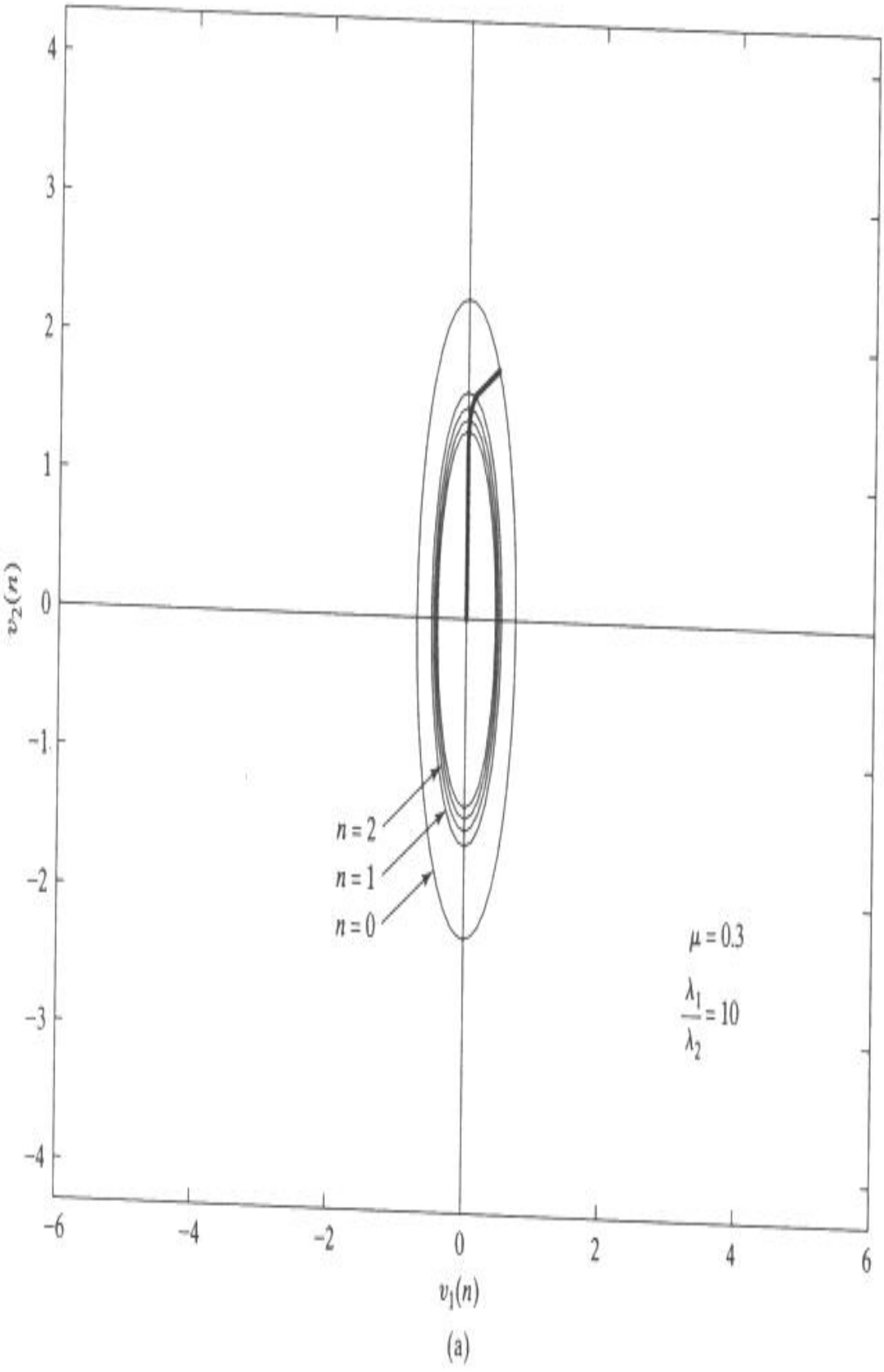
1. The trajectory of $[v_1(n), v_2(n)]$, with the number of iterations n as running parameter, is normal to the locus of $[v_1(n), v_2(n)]$ for fixed $J(n)$. This statement also applies to the trajectory of $[w_1(n), w_2(n)]$ for fixed $J(n)$.
2. When the eigenvalues λ_1 and λ_2 are equal, the trajectory 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 situation is illustrated in Fig. 4.8(a) or 4.9(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 situation is illustrated in Fig. 4.8(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. 4.9(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. 4.8(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. 4.9(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. 4.8(d) and 4.9(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 properly, for which cases the trajectory formed by joining the points $v(0), v(1), v(2), \ldots$, is a straight line, the shortest possible path.


4
3
2
1
0
−1
−2
−3
−4
−6
−4
−2
0
2
4
6
$v_2(n)$
$v_1(n)$
$n = 2$
$n = 1$
$n = 0$
$\mu = 0.3$
$\frac{\lambda_1}{\lambda_2} = 10$

(a)

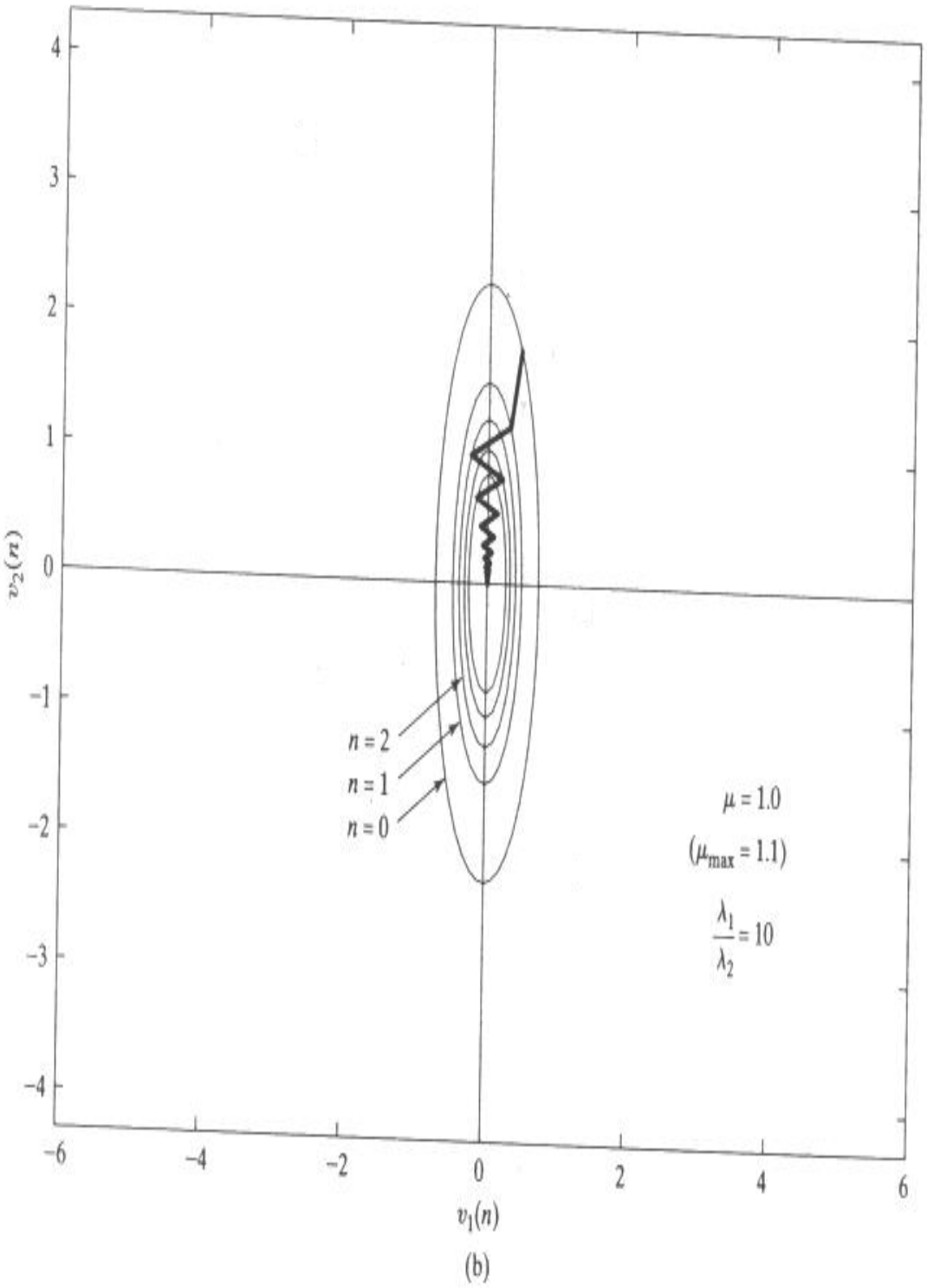
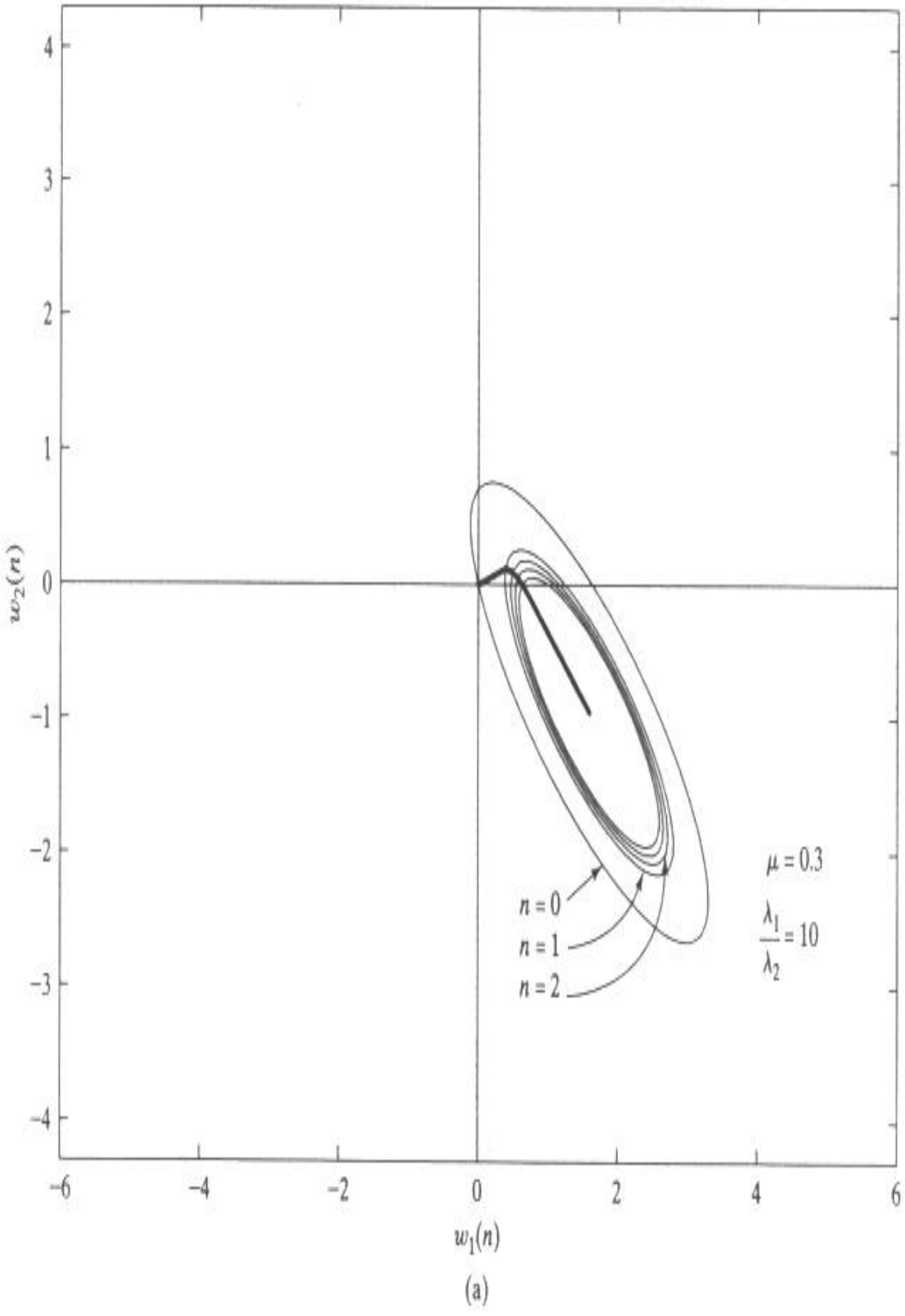


FIGURE 4.11 Loci of $v_1(n)$ versus $v_2(n)$ for the steepest-descent algorithm with eigenvalue spread $\chi(\mathbf{R}) = 10$ and varying step-size parameters: (a) overdamped, $\mu = 0.3$; (b) underdamped, $\mu = 1.0$.


4
3
2
1
0
−1
−2
−3
−4
$w_2(n)$
−6
−4
−2
0
2
4
6
$w_1(n)$
$n = 0$
$n = 1$
$n = 2$
$\mu = 0.3$
$\frac{\lambda_1}{\lambda_2} = 10$

(a)

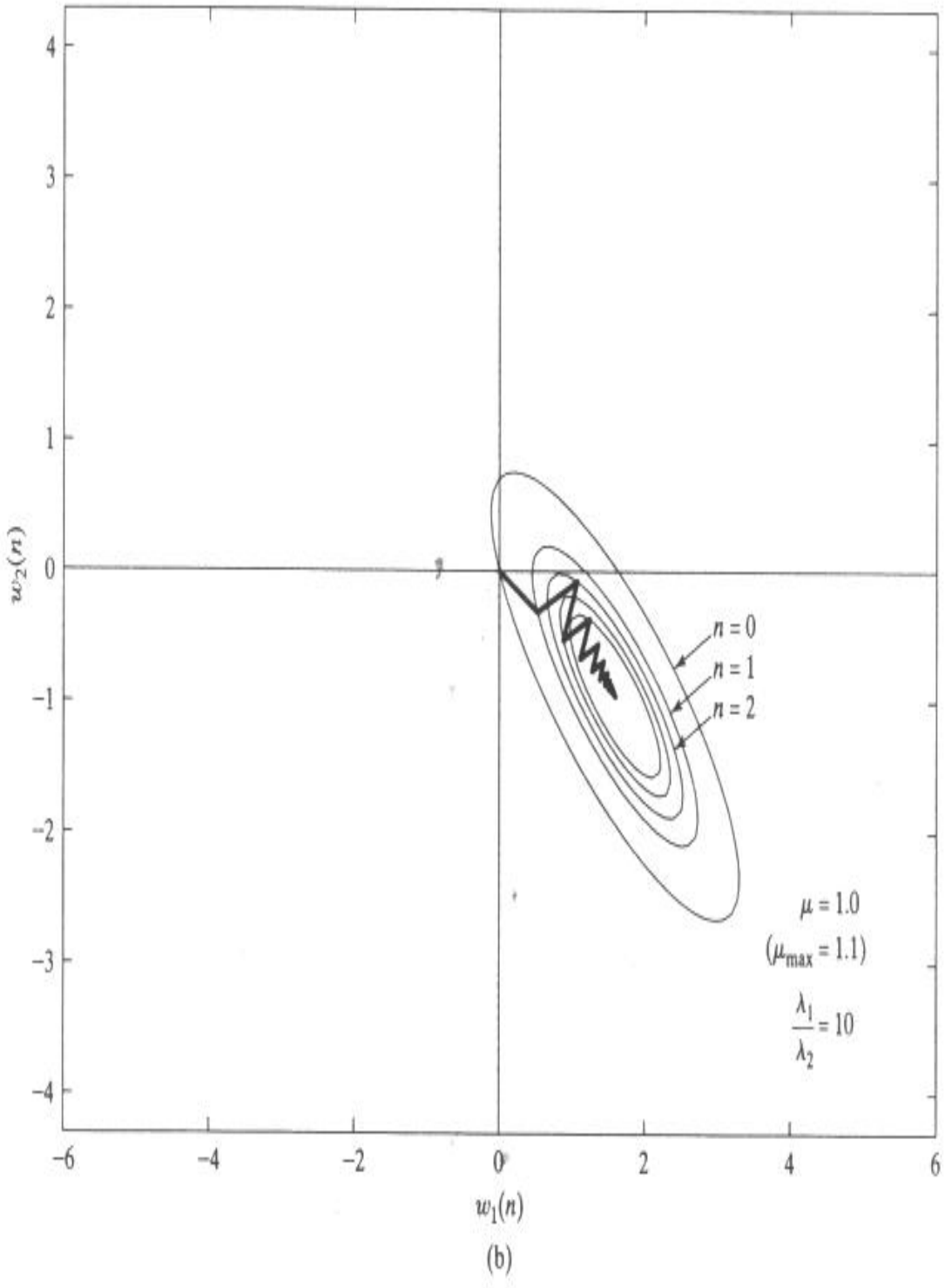


FIGURE 4.12 Loci of $w_1(n)$ versus $w_2(n)$ for the steepest-descent algorithm with eigenvalue spread $\chi(\mathbf{R}) = 10$ and varying step-size parameters: (a) overdamped, $\mu = 0.3$; (b) underdamped, $\mu = 1.0$.

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 $\mathbf{v}(0), \mathbf{v}(1), \mathbf{v}(2), \ldots$ 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. 4.11 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. 4.12.

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.

4.5 THE STEEPEST-DESCENT ALGORITHM AS A DETERMINISTIC SEARCH METHOD

The error-performance surface of an adaptive transversal filter operating in a wide-sense stationary stochastic environment is a bowl-shaped (i.e., quadratic) surface with a distinct minimum point. The steepest-descent algorithm provides a local search method for seeking the minimum point of the error-performance surface, starting from an arbitrary initial point. For its operation, the steepest-descent algorithm depends on three quantities:

- The *starting point*, which is specified by the initial value $\mathbf{w}(0)$ of the tap-weight vector.
- The *gradient vector*, which, at a particular point on the error-performance surface (i.e., a particular value of the tap-weight vector), is uniquely determined by the cross-correlation vector $\mathbf{p}$ and the correlation matrix $\mathbf{R}$ that characterize the environment.
- The *step-size parameter* μ, which controls the size of the incremental change applied to the tap-weight vector of the transversal filter from one iteration of the algorithm to the next; for stability of the algorithm, μ must satisfy the condition of Eq. (4.22).

Once these three quantities are specified, the steepest-descent algorithm follows a distinct path in the multidimensional weight space, starting from the initial point $\mathbf{w}(0)$ and ultimately terminating on the optimum solution $\mathbf{w}_o$. In other words, the steepest-descent algorithm is a *deterministic search method* in weight space. This statement is confirmed by the experimental results presented in Section 4.4. In theory, the algorithm requires an "infinite" number of iterations to move from the starting point $\mathbf{w}(0)$ to the optimum point $\mathbf{w}_o$. However, in practice, we need to execute just a "finite" number of iterations of the algorithm for the transversal filter to attain a tap-weight vector close enough to the optimum solution $\mathbf{w}_o$—closeness is clearly a subjective matter that can be determined only by a designer's objective.

4.6 VIRTUE AND LIMITATION OF THE STEEPEST-DESCENT ALGORITHM

The important virtue of the steepest-descent algorithm is the *simplicity* of its implementation, which is readily seen from Eq. (4.10). However, as pointed out in Section 4.5, we may require a large number of iterations for the algorithm to converge to a point sufficiently close to the optimum solution $\mathbf{w}_o$. This performance limitation is due to the fact that the steepest-descent algorithm is based on a straight-line (i.e., first-order) approximation of the error-performance surface around the current point.

Newton's Method

To overcome the aforesaid limitation of the steepest-descent algorithm, we may use a quadratic (i.e., second-order) approximation of the error-performance surface around the current point denoted by $\mathbf{w}(n)$. Then, invoking the second-order *Taylor series* expansion of the cost function $J(\mathbf{w})$ around $\mathbf{w}(n)$, we have

$$J(\mathbf{w}) \approx J(\mathbf{w}(n)) + (\mathbf{w} - \mathbf{w}(n))^H \mathbf{g}(n) + \frac{1}{2}(\mathbf{w} - \mathbf{w}(n))^H \mathbf{H}(n)(\mathbf{w} - \mathbf{w}(n)), \tag{4.38}$$

where the superscript H denotes Hermitian transposition, the vector

$$\mathbf{g}(n) = \left.\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}}\right|_{\mathbf{w}=\mathbf{w}(n)} \tag{4.39}$$

is the gradient evaluated at $\mathbf{w}(n)$, and the matrix

$$\mathbf{H}(n) = \left.\frac{\partial^2 J(\mathbf{w})}{\partial \mathbf{w}^2}\right|_{\mathbf{w}=\mathbf{w}(n)} \tag{4.40}$$

is the *Hessian* of the cost function evaluated at $\mathbf{w}(n)$. The straight-line approximation of $J(\mathbf{w})$ around the current point $\mathbf{w}(n)$ is clearly a simplification of Eq. (4.38). Differentiating that equation with respect to $\mathbf{w}$ and setting the result to zero, we find that the next iterate (i.e., the updated point on the error-performance surface) is given by

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \mathbf{H}^{-1}\mathbf{g}(n), \tag{4.41}$$

where $\mathbf{H}^{-1}(n)$ is the *inverse* of the Hessian $\mathbf{H}(n)$. This iterative equation is the pure *Newton method* of optimization theory. (See Problem 15 for a modified form of Newton's algorithm.)

For the quadratic cost function of Eq. (4.8), the gradient vector is defined by Eq. (4.9). Moreover, differentiating the last line of Eq. (4.9) with respect to $\mathbf{w}(n)$, we get

$$\mathbf{H}(n) = 2\mathbf{R}. \tag{4.42}$$

That is, except for a scaling factor, the Hessian of the quadratic cost function of Eq. (4.8) is exactly equal to the correlation matrix $\mathbf{R}$ of the tap-input vector $\mathbf{u}(n)$. Hence, substituting Eqs. (4.9) and (4.42) into Eq. (4.41), we obtain

$$\begin{aligned}\mathbf{w}(n+1) &= \mathbf{w}(n) - \frac{1}{2}\mathbf{R}^{-1}\left(-2\mathbf{p} + 2\mathbf{R}\mathbf{w}(n)\right)\\ &= \mathbf{R}^{-1}\mathbf{p}\\ &= \mathbf{w}_o.\end{aligned} \tag{4.43}$$

Equation (4.43) shows that Newton's method attains the optimum solution $\mathbf{w}_o$ from an arbitrary point $\mathbf{w}(n)$ on the error surface in a single iteration. However, this improvement in performance requires the inversion of the correlation matrix $\mathbf{R}$, the foregoing of which is the very thing that motivates the use of the steepest-descent algorithm.

The conclusion to be drawn from this discussion is that if computational simplicity is of paramount importance, then the method of steepest descent is the preferred iterative method for computing the tap-weight vector of an adaptive transversal filter operating in a wide-sense stationary environment. If, on the other hand, the rate of convergence is the issue of interest, then Newton's method or a modified version of it is the preferred approach.

4.7 SUMMARY

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

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

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.