Adaptive Filters

V. John Mathews
Scott C. Douglas

Contents

Chapter 4

Stochastic Gradient Adaptive Filters

This chapter introduces a class of adaptive filters that employ a *gradient descent* optimization procedure. Implementing this procedure exactly requires knowledge of the input signal statistics, which are almost always unknown for real-world problems. Instead, an approximate version of the gradient descent procedure can be applied to adjust the adaptive filter coefficients using only the measured signals. Such algorithms are collectively known as *stochastic gradient* algorithms.

4.1 Gradient Adaptation

We introduce the method of gradient descent in this section using a real-world analogy. We develop the concept of a *cost function* using this analogy. The gradient descent procedure can be used to find the minimum of this function. We then apply these ideas to the adaptive filtering problem and derive an entire family of *stochastic gradient* adaptive filters.

4.1.1 An Analogy

Consider Figure 4.1, which shows a bowl-shaped surface and a ball perched on the edge of this bowl. If we were to let this ball go, gravity would cause the ball to roll down the sides of this bowl to the bottom.

If we observe the ball's movement from directly above the bowl, its path would look something like that shown in Figure 4.2. The elliptical curves in the figure denote contours of equal height, and the path that the ball travels is indicated by the dotted line. Gravity's net pull on the ball at any time instant would be in a direction perpendicular to the line that is tangential to the contour line at the ball's current location. Moreover, the ball would descend faster for steeper sections of the bowl.

The shape of the bowl's surface plays an important role in the path the ball takes to reach the bottom. In particular, if the surface has two or more depressions where the ball could sit idle, there is no guarantee that the ball will descend to the lowest point on the surface.

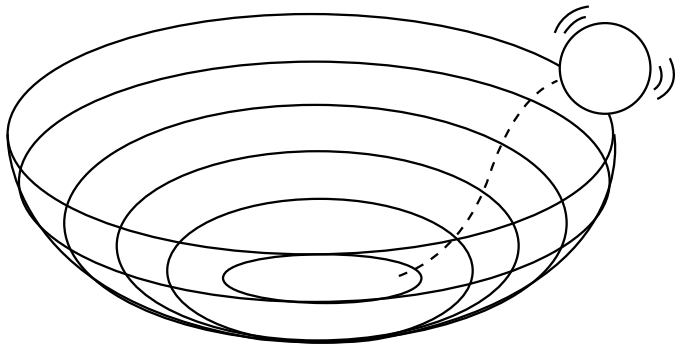

Figure 4.1: A ball rolling into a valley is a useful analogy for visualizing the method of steepest descent.

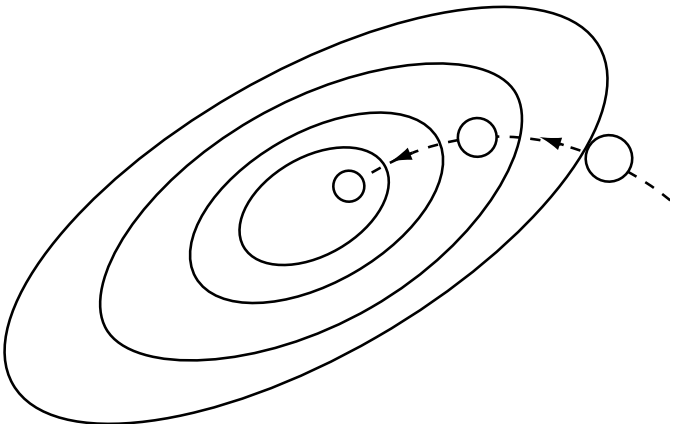

Figure 4.2: The path of a ball descending into the valley.

Figure 4.3 shows a surface with two depressions, which are also known as *local minima* of the surface. A ball placed nearer to the right-most local minimum will travel to that local minimum as opposed to the lower *global minimum* point on the left.

4.1.2 The Method of Steepest Descent

The above simple analogy illustrates some of the features of an optimization procedure called the *method of steepest descent*. As the name implies, the method relies on the slope at any point on the surface to provide the best direction in which to move. The steepest descent direction gives the greatest change in elevation of the surface of the cost function for a given step laterally. The steepest descent procedure uses the knowledge of this direction to move to a lower point on the surface and find the bottom of the surface in an iterative manner.

Mathematical Preliminaries

Consider a system identification problem in which we wish to have the output of a linear filter match a desired response signal $d(n)$ as closely as possible. For simplicity of our discussion, we choose the FIR filter structure for the system model. The output of this filter is given by

$$\begin{aligned} \widehat{d}(n) &= \sum_{i=0}^{L-1} w_i(n)x(n-i) \\ &= \mathbf{W}^T(n)\mathbf{X}(n), \end{aligned} \tag{4.1}$$

where $\mathbf{X}(n) = [x(n) \;\; x(n-1) \;\; \cdots \;\; x(n-L+1)]^T$ is a vector of input signal samples and $\mathbf{W}(n) = [w_0(n) \;\; w_2(n) \;\; \cdots \;\; w_{L-1}(n)]^T$ is a vector containing the coefficients of the FIR filter at time n.

Our objective is to find the coefficient vector $\mathbf{W}(n)$ that "best" models the input-output relation of the unknown system such that some positive-valued cost function of the estimation error

$$e(n) = d(n) - \widehat{d}(n), \tag{4.2}$$

is the smallest among all possible choices of the coefficient vector. An additional constraint on this cost function is that it has no local minima, due to the nature of the search method as illustrated by our analogy.

Cost Functions

We need to define an appropriate *cost function* to formulate the steepest descent algorithm mathematically. In analogy with the example discussed above, this cost function provides a surface on which we can descend to find the lowest point. The location of this lowest point defines the optimum values for the coefficients.

For our main discussion, we consider the *mean-square-error cost function* defined in Chapter 2 as

$$\begin{aligned} J(n) &= E\{(e(n))^2\} \\ &= E\{(d(n) - \mathbf{W}^T(n)\mathbf{X}(n))^2\}. \end{aligned} \tag{4.3}$$

Recall from Chapter 2 that $J(n)$ is a quadratic, non-negative function of the coefficient vector. If the autocorrelation matrix $\mathbf{R}_{\mathbf{xx}}(n)$ is invertible, the cost function has a unique minimum given by

$$\mathbf{W}_{opt}(n) = \mathbf{R}_{\mathbf{xx}}^{-1}(n)\mathbf{P}_{\mathrm{d}\mathbf{x}}(n). \tag{4.4}$$

Our objective is to iteratively descend to the bottom of the cost function surface, so that $\mathbf{W}(n)$ approaches $\mathbf{W}_{opt}(n)$, using a strategy analogous to that of the ball rolling in a bowl.

The Algorithm

Consider Figure 4.4, which shows the mean-square-error cost function for a single-coefficient FIR filter with parameter $w_1(n)$. Shown in the figure are five different points in the range of the unknown parameter, along with the tangents of the cost function at each point. We notice the following facts from the figure:

1. The cost function has no local minima.

2. At the optimum parameter value associated with the minimum of the cost function, the slope of the function is zero.

3. The slope of the cost function is always positive at points located to the right of the optimum parameter value. Conversely, the slope of the cost function is always negative at points located to the left of the optimum parameter value.

4. For any given point, the larger the distance from this point to the optimum value, the larger is the magnitude of the slope of the cost function.

These facts suggest an iterative approach for finding the parameter value associated with the minimum of the cost function: simply move the current parameter value in the direction opposite to that of the slope of the cost function at the current parameter value. Furthermore, if we make the magnitude of the change in the parameter value proportional to the magnitude of the slope of the cost function, the algorithm will make large adjustments of the parameter value when its value is far from the optimum value and will make smaller adjustments to the parameter value when the value is close to the optimum value. This approach is the essence of the steepest descent algorithm.

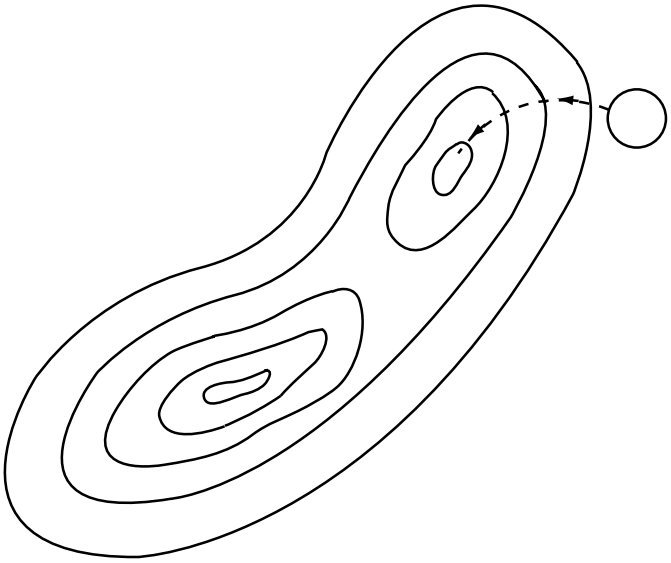

Figure 4.3: A ball cannot be expected to descend to the lowest point on a surface with multiple depressions.

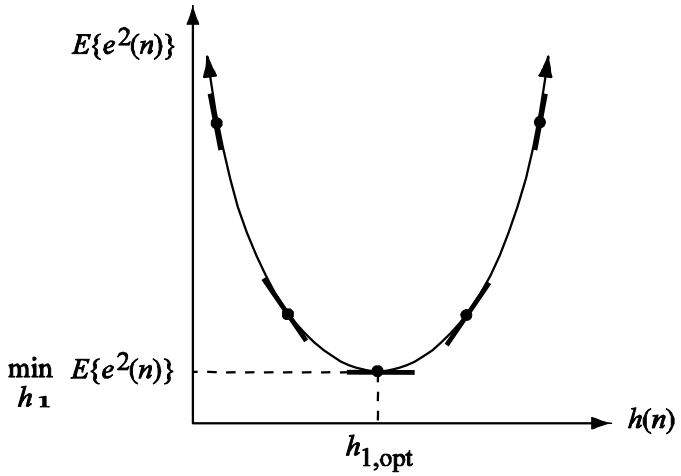

Figure 4.4: Mean-square-error cost function for a single-coefficient FIR filter.

We can generalize the above approach for an arbitrary cost function $J(n)$ and a vector of parameters $\mathbf{W}(n)$. The new coefficient vector $\mathbf{W}(n+1)$ is computed in this case as

$$\mathbf{W}(n+1) \;=\; \mathbf{W}(n) - \alpha \frac{\partial J(n)}{\partial \mathbf{W}(n)}, \tag{4.5}$$

where $\partial J(n)/\partial \mathbf{W}(n)$ denotes a vector whose ith value is given by $\partial J(n)/\partial w_i(n)$ and α is a proportionality constant. This vector is known as the *gradient* of the error surface.

For the mean-square-error cost function, the above algorithm becomes

$$\mathbf{W}(n+1) \;=\; \mathbf{W}(n) - \frac{\mu}{2} \frac{\partial E\{e^2(n)\}}{\partial \mathbf{W}(n)}, \tag{4.6}$$

where we have defined $\alpha = \mu/2$. The parameter μ is termed the *step size* of the algorithm. The additional factor of $1/2$ in (4.6) is introduced for notational convenience.

Characteristics of Cost Functions

We are not limited to mean-square-error cost functions or those that depend on statistical expectations. In general, we can consider arbitrary functions of the error $\phi(e(n))$ that have the following characteristics:

1. The function $\phi(e(n))$ is an *even function* of the estimation error signals; *i.e.,* $\phi(e(n)) = \phi(-e(n))$.
2. The function $\phi(e(n))$ is *monotonically-increasing in the argument* $|e(n)|$. In other words, for two errors e_1 and e_2, the inequality $|e_1| < |e_2|$ implies that $\phi(e_1) < \phi(e_2)$.

Examples of commonly-employed cost functions that satisfy the above two characteristics include:

Mean-square-error:	$E\{e^2(n)\}$
Mean-absolute-error:	$E\{\mid e(n) \mid\}$
Mean-Kth-power-error:	$E\{\mid e(n) \mid^K\}$
Mean-normalized-squared-error:	$E\left\{ \dfrac{e^2(n)}{\sum_{j=n-L+1}^{n} x^2(j)} \right\}$
Least-squares error:	$\sum_{i=1}^{n} e_n^2(i); \quad e_n(i) = d(i) - \mathbf{W}^T(n)\mathbf{X}(i)$
Instantaneous squared error:	$e^2(n)$

The least-squares error criterion was considered extensively in Chapter 2 and will be discussed further in Chapter 5. The last error criterion listed above is an instantaneous approximation of the mean-square-error criterion. This approximation forms the basis of stochastic gradient adaptive filtering algorithms.

4.1.3 Implementation of the Steepest Descent Algorithm

To implement the steepest descent algorithm, we must first evaluate the partial derivatives of the cost function with respect to the coefficient values. Since derivatives and expectations are both linear operations, we can change the order in which the two operations are performed on the squared estimation error. With this change, we have

$$\begin{aligned}\frac{\partial E\{e^2(n)\}}{\partial \mathbf{W}(n)} &= E\left\{\frac{\partial e^2(n)}{\partial \mathbf{W}(n)}\right\} \\ &= E\left\{2e(n)\frac{\partial e(n)}{\partial \mathbf{W}(n)}\right\} \\ &= E\left\{2e(n)\frac{\partial (d(n)-\mathbf{W}^T(n)\mathbf{X}(n))}{\partial \mathbf{W}(n)}\right\} \\ &= -2E\{e(n)\mathbf{X}(n)\}. \end{aligned} \tag{4.7}$$

Thus, we can restate the steepest descent algorithm as

$$\mathbf{W}(n+1) = \mathbf{W}(n) + \mu E\{e(n)\mathbf{X}(n)\}. \tag{4.8}$$

To proceed further, we must evaluate the expectation in (4.8) directly. This expectation is

$$\begin{aligned} E\{e(n)\mathbf{X}(n)\} &= E\{\mathbf{X}(n)(d(n)-\widehat{d}(n))\} \\ &= E\{d(n)\mathbf{X}(n)\} - E\{\mathbf{X}(n)\mathbf{X}^T(n)\mathbf{W}(n)\} \\ &= \mathbf{P}_{d\mathbf{X}}(n) - \mathbf{R}_{\mathbf{XX}}(n)\mathbf{W}(n), \end{aligned} \tag{4.9}$$

where $\mathbf{P}_{d\mathbf{X}}(n) = E\{d(n)\mathbf{X}(n)\}$ is the cross-correlation vector of the desired response signal and the input vector at time n and $\mathbf{R}_{\mathbf{XX}}(n)$ is the autocorrelation matrix of the input vector. Thus, the steepest descent procedure for mean-square-error minimization can be written as

$$\mathbf{W}(n+1) = \mathbf{W}(n) + \mu(\mathbf{P}_{d\mathbf{X}}(n) - \mathbf{R}_{\mathbf{XX}}(n)\mathbf{W}(n)). \tag{4.10}$$

Table 4.1 shows a MATLAB function for implementing the steepest descent algorithm for a given autocorrelation matrix and cross-correlation vector.

Example 4.1: Behavior of the Steepest Descent Algorithm

Consider a two-coefficient system with autocorrelation matrix and cross-correlation vector given by

$$\mathbf{R}_{\mathbf{XX}}(n) = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix} \quad \text{and} \quad \mathbf{P}_{d\mathbf{X}}(n) = \begin{bmatrix} 1.5 \\ 1.5 \end{bmatrix},$$

Table 4.1: MATLAB function for performing the steepest descent search.

```
function [W] = steepdes(mu,W0,R,P,num_iter);

%  This function adapts a finite-impulse-response (FIR) filter using
%  the method of steepest descent.
%
%  Input parameters:
%      mu       = step size
%      W0       = Initial value of W(0) coefficients (L x 1)
%      R        = Input autocorrelation matrix (L x L)
%      P        = Cross-correlation vector (L x 1)
%      num_iter = number of iterations for simulation
%
%  Output of program:
%      W        = Evolution of coefficients (L x (num_iter + 1))

L = length(W0);
start_iter = 1;
end_iter = num_iter;

W = zeros(L,end_iter);
W(:,1:start_iter) = W0*ones(1,start_iter);

for n = start_iter:end_iter;

        W(:,n+1) = W(:,n) + mu*(P - R*W(:,n));

end;
```

respectively. These statistics correspond to a set of optimum MMSE coefficients given by

$$\mathbf{W}_{opt}(n) = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

The mean-squared error surface for this problem is plotted in Figure 4.5. We now investigate how the steepest descent algorithm behaves for different choices of the step size parameter μ and starting coefficient values $\mathbf{W}(0)$.

Figure 4.6 shows the evolution of the coefficients for a step size of $\mu = 0.01$ and three different starting vectors $\mathbf{W}(0)$. For each of the three adaptation curves, a single dot ($\bullet$) denotes one iteration of the algorithm. As can be seen from this graph, all the adaptation curves approach the optimum coefficient values $\mathbf{W}_{opt} = [1\ 1]^T$. For the two initial starting vectors that fall along the principal axes of the elliptical contours of the MSE surface, adaptation occurs along a straight line in the two-dimensional coefficient space. In contrast, when $\mathbf{W}(0) = [3.0\ 1.5]^T$, the coefficients take a curved path towards the bottom of the error surface.

Figure 4.7 shows the evolution of the coefficients for each of the initial starting vectors for $\mu = 0.1$. The behavior of the algorithm is similar to that shown in Figure 4.6, except that the spatial distances between successive values of $\mathbf{W}(n)$ are increased, indicating faster adaptation for this step size as compared to the previous case.

Figure 4.8 shows the behavior of the algorithm for a step size of $\mu = 1$. We have traced the coefficient paths for each of the different starting conditions using dashed lines in this figure. The larger dots in the figure indicate the coefficient values after individual iterations. The results of Figure 4.8 indicate that the behavior of the coefficients is more erratic for starting vectors of $\mathbf{W}(0) = [3.0\ 1.5]^T$ and $\mathbf{W}(0) = [-0.5\ -0.5]^T$, as the coefficients oscillate between the two sides of the error surface.

Figures 4.9 and 4.10 show the evolution of the coefficients $w_1(n)$ and $w_2(n)$, respectively, for different step sizes with an initial coefficient vector $\mathbf{W}(0) = [3\ \ 1.5]^T$. The x-axes on both plots are logarithmic in scale. We can see that a larger step size causes faster convergence of the coefficients to their optimum values. However, the behavior of the coefficient vector is more erratic for very large step sizes. We can also observe from each of the figures that the corrections made to the coefficient values are smaller when the coefficients are near the vicinity of their optimum values as compared to the changes made during the initial stages of adaptation. This characteristic is desirable for any adaptation algorithm, as it enables the coefficients to smoothly approach their optimum values.

We can see from Example 4.1 that the choice of step size is critical in obtaining good results with the steepest descent method. Too small a step size requires an excessive number of iterations to reach the vicinity of the minimum point on the error surface. Too large a step size causes the path to "bounce" from one side of the surface to the other, which can slow convergence as well. An excessively large step size will cause the next cost to be greater than the current cost, and the algorithm may diverge! Clearly, the success of the algorithm hinges on a good step size choice. Guidelines for selecting a good value for the step size μ can be determined through a performance analysis of the steepest descent algorithm.

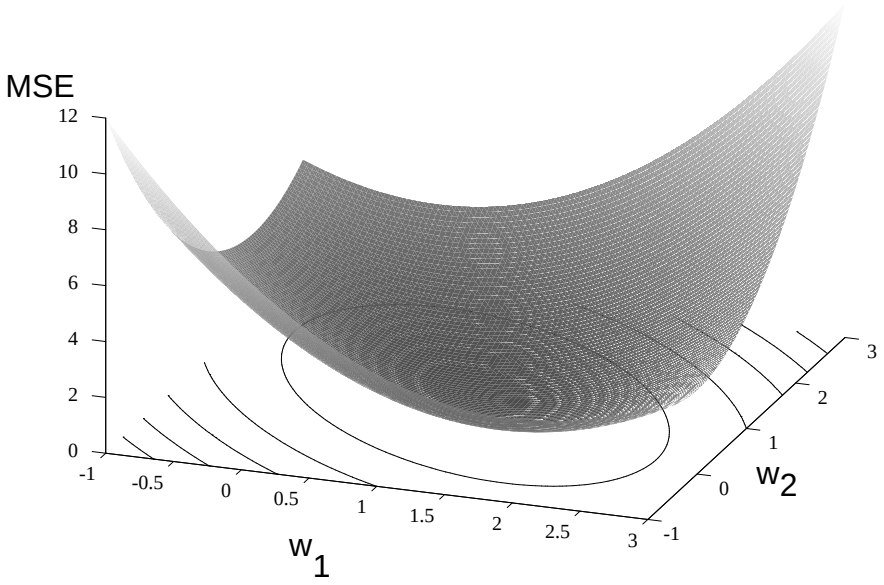

Figure 4.5: The mean-squared error surface for Example 4.1.

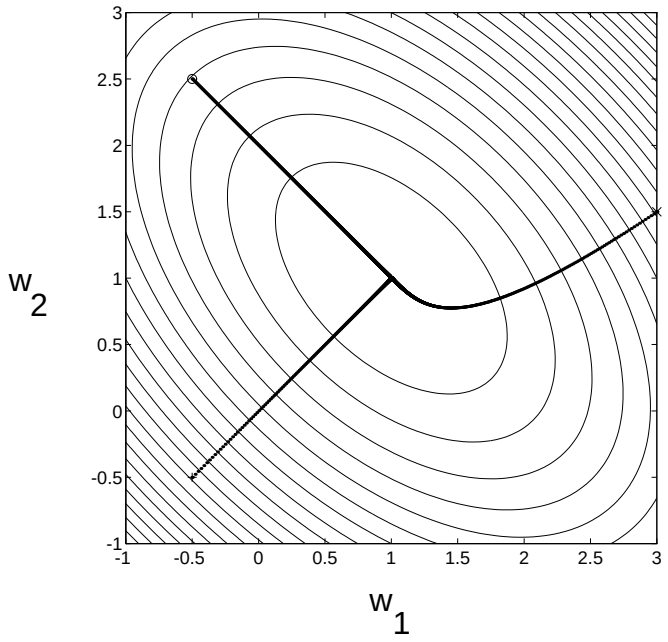

Figure 4.6: Evolution of the coefficients in Example 4.1 for different starting values of $\mathbf{W}(n)$ with $\mu = 0.01$.

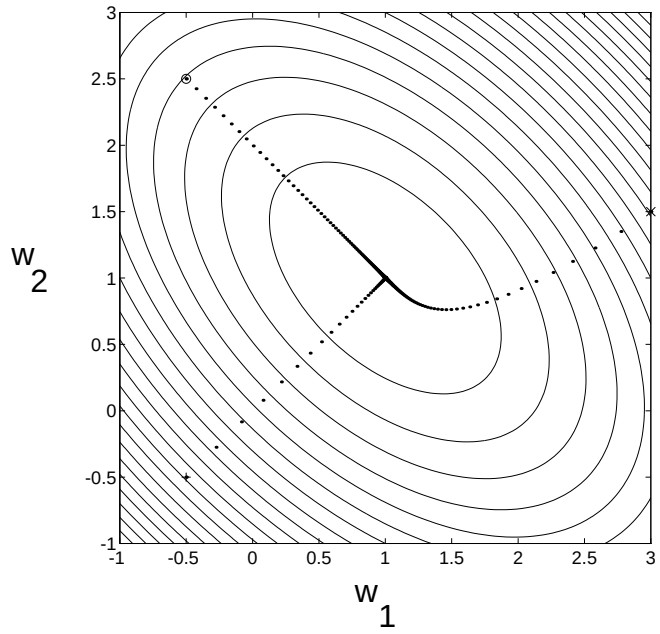

Figure 4.7: Evolution of the coefficients in Example 4.1 for different starting values of $\mathbf{W}(n)$ with $\mu = 0.1$.

Steady-State Properties of the Algorithm

Example 4.1 suggests that the steepest descent algorithm can converge to the minimum point on the error surface for a proper choice of step size. However, we have not yet proven that such convergence of the steepest descent algorithm will occur in general. To pursue this issue further, assume that the autocorrelation matrix and cross correlation vector are constant over time, such that $\mathbf{R_{xx}}(n) = \mathbf{R_{xx}}$ and $\mathbf{P}_{\mathrm{d}\mathbf{x}}(n) = \mathbf{P}_{\mathrm{d}\mathbf{x}}$. We ask the question: *what coefficient values* $\mathbf{W}(n)$ *are not changed by the steepest descent update?* Let $\mathbf{W}_{ss}$ be such a value of the coefficient vector. We can write the steepest descent update for this special value of $\mathbf{W}(n)$ as

$$\begin{aligned} \mathbf{W}(n+1) &= \mathbf{W}(n) + \mu(\mathbf{P}_{\mathrm{d}\mathbf{x}} - \mathbf{R_{xx}}\mathbf{W}(n)) \\ &= \mathbf{W}(n) = \mathbf{W}_{ss}. \end{aligned} \tag{4.11}$$

The algorithm applies no correction to the coefficient vector in this situation, indicating that the system has converged to a stationary point. Equations (4.8) and (4.11) imply that

$$E\{e(n)\mathbf{X}(n)\} = 0. \tag{4.12}$$

at the stationary point of the system.

The above condition is the same as the orthogonality principle described in Chapter 2. This result implies that if the steepest descent algorithm converges, then the coefficient

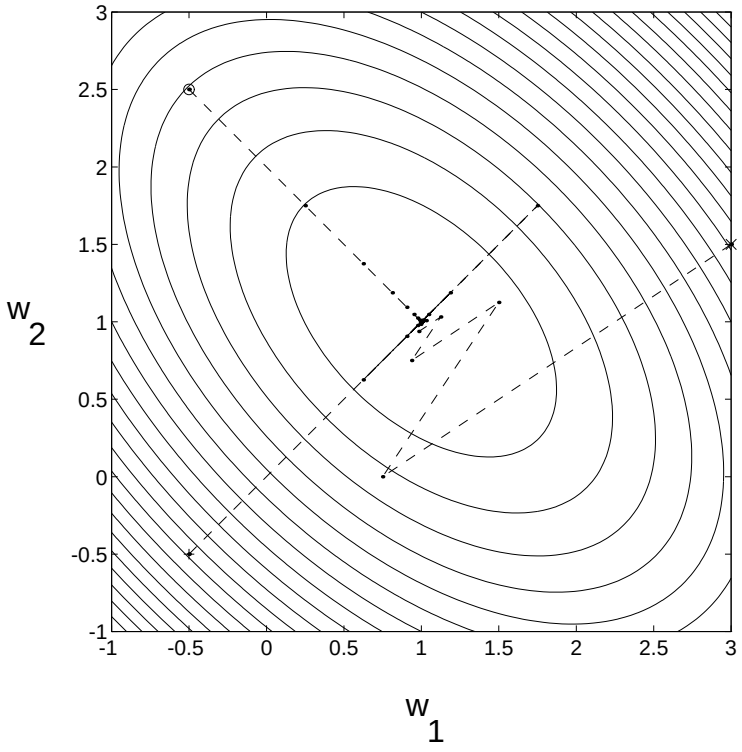

Figure 4.8: Evolution of the coefficients in Example 4.1 for different starting values of $\mathbf{W}(n)$ for $\mu = 1$.

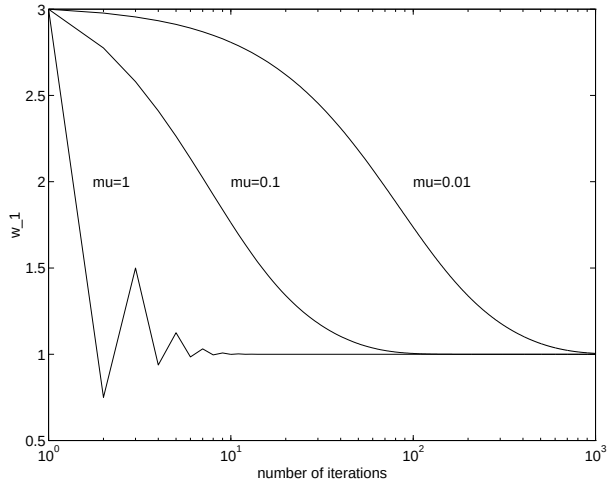

Figure 4.9: Evolution of $w_1(n)$ for different step sizes in Example 4.1.

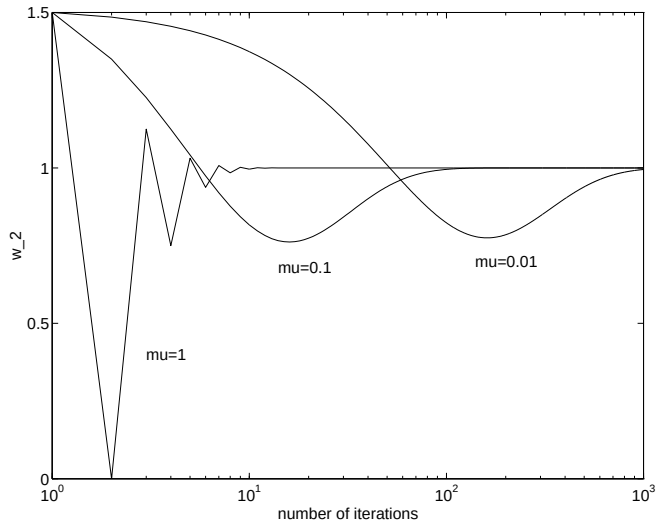

Figure 4.10: Evolution of $w_2(n)$ for different step sizes in Example 4.1.

values at convergence correspond to the optimal solution to the minimum mean-square-error estimation problem! The steepest descent procedure can potentially be used to find this optimal solution iteratively. As further evidence of this fact, we can determine the value of $\mathbf{W}(n) = \mathbf{W}_{ss}$ at the stationary point of the iteration by solving the L equations defined by (4.11) to get

$$\mathbf{R}_{\mathbf{xx}}\mathbf{W}_{ss} \;=\; \mathbf{P}_{\mathrm{d}\mathbf{x}}. \tag{4.13}$$

We can solve for $\mathbf{W}_{ss}$ if the inverse of the autocorrelation matrix exists. The steady-state solution in this case is

$$\mathbf{W}_{ss} \;=\; \mathbf{R}_{\mathbf{xx}}^{-1}\mathbf{P}_{\mathrm{d}\mathbf{x}} \;=\; \mathbf{W}_{opt}, \tag{4.14}$$

which is simply the optimal solution for the MMSE estimation problem.

The solution in (4.14) is *unique* whenever $\mathbf{R}_{\mathbf{xx}}^{-1}$ exists. In other words, there exists only one possible stationary point for the iteration, and it corresponds to the optimum MMSE solution for the problem. The value of the mean-squared error at this stationary point corresponds to the minimum mean-squared error value for this problem and can be evaluated using (2.50) as

$$\begin{aligned} E\{e^2(n)|\mathbf{W}(n) = \mathbf{W}_{opt}\} &= \sigma_d^2 - \mathbf{P}_{\mathrm{d}\mathbf{x}}^T\mathbf{R}_{\mathbf{xx}}^{-1}\mathbf{P}_{\mathrm{d}\mathbf{x}} \\ &= \sigma_d^2 - \mathbf{P}_{\mathrm{d}\mathbf{x}}^T\mathbf{W}_{opt}. \end{aligned} \tag{4.15}$$

Convergence of the Steepest Descent Method

Given that the stationary point of the steepest descent algorithm is the optimum MMSE solution, a second, equally-important consideration is whether the algorithm converges at all. We now explore the conditions on the step size to guarantee convergence for a single coefficient system. The results that we derive are similar in flavor to more complete results that we will derive in Chapter 4 for data-driven approximate versions of the steepest descent method.

For a single coefficient system with $L = 1$, the evolution equation in (4.10) is given by

$$\begin{aligned} w(n+1) &= w(n) + \mu(p_{dx} - r_{xx}(0)w(n)) \\ &= (1 - \mu r_{xx}(0))w(n) + \mu p_{dx}, \end{aligned} \tag{4.16}$$

where $p_{dx} = E\{d(n)x(n)\}$ and $r_{xx}(0) = E\{x^2(n)\}$. This equation is simply a first-order scalar difference equation in the coefficient $w(n)$. In fact, the coefficient sequence $w(n+1)$ is exactly the same as the output $y(n)$ of a linear, time-invariant digital filter defined by the equation

$$y(n) = ay(n-1) + \zeta(n), \tag{4.17}$$

where $a = 1 - \mu r_{xx}(0)$, $y(-1) = w(0)$, and the input signal is given by $\zeta(n) = \mu p_{dx} u(n)$, where $u(n)$ is the discrete-time step function. From the theory of digital filters, we know that the stability of a causal, linear, time-invariant discrete-time filter in (4.17) is controlled by the constant a. For $|a| < 1$, the digital filter of (4.17) is stable; *i.e.*, the sequence $y(n)$ is finite-valued as n tends toward infinity. Using this relationship, we find that the steepest descent method is stable if and only if

$$-1 \; < \; (\mu r_{xx}(0) - 1) \; < \; 1. \tag{4.18}$$

Adding one to both sides of the above inequalities and dividing all quantities by $r_{xx}(0)$, we find that the conditions given by

$$0 < \mu < \frac{2}{r_{xx}(0)} \tag{4.19}$$

guarantee the convergence of the steepest descent method for a single-coefficient system. Note that $r_{xx}(0)$ is also the power in the input signal, a quantity that can be easily estimated using signal measurements.

We can also show that the coefficient of the steepest descent method converges to its optimal value $w_{opt} = p_{dx}/r_{xx}(0)$ when the system is stable. To see this, let us subtract w_{opt} from both sides of (4.16). After substituting $p_{dx} = r_{xx}(0)w_{opt}$ in the resulting equation, we get

$$[w(n+1) - w_{opt}] = (1 - \mu r_{xx}(0))[w(n) - w_{opt}]. \tag{4.20}$$

It is easy to see from the above equation that if $0 < \mu < 2/r_{xx}(0)$, the coefficient error $w(n) - w_{opt}$ decreases exponentially to zero as the number of iterations n increases.

These results indicate three important facts concerning the stability of the steepest descent method:

- *For stable operation, the step size must be positive.* This result is intuitively pleasing, as a negative step size would cause the coefficients to move *up* the mean-square-error surface.

- *The range of stable step sizes decreases as the input signal power increases.* This fact also makes sense, as the input data power is directly related to the curvature of the mean-square-error surface used by the steepest descent method. If the curvature of the error surface is too great, the oscillatory behavior observed in previous examples becomes more likely as the step size is increased.

- *When the system operates in a stable manner, the coefficient converges to its optimal value in stationary environments.* This fact is also essential if an adaptive filter is to be useful in practice.

This single coefficient example does not illustrate the dependence of the step size bounds on the filter length L. We defer such a discussion to the next section.

4.2 Stochastic Gradient Adaptive Filters

The method of steepest descent can be used to find the optimum minimum mean-square-error estimate of $\mathbf{W}(n)$ in an iterative fashion. However, this procedure uses the *statistics* of the input and desired response signals and not on the actual measured signals. In practice, the input signal statistics are not known *a priori.* Moreover, if these statistics were known and if the autocorrelation matrix $\mathbf{R_{xx}}(n)$ were invertible, we could find the optimum solution given in (4.14) directly in one step! Thus, the method of steepest descent, as described in the previous section, is not useful as an estimation procedure on its own in most practical situations. We now describe a simple approximation that yields a practical and efficient variation of the steepest descent algorithm.

The Instantaneous Gradient

We can see from (4.8) that the method of steepest descent depends on the input data and desired response signal statistics through the expectation operation that is performed on the product $-e(n)\mathbf{X}(n)$. This product is the *gradient* of the squared error function $(e^2(n))/2$ with respect to the coefficient vector $\mathbf{W}(n)$. We can consider the vector $-e(n)\mathbf{X}(n)$ as an approximation of the true gradient of the mean-squared error estimation surface. This approximation is known as the *instantaneous gradient* of the mean-squared error surface.

Our approach to developing a useful and realizable adaptive algorithm is to replace the gradient vector $-E\{e(n)\mathbf{X}(n)\}$ in the steepest descent update in (4.8) by its instantaneous approximation $-e(n)\mathbf{X}(n)$. Adaptive filters that are based on the instantaneous gradient approximation are known as *stochastic gradient* adaptive filters.

4.2.1 The Least-Mean-Square Algorithm

We get the following strategy for updating the coefficients by using the instantaneous gradient approximation in the steepest descent algorithm:

$$\mathbf{W}(n+1) \;=\; \mathbf{W}(n) + \mu e(n)\mathbf{X}(n), \tag{4.21}$$

where the error $e(n)$ is given by

$$e(n) \;=\; d(n) - \mathbf{W}^T(n)\mathbf{X}(n). \tag{4.22}$$

The coefficient vector $\mathbf{W}(n)$ may be initialized arbitrarily and is typically chosen to be the zero vector. The only difference between the procedure of (4.21) and (4.22), and the steepest descent procedure of (4.8) is that we have removed the expectation operator $E\{\cdot\}$ from the gradient estimate. The above algorithm has become known as the *Least-Mean-Square (LMS) adaptive filter,* a name coined by its originators [Widrow 1960]. Because of its simplicity and properties, it is the most widely-used adaptive filter today. Table 4.2 lists a MATLAB function that implements the LMS adaptive filter.

REMARK 4.1: Substituting $e(n)\mathbf{X}(n)$ for $E\{e(n)\mathbf{X}(n)\}$ is a crude approximation for the gradient of the mean-square-error surface. However, the value of $e(n)\mathbf{X}(n)$ points in the same direction as the true gradient on average. In other words, the instantaneous gradient is an *unbiased estimate* of the true gradient. Since the step size parameter μ is chosen to be a small value, any errors introduced by the instantaneous gradient are averaged over several iterations, and thus the performance loss incurred by this approximation is relatively small.

4.2.2 General Stochastic Gradient Adaptive Filters

Recall from our discussion of the steepest descent algorithm that the choice of cost function $J(n) = E\{e^2(n)\}$ was an arbitrary one and that other cost functions can provide adequate error surfaces for a gradient search. Some alternative cost functions were discussed in Section 4.1.2. We now consider a particular class of cost functions of the form

$$J(n) \;=\; E\{g(e(n))\}, \tag{4.23}$$

where $g(e(n))$ is an even function of $e(n)$. We can develop a family of steepest descent procedures that attempt to minimize the cost function in (4.23) using (4.5). The coefficient

Table 4.2: MATLAB function for applying the FIR LMS adaptive filter.

```
function [W,dhat,e] = fir_lms(mu,W0,x,d);

%  This function adapts a finite-impulse-response (FIR)
%  filter using the least-mean-square (LMS) adaptive
%  algorithm.
%
%  Input parameters:
%      mu       = step size
%      W0       = Initial value of W(0) coefficients (L x 1)
%      x        = input data signal (num_iter x 1)
%      d        = desired response signal (num_iter x 1)
%
%  Output of program:
%      W        = Evolution of coefficients (L x (num_iter + 1))
%      dhat    = output of adaptive filter (num_iter x 1)
%      e        = error of adaptive filter (num_iter x 1)

L = length(W0);
start_iter = 1;
end_iter = min([length(x) length(d)]);

W = zeros(L,end_iter);
dhat = zeros(end_iter,1);
e = zeros(end_iter,1);

W(:,1:start_iter) = W0*ones(1,start_iter);
X = zeros(L,1);

for n = start_iter:end_iter;

        X(2:L) = X(1:L-1);
        X(1) = x(n);
        dhat(n) = X'*W(:,n);
        e(n) = d(n) - dhat(n);

        W(:,n+1) = W(:,n) + mu*e(n)*X;

end;
```

vector update is given by

$$\begin{aligned}\mathbf{W}(n+1) &= \mathbf{W}(n) - \alpha \frac{\partial E\{g(e(n))\}}{\partial \mathbf{W}(n)} \\ &= \mathbf{W}(n) + \alpha E\{f(e(n))\mathbf{X}(n)\}, \end{aligned} \tag{4.24}$$

where we define $f(e)$ to be

$$f(e) = \frac{dg(e)}{de}. \tag{4.25}$$

We can use the instantaneous gradient approximation to provide realizable adaptive filters of the form

$$\mathbf{W}(n+1) = \mathbf{W}(n) + \alpha f(e(n))\mathbf{X}(n). \tag{4.26}$$

The only difference of this general form of the stochastic gradient adaptive filter from the LMS adaptive filter is the use of the nonlinearity $f(\cdot)$ on the error $e(n)$ in the update.

From the constraints on $\phi(e)$ presented in Section 4.1.2, we see that $g(e)$ is an even function that monotonically increases with $|e|$. Consequently, the nonlinearity $f(\cdot)$ is an odd function that preserves the polarity of $e(n)$; *i.e.,*

$$\text{sgn}(f(e(n))) = \text{sgn}(e(n)), \tag{4.27}$$

where the $\text{sgn}(\cdot)$ operation is defined to be

$$\text{sgn}(e) = \begin{cases} 1 & e > 0 \\ 0 & e = 0 \\ -1 & e < 0. \end{cases} \tag{4.28}$$

We can derive many useful stochastic gradient adaptive filters from the general structure given in (4.26) using different functions $g(e)$. We now describe several such adaptive filters.

The Sign-Error Adaptive Filter

Consider the mean-absolute-error cost function $J(n) = E\{|e(n)|\}$. Since the derivative of $|e(n)|$ with respect to the error is

$$f(e(n)) = \text{sgn}(e(n)), \tag{4.29}$$

we obtain the following stochastic gradient adaptive filter using (4.26):

$$\mathbf{W}(n+1) = \mathbf{W}(n) + \mu \text{sgn}(e(n))\mathbf{X}(n), \tag{4.30}$$

where we have defined $\mu = \alpha$ for convenience. The coefficient vector update for this adaptive filter is known as the *sign-error algorithm*, or simply the sign algorithm, as it uses the sign of the error in the gradient update. Although we derived this algorithm from a gradient descent argument, it is interesting to note that it has in the past been interpreted as a "simplified LMS update" algorithm, where the sign operation allows a simpler multiplier structure in dedicated signal processing hardware [Duttweiler 1981].

The Least-Mean-Kth-Power Adaptive Filter

We can generalize the mean-square-error and mean-absolute-error cost functions in a natural way by defining this cost function as

$$J(n) \quad = \quad E\{|e(n)|^K\}, \tag{4.31}$$

where K is a positive integer. Following a similar development as before and noting that $d|e|^K/de = K|e|^{K-1}\text{sgn}(e)$, we arrive at the following *least-mean-Kth-power adaptive filter*:

$$\mathbf{W}(n+1) \quad = \quad \mathbf{W}(n) + \mu|e(n)|^{K-1}\text{sgn}(e(n))\mathbf{X}(n), \tag{4.32}$$

where we have defined $\mu = K\alpha$ for convenience. It has been shown that this algorithm can achieve better performance than the LMS adaptive filter by adjusting the integer-valued parameter K for certain signal and noise statistics [Walach 1984].

Quantized-Error Algorithms

Consider a *piecewise-linear* cost function $g(e)$ shown in Figure 4.11a. We can derive a stochastic gradient adaptive filter for which the nonlinearity $f(e)$ is as shown in Figure 4.11b. This nonlinearity represents a *quantizer*, since values of e in different ranges are mapped to specific constants. In a digital computer, quantization of signals is necessary for implementing algorithms in general. In dedicated VLSI hardware, however, it may be necessary to quantize certain signals to a fewer number of bits, in order to allow a reasonable multiplier structure. Thus, we are motivated to study the performance of these *quantized stochastic gradient adaptive filters* to see how they behave relative to floating-point versions that suffer from the effects of quantization to a much lesser degree.

Quantized error algorithms can also be designed to provide larger than normal coefficient changes when the estimation errors are large in magnitude and smaller changes when the estimation errors are smaller in magnitude. Such algorithms include as special cases

- the sign-error algorithm in (4.30);
- the *dual-sign algorithm,* where $f(e(n))$ is given by

$$f(e(n)) \quad = \quad \begin{cases} K\text{sgn}(e(n)) & \text{if } |e(n)| \geq t_0 \\ \text{sgn}(e(n)) & \text{if } |e(n)| < t_0, \end{cases} \tag{4.33}$$

  where K and t_0 are parameters of the nonlinearity [Kwong 1986]; and
- the *power-of-two quantized algorithm,* where $f(e(n))$ is given by

$$f(e(n)) \quad = \quad \begin{cases} 2^{\lceil \log_2(|e(n)|) \rceil}\text{sgn}(e(n)), & \text{if } |e(n)| < 1 \\ \text{sgn}(e(n)) & \text{if } |e(n)| \geq 1, \end{cases} \tag{4.34}$$

  where $\lceil \cdot \rceil$ denotes the next largest integer value [Ping1986].

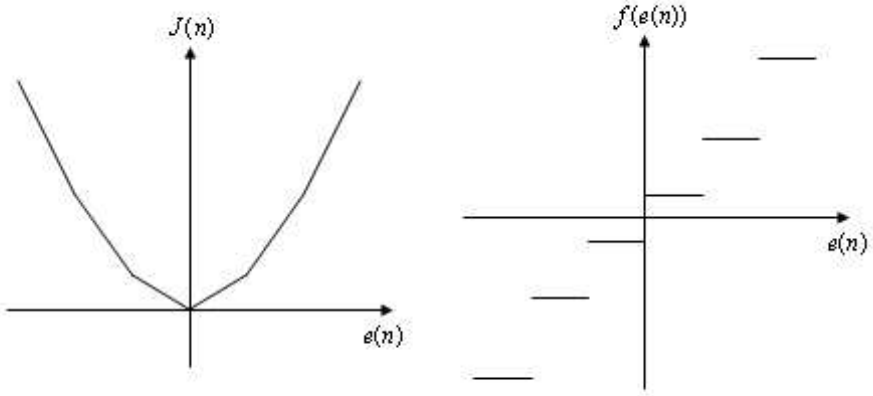

Figure 4.11: a) A piecewise-linear cost function. b) The resulting quantizer nonlinearity.

Block LMS Algorithm

Consider the following error criterion that is based on a finite sum-of-squared errors:

$$E\left\{\frac{1}{N}\sum_{i=n-N+1}^{n}(e_n(i))^2\right\} = \frac{1}{N}\sum_{i=n-N+1}^{n}E\{(d(i)-\mathbf{W}^T(n)\mathbf{X}(i))^2\}, \tag{4.35}$$

where the subscript on the error $e_n(i)$ explicitly indicates that its calculation depends on the coefficients at time n. Moreover, since $\mathbf{W}(n)$ is used for N consecutive time samples, we need to consider updating the coefficients only once every N samples. Using the instantaneous approximation to the gradient of this cost function as in (4.26), we arrive at the following *block LMS adaptive filter:*

$$\mathbf{W}(n+N) = \mathbf{W}(n)+\frac{\mu}{N}\sum_{i=n-N+1}^{n}e_n(i)\mathbf{X}(i). \tag{4.36}$$

This update uses an *average* of a set of consecutive instantaneous gradients to adjust the coefficients of the filter in one step. This averaging results in a more accurate estimate of the gradient of the mean-squared error surface as the block length is increased. However, the adaptive filter coefficients are updated less frequently, and this may result in a slower speed of adaptation.

At first glance, the block LMS algorithm looks more complicated than the LMS algorithm because of the summation of the consecutive gradient terms. However, since the coefficients of the filter are fixed over the block, efficient convolution techniques employing the fast Fourier transform (FFT) algorithms can be used to implement the filtering operation. Moreover,

FFT-based techniques can also be used to implement the gradient summation, leading to significant savings in multiplications for long block lengths [Clark 1981].

We will discuss the performance and behavior of many of the stochastic gradient adaptive filters discussed above in the following chapters.

4.2.3 Examples of LMS Adaptive Filters

By far the most popular adaptive filter, the LMS adaptive filter has been studied extensively by many in the signal processing community. We conclude this chapter with several simulation examples to illustrate the LMS adaptive filter's behavior.

Example 4.2: Stationary System Identificationm

This example considers the identification of the system in Example 4.1 using measurements of its input and output signals. For this system, we generated a correlated input data sequence using the single-pole IIR digital filter whose input-output relationship is given by

$$x(n) = ax(n-1) + b\xi(n),$$

where $\xi(n)$ is an i.i.d., zero-mean, unit-variance Gaussian sequence and a and b have been chosen as

$$\begin{aligned} a &= 0.5 \\ b &= \frac{\sqrt{3}}{2}. \end{aligned}$$

The desired response signal was generated using the following FIR model:

$$d(n) = x(n) + x(n-1) + \eta(n),$$

where $\eta(n)$ is an i.i.d. zero-mean Gaussian sequence with variance $\sigma_\eta^2 = 0.01$. The statistics of this problem match those in Example 4.1, allowing us to compare the results of the LMS adaptation with those produced by the steepest descent algorithm.

Figure 4.12 shows the evolution of the coefficients for 1000 iterations of both the steepest descent and the LMS adaptive filter superimposed on the MSE surface for a step size of $\mu = 0.01$. The coefficient vector was initialized as $\mathbf{W}(0) = [3 \;\; 1.5]^T$. Each dot on the solid-line curve indicates one iteration of the LMS algorithm, and the solid line is an ensemble average of one hundred different runs of the LMS adaptive filter over independent data sets with identical individual statistics. The dashed line on the plot corresponds to the path of the coefficients adapted using the steepest descent method. The same information is plotted as a function of time in Figure 4.13. The evolutions of the LMS adaptive filter coefficients, both as individual and ensemble averages of the convergence paths, closely follow the path produced by the steepest descent algorithm. However, the behavior of the coefficients of the LMS adaptive filter is more "noisy" for each individual run. The coefficients of both systems approach the optimum filter coefficient values in this example.

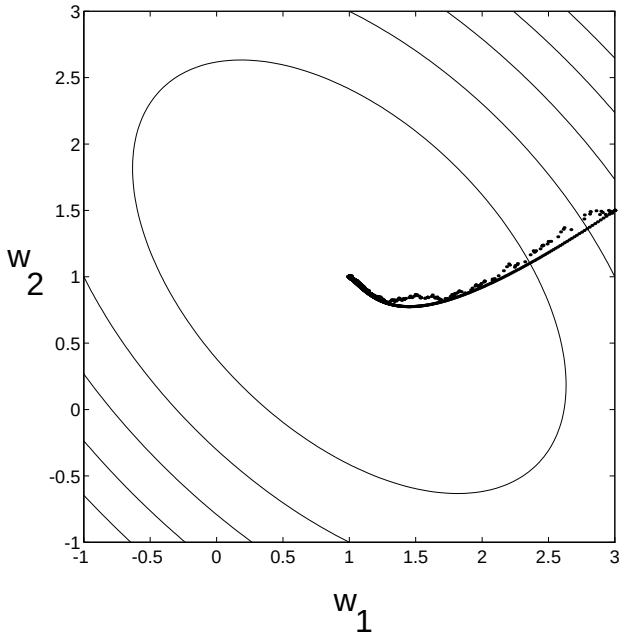

Figure 4.12: Evolution of the coefficients of the LMS (dotted curve), ensemble-averaged LMS (solid curve), and steepest descent (dashed curve) algorithms in Example 4.2 for $\mu = 0.01$.

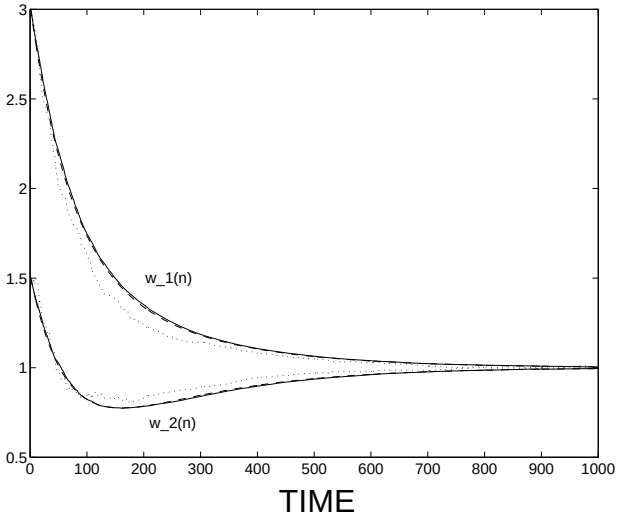

Figure 4.13: Evolution of the coefficients of the LMS (dotted curve), ensemble-averaged LMS (solid curve), and steepest descent (dashed curve) algorithms in Example 4.2 for $\mu = 0.01$.

Figure 4.14 displays the evolution of the error signal $e(n)$ for the LMS adaptive filter. Starting from large initial values, the errors decrease to smaller values as time progresses. The error never goes to zero because of the random noise $\eta(n)$ that perturbs our measurements $d(n)$ of the system.

We now investigate the behavior of the LMS algorithm for a larger step size $\mu = 0.1$. Figures 4.15 and 4.16 show the behaviors of the coefficients for this case. We can see that the evolution of the LMS adaptive filter coefficients follows the general path of those adapted using the steepest descent algorithm. However, the behavior of the LMS adaptive filter coefficients is considerably more erratic for this larger step size. As we might expect, the coefficients approach their optimum values much faster for this larger step size.

Figure 4.18 and 4.17 show the evolutions of the absolute value of the first adaptive filter coefficient and the squared value of the estimation error for a single experiment of the LMS adaptive filter operating with a step size of $\mu = 1$ in this case. Clearly, the evolution of the system is erratic, with large variations in both the magnitudes of the filter coefficients and the estimation error. Since the steepest descent procedure converges in this case as observed in Example 4.1, we infer that the behaviors of the LMS and steepest descent adaptation procedures are quite different for large step sizes. The reasons for these differences are explored in the next chapter.

Example 4.3: Nonstationary Channel Equalization

We now consider an example drawn from digital communications, in which an adaptive filter is used to compensate for the non-ideal characteristics of a communications channel. Figure 4.19 shows the block diagram of the system, in which a message is encoded in the form of a digital bit stream before it is modulated and transmitted over a channel. At the receiver, the signal is sampled and then processed to retrieve the original message. For this example, we model the encoding, transmission, and decoding of the signal as a time-varying linear filter whose output is corrupted by noise. The task of the adaptive filter is to recover the original bits transmitted by developing an approximate inverse of the channel. This process is known as *equalization*. Because the properties of the channel are typically unknown or changing over time, an adaptive filter is used to approximate the inverse of this system. To initially adapt the filter, a known series of bits are transmitted over the channel, and the adaptive filter is trained using a delayed version of this known sequence, where the sample delay Δ is chosen for best performance. Then, a *decision-directed* technique can be used to maintain the proper equalization of the channel.

For our example, we assume that the noise is negligible and that the channel can be modeled using the first-order difference equation given by

$$x(n) \;=\; a(n)x(n-1) + s(n),$$

where $s(n)$ are the bits transmitted and $a(n)$ is a time-varying coefficient. The bit sequence $s(n)$ is an i.i.d. binary sequence where

$$\Pr(x(n) = 1) \;=\; \Pr(x(n) = -1) \;=\; 0.5.$$

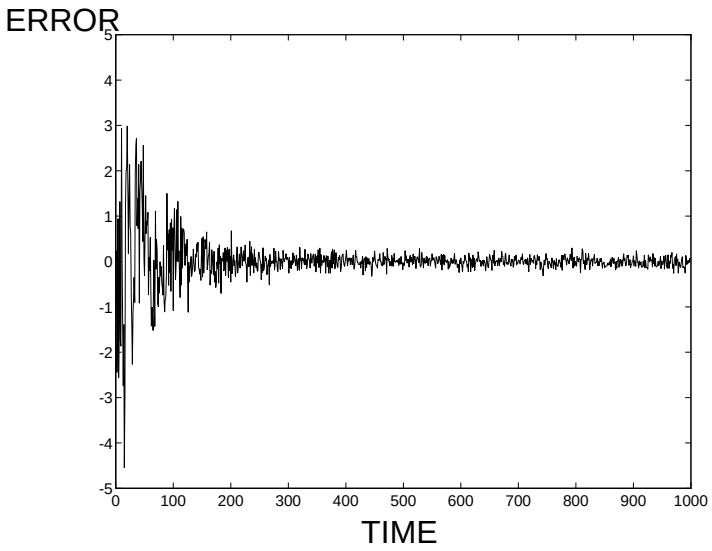

Figure 4.14: Evolution of error $e(n)$ in Example 4.2 for $\mu = 0.01$.

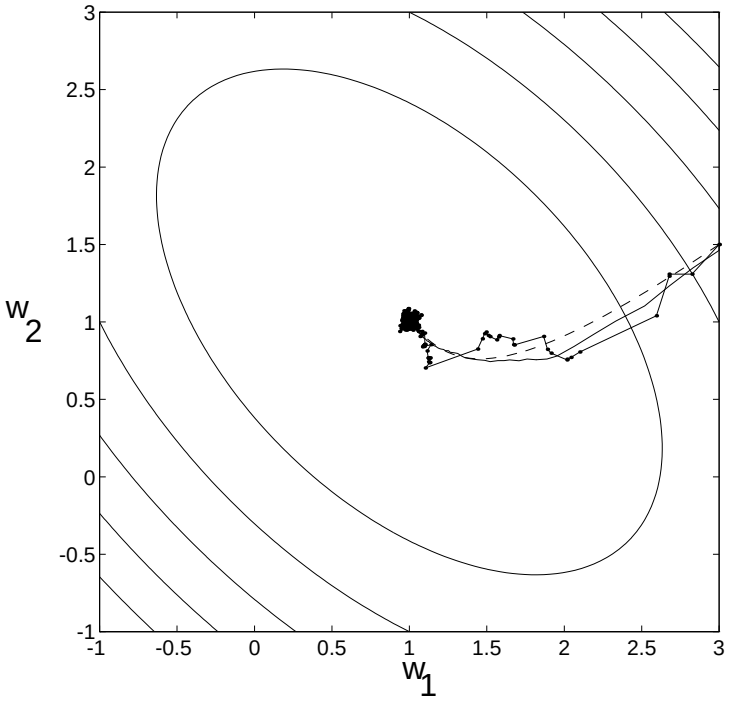

Figure 4.15: Evolution of the coefficients of the LMS (dotted curve), ensemble-averaged LMS (solid curve), and steepest descent (dashed curve) algorithms in Example 4.2 for $\mu = 0.1$.

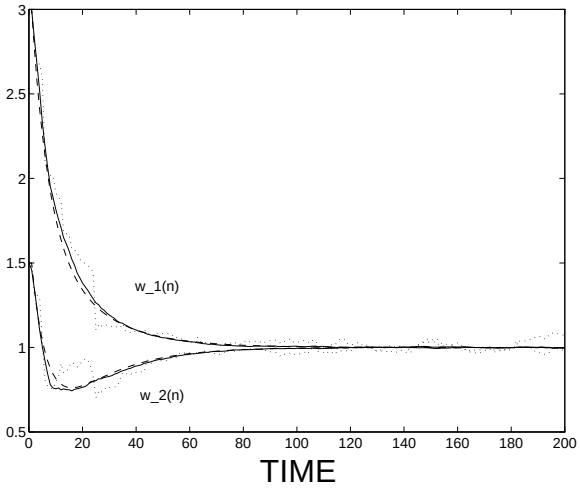

Figure 4.16: Evolution of the coefficients of the LMS (dotted curve), ensemble-averaged LMS (solid curve), and steepest descent (dashed curve) algorithms in Example 4.2 for $\mu = 0.1$.

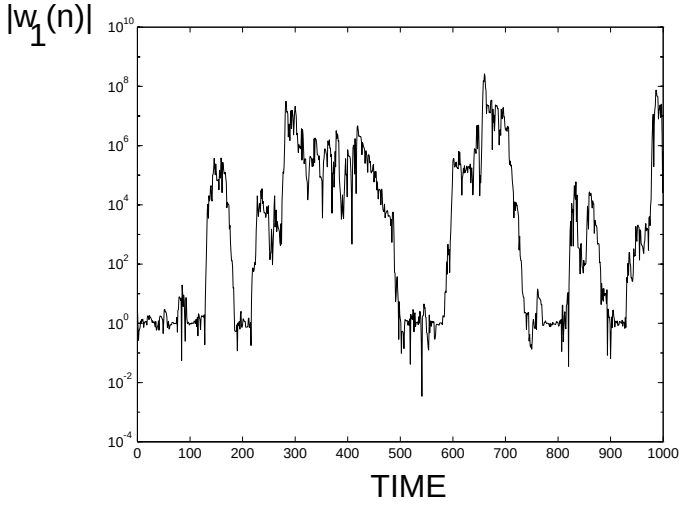

Figure 4.17: Evolution of the absolute value of the first coefficient of the LMS adaptive filter in Example 4.2 for $\mu = 1$.

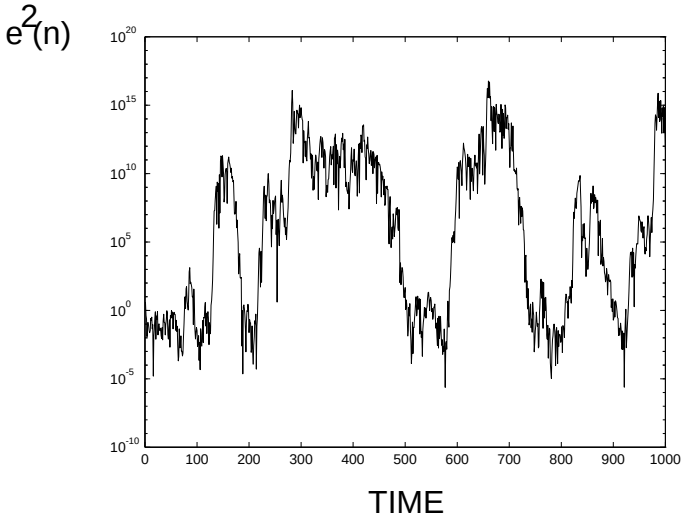

Figure 4.18: Evolution of the squared error $e^2(n)$ in Example 4.2 for $\mu = 1$.

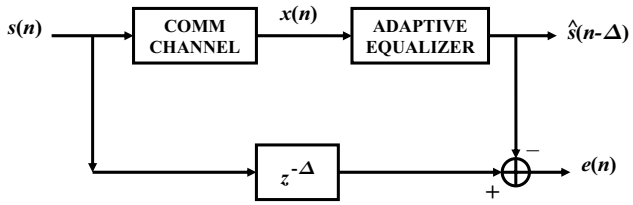

Figure 4.19: Block diagram of an adaptive equalizer used in digital communication systems.

The actual behavior of the coefficient $\{a(n)\}$ is

$$a(n) = \begin{cases} 0 & 0 \leq n \leq 100 \\ \dfrac{9(n-100)}{2000} & 101 \leq n \leq 300 \\ 0.9 & 301 \leq n \leq 600. \end{cases}$$

Thus, the coefficient $a(n)$ undergoes a linear change from $a(100) = 0$ to $a(300) = 0.9$.

The inverse system for the channel in the absence of any noise is described by the relationship

$$s(n) = x(n) - a(n)x(n-1).$$

Consequently, we can use a two-coefficient adaptive filter whose input signal is $x(n)$ and whose desired response signal is $s(n-\Delta) = s(n)$ to equalize the received signal for the effects of the channel. The optimal coefficient vector is given by

$$\mathbf{W}_{opt}(n) = \begin{bmatrix} 1 \\ -a(n) \end{bmatrix}.$$

The adaptive filter coefficients were initialized to their optimum values $\mathbf{W}(0) = [1\ 0]$ in this example in order to observe the tracking behavior of the system.

Figure 4.20 shows the evolution of the filter coefficients $w_1(n)$ and $w_2(n)$ for a step size of $\mu = 0.1$. The adaptive filter coefficients track their optimum values as the system function changes with a lag from the true coefficient values. This *lag error* is in general greater for smaller step sizes due to the decreased speed of adaptation for smaller step sizes. We can also see that, even though the optimum value of the first coefficient does not change, the value of $w_1(n)$ produced by the adaptive filter changes. This effect is due to the coupled nature of the coefficient adaptation. Figure 4.21 shows the behavior of the same system for $\mu = 0.01$, in which case the lag error in the coefficients is much greater.

Example 4.4: Adaptive Line Enhancement

In Example 2.13 of Chapter 2, we considered the task of line enhancement, whereby a sinusoidal signal is recovered from a noisy version of the sinusoid using a one-step linear predictor. Figure 4.22 shows the block diagram of the adaptive system. In this example, we employ the LMS algorithm to find the coefficients of the filter. For this example, we choose the signals to be the same as those for Example 2.13, so that we can compare the adaptive filter's output with that of the optimum MMSE fixed-coefficient line enhancer.

Figure 4.23 plots the difference between the output $\widehat{d}(n)$ of the LMS adaptive line enhancer and the output of the optimum MMSE line enhancer, given by $\widehat{d}_o(n) = \mathbf{W}_{opt}^T(n)\mathbf{X}(n)$ for a step size of $\mu = 0.0001$. Initial convergence of the system occurs over the first 5000 samples. Figure 4.24 shows the spectra of the input signal as well as the enhanced signals as obtained from the optimum MMSE estimator and from the adaptive LMS line enhancer for the sequence of values from $5001 \leq$

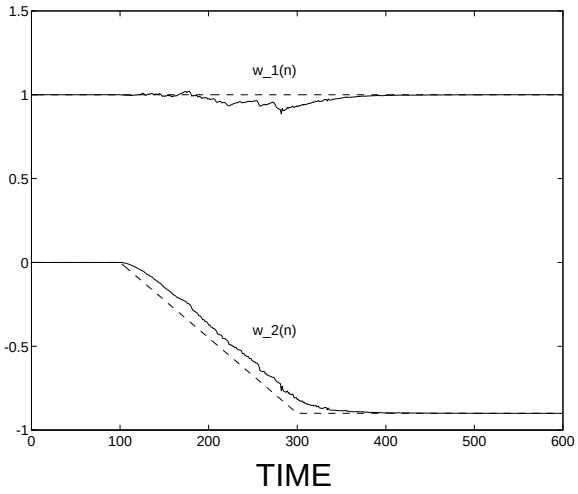

Figure 4.20: Tracking of optimal coefficients in Example 4.3 for $\mu = 0.1$.

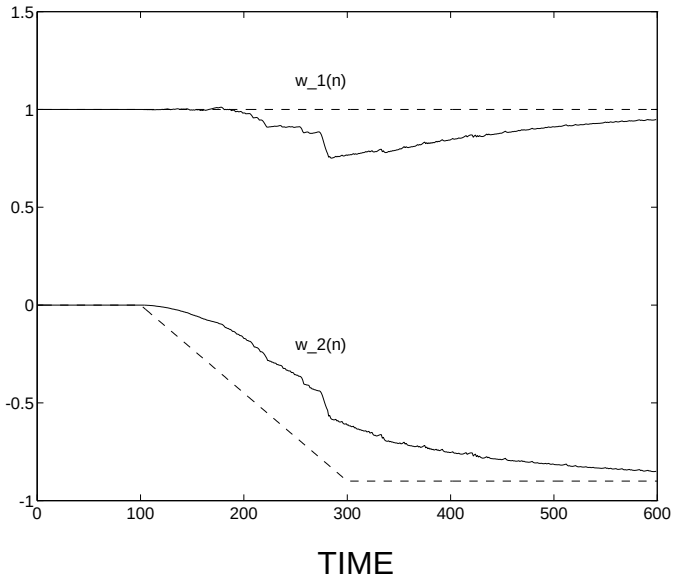

Figure 4.21: Tracking of optimal coefficients in Example 4.3 for $\mu = 0.01$.

$n < 10000$. As can be seen, the adaptive line enhancer's performance closely follows that of the fixed system. In verification of this fact, Figure 4.25 shows the output signals of both the optimum MMSE and adaptive LMS line enhancers after convergence, along with the original uncorrupted sinusoid. Clearly, both line enhancers perform nearly as well, indicating that the LMS adaptive line enhancer can achieve similar performance as the optimum MMSE line enhancer after a sufficient number of iterations.

4.3 Main Points of This Chapter

- The method of steepest descent is an iterative procedure for finding the minimum point of a smooth error surface.
- When searching the MMSE surface for an FIR system model, the method of steepest descent converges to the optimum MMSE solution for adequately small step sizes.
- Convergence of the method of steepest descent is controlled by the autocorrelation statistics of the input signal, the cross-correlation between the input and desired response signals, and the step size. Too large a step size can cause divergence of the algorithm.
- Stochastic gradient adaptive algorithms are approximate implementations of steepest descent procedures in which an instantaneous estimate of the cost function $\phi(e(n))$ is used in place of the expected value $E\{\phi(e(n))\}$.
- The least-mean-square (LMS) adaptive filter is a stochastic gradient version of the method of steepest descent that minimizes the mean-squared estimation error.
- The LMS algorithm is the most widely-used adaptive algorithm for FIR filters due to its computational simplicity and robust adaptation properties.
- Variants of the LMS adaptive filter include the sign-error, least-mean-K, and block LMS adaptive filters as well as adaptive filters with quantized updates. These other adaptive filters are useful in certain situations, depending on the implementation constraints and signals being processed.
- It is seen through examples that the LMS adaptive filter's behavior closely follows that of the method of steepest descent for small step sizes, and the LMS adaptive algorithm can achieve performance that approaches that of the optimum MMSE estimator in certain situations.

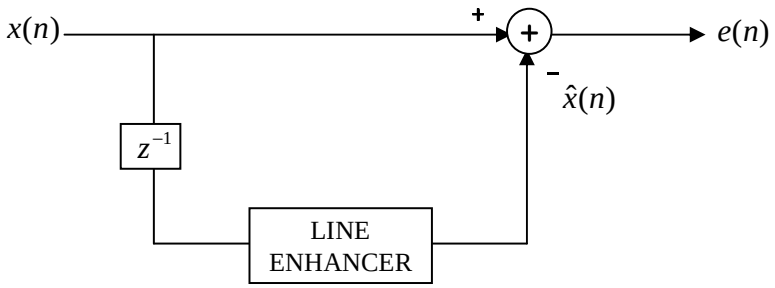

Figure 4.22: The configuration of the adaptive line enhancer for Example 4.4.

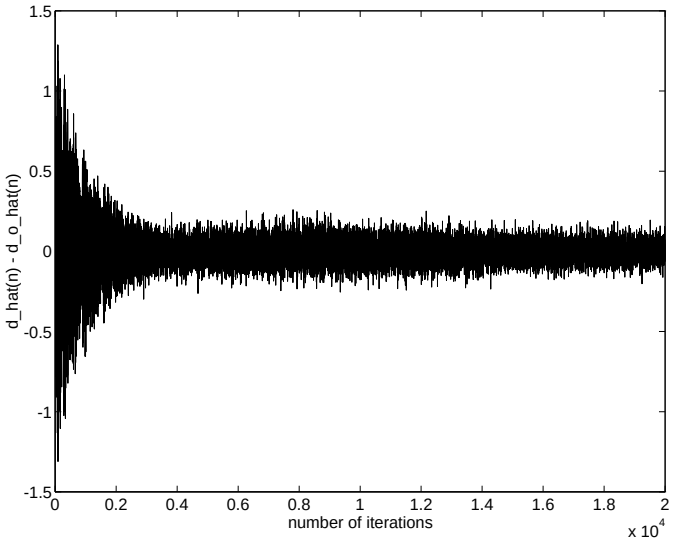

Figure 4.23: Difference between the outputs of the LMS adaptive line enhancer and the optimum MMSE line enhancer in Example 4.4.

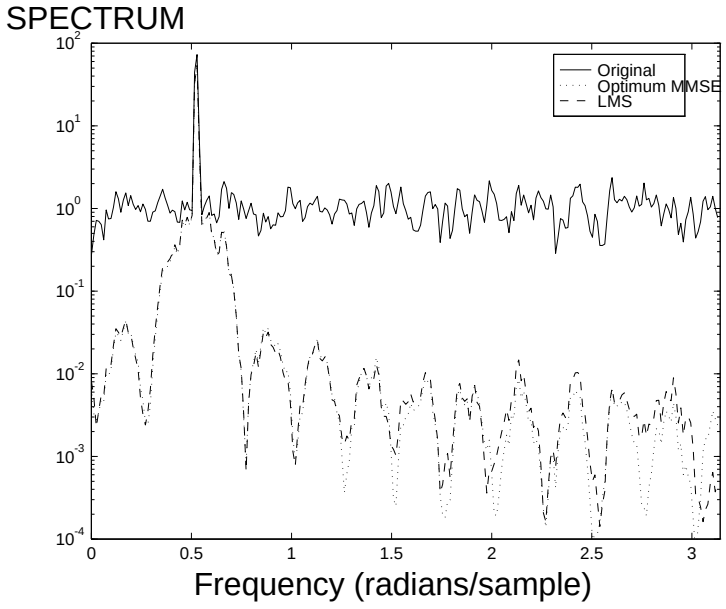

Figure 4.24: Spectra of the original noisy signal, the output of the optimum MMSE line enhancer, and the LMS adaptive line enhancer in Example 4.4.

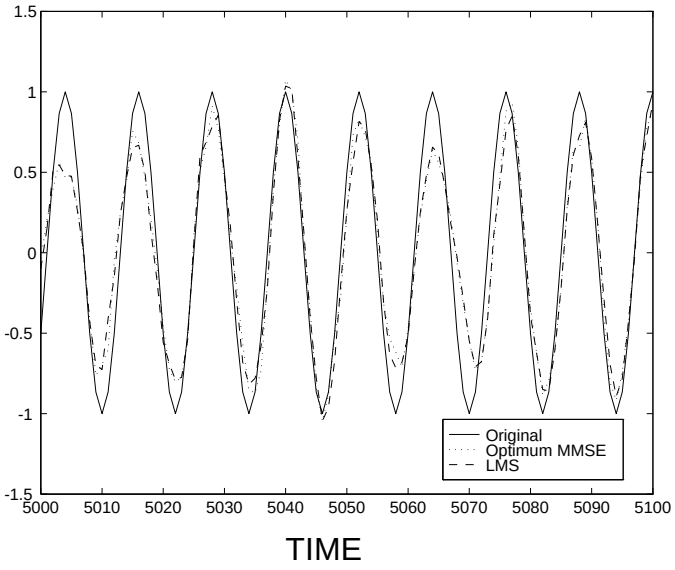

Figure 4.25: Time series of the original noiseless signal, the optimum MMSE line enhancer output, and the LMS adaptive line enhancer output in Example 4.4.

4.4 Bibliographical Notes

Method of Steepest Descent. The method of steepest descent first appeared in the context of the theory of optimization of parameterized functions [Curry 1944]. An excellent introduction to these methods can be found in [Luenberger 1984]. Newton's method is another well-known algorithm for determining the minimum of a locally-quadratic error surface (see Exercise 4.3). We also discuss this method in the context of recursive least-squares adaptive filters in Chapter 10.

Development of the LMS Adaptive Filter. The least-mean-square adaptive filter grew out of the efforts of several researchers working in the field of learning system in the late 1950's and early 1960's. The work of Widrow and Hoff [Widrow1960] is often credited as the first appearance of the algorithm in the literature, although the work of Rosenblatt [Rosenblatt 1957] is similar in both motivation and developed results. References to even earlier works than these have been noted in the literature; for example, Tsypkin [Tsypkin 1973] credits [Kaczmarz 1937] as the original work on the LMS algorithm with normalized step size. In the control literature, the LMS adaptive filter often appears in its continuous-time form and is referred to as "the MIT rule," in deference to the promoters of the algorithm in that field [Aström 1995].

Variations of the LMS Adaptive Filter. Due to the difficulties in computing multiplications in early digital hardware, early users of adaptive filters were forced to approximate the LMS adaptive filter's implementation using reduced-bit multiplications and additions. For an early application of these ideas, see [Lucky 1966]. A formal study of the sign-error adaptive filter is presented in [Gersho 1984], and least-mean-K adaptive algorithms are presented in [Walach 1984], respectively. A balanced presentation and analysis of several types of adaptive filters involving nonlinearities in the gradient update term can be found in [Duttweiler 1982]. Algorithms involving dual-sign and power-of-two error quantizers are considered in [Kwong 1986] and [Ping1986], respectively. We explore the performance and behavior of these modified algorithms more extensively in Chapter 7.

Work in the mid-1960's on the fast Fourier transform [Cooley1965] and fast convolution [Stockham 1966] paved the way for the development of the block LMS adaptive filter [Clark 1981]. For a good review of more recent work in block and frequency-domain adaptive filters, see [Shynk 1992]. These algorithms are also discussed in Chapter 8.

Applications of Adaptive Filters One of the first successful widespread applications of adaptive filters was in digital communications, where a modified version of the LMS adaptive filter was used in channel equalization [Lucky 1966]. Applications followed in geophysical exploration [Burg 1967], radar and sonar [Capon 1969, Frost 1972, Haykin 1985], medicine [Widrow 1975], speech processing and coding [Makhoul 1975, Gibson 1984], echo cancellation [Gritton 1984], image processing and coding [Benvenuto 1986], spread-spectrum

communications [Milstein 1986], beamforming [Van Veen 1988], and noise control [Elliott 1993], among others. A good review of applications in noise cancellation can be found in [Widrow 1975]. Quereshi [Quereshi 1988] gives an excellent overview of adaptive filters as used in digital communcations for channel equalization. A discussion of linear prediction as it applies to adaptive line enhancement appears in [Zeidler 1990].

4.5 Exercises

4.1. *Adaptive Filters are Nonlinear and Time-Invariant Systems:* Show, using the classical definitions of linearity and time-invariance that the LMS adaptive filter is a nonlinear and time-invariant system.

4.2. *The Sign-Sign Adaptive Filter:* Consider the following search technique, based on a simplification of the gradient search technique described by (4.5):

$$\begin{aligned}\mathbf{W}(n+1) &= \mathbf{W}(n) - \alpha \text{sgn}\left(\frac{\partial J(n)}{\partial \mathbf{W}(n)}\right) \\ &= \mathbf{W}(n) + \alpha \text{sgn}(e(n))\text{sgn}(\mathbf{X}(n)),\end{aligned}$$

where $[\text{sgn}(\mathbf{X})]_i = \text{sgn}(x_i)$ is as defined in (4.28) and $J(n)$ is the mean-square-error cost function.

a. Is the above search technique a true gradient search procedure? Why or why not?

b. Consider the one-dimensional case, for which $\mathbf{W}(n) = w(n)$. Explain why the above search technique will not converge to $\lim_{n\to\infty} w(n) = w_{opt}$ in general. Determine a bound on the coefficient error $\lim_{n\to\infty} |w(n) - w_{opt}|$ for an arbitrary initial value $w(0)$.

Hint: The bound depends on the value of the chosen step size α.

c. Even though the above method works well for most signals, there are a few situations in which the adaptive filter will diverge for all positive choices of μ. Show that the following situation is one such case.

The input signal $x(n)$ is periodic with a period of three samples and the first period is given by 3, -1, and -1. The desired response signal takes a constant value of one for all samples. The adaptive filter has three coefficients. Assume that at the beginning of some period, the adaptive filter coefficients are all zero.

4.3. *Newton's Method:* Newton's method is a classical technique for finding the minimum of a locally-quadratic performance surface [Luenberger 1984]. The algorithm is defined as

$$\mathbf{W}(n+1) = \mathbf{W}(n) - \alpha(\mathbf{F}(n))^{-1}\frac{\partial J(n)}{\partial \mathbf{W}(n)}, \tag{4.37}$$

where $\mathbf{F}(n)$ is an $L \times L$-element matrix whose (i,j)th value is given by

$$[\mathbf{F}(n)]_{i,j} = \frac{\partial^2 J(n)}{\partial w_i(n)\partial w_j(n)}. \tag{4.38}$$

a. Determine $\mathbf{F}(n)$ for the mean-square error criterion $J(n) = E\{e^2(n)\}$. Is the matrix a function of the time index n?

b. For your result in part a, determine conditions on the input and desired response signals so that $\mathbf{F}(n)$ can be inverted for all n.

c. Derive the coefficient update equation for Newton's method for the mean-square error criterion, and describe its convergence properties.

d. Describe the difficulties in developing a stochastic gradient version of Newton's method. Consider the amount of computation and knowledge of the signals required.

4.4. *Constant Modulus Error Criteria:* Consider the following cost function for a steepest descent procedure:

$$J(n) = E\{(A^2 - (\mathbf{W}^T(n)\mathbf{X}(n))^2)^2\}.$$

where A is a known amplitude. Such a *constant modulus* cost function makes use of the knowledge that the squared values of the desired output signal of the system is a constant value A at each iteration, a situation that is realistic in many digital communication systems.

a. Derive the steepest descent procedure for this cost function.

b. Determine a stochastic gradient version of this steepest descent procedure. How is it similar to the LMS adaptive filter?

c. Repeat parts a) and b) for the general constant modulus cost function given by

$$J(n) \quad = \quad E\{||A|^m - |\mathbf{W}^T(n)\mathbf{X}(n)|^m|^p\},$$

where $|\cdot|$ denotes absolute value and m and p are positive integers.

4.5. *The Statistics of the Output of a Single-Pole Filter With an I.I.D. Input Signal:* Consider the input signal model given by

$$x(n) \quad = \quad ax(n-1) + b\xi(n),$$

where $\xi(n)$ is an i.i.d. sequence with zero mean value and unit variance and a, b, and $x(0)$ are values to be specified.

a. Find an expression for $x(0)$ in terms of a and b such that the random sequence $\{x(n)\}$ has stationary second-order statistics, *i.e.*, $E\{x(n-i)x(n-j)\} = r_{xx}(i-j)$ for all $n-i > 0$, $n-j > 0$.

b. For your value of $x(0)$ in part a, find expressions for a and b in terms of $r_{xx}(0)$ and $r_{xx}(1)$.

4.6. *Equation Error Adaptive Recursive Filters:* Consider the identification of a recursive linear system as described in Example 2.16. We wish to develop an adaptive method for identifying such systems. An identification algorithm that employs feedback of the desired response signal $d(n)$ in the system model as in Example 2.16 is known as an *equation error algorithm.* (Another class of algorithms that uses delayed samples of $\hat{d}(n)$ in the system model is known as the *output error algorithms.* Output error adaptive recursive filters are described in Chapter 13.) Derive the the coefficient-updating strategy of an equation-error LMS adaptive filter using the recursive system model

$$\hat{d}(n) = \sum_{i=0}^{L} b_i(n)x(n-i) + \sum_{i=1}^{N} a_i(n)d(n-i),$$

where $x(n)$ and $d(n)$ are the input signal and the desired response signal, respectively, of the adaptive filter. Explain the possible advantages and disadvantages of this adaptive filter over the adaptive FIR filter.

4.7. *Adaptive Quadratic Filters:* Develop an adaptive LMS quadratic filter that models the relationship between the input signal and the desired response signal as

$$\hat{d}(n) = \sum_{i_1=0}^{L-1} \sum_{i_2=i_1}^{L-1} h_2(i_1, i_2; n)x(n-i_1)x(n-i_2).$$

A quadratic system identification problem is briefly discussed in Example 2.17.

4.8. *Linear Phase Adaptive Filters:* Derive an LMS adaptive filter that is constrained such that $w_i(n) = w_{L-i}(n)$ so that the filter coefficients at any time corresponds to that of a linear phase filter.

4.9. *Adaptive Filters With Variable Update Equations:* Develop a stochastic gradient adaptive filter that attempts to minimize the following cost function:

$$J(n) = \begin{cases} E\{|e^2(n)|\} & ; \quad |e(n)| < 1 \\ E\{|e^3(n)|\} & ; \quad |e(n)| \geq 1. \end{cases}$$

Discuss the possible advantages and disadvantages of your algorithm over the LMS adaptive filter.

4.10. *The Backpropagation Algorithm for A Single Artificial Neuron:* Consider the block diagram of the system in Figure 4.26, which depicts the structure of an L-input, one-output artificial neuron. When several of these structures are cascaded together, they form a *feedforward artificial neural network.* The output of this system is

$$y(n) \quad = \quad f\left(\sum_{i=1}^{N} w_i(n)x_i(n)\right), \tag{4.39}$$

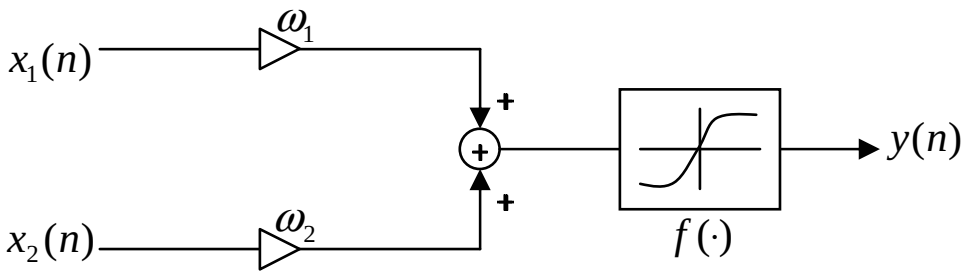

Figure 4.26: A single artificial neuron.

where $x_i(n)$ is the ith input signal and $w_i(n)$ is the ith neuron coefficient. A common choice for the function $f(u)$ is

$$\begin{aligned} f(u) &= \frac{e^{\alpha u} - e^{-\alpha u}}{e^{\alpha u} + e^{-\alpha u}} \\ &= \tanh(\alpha u), \end{aligned}$$

which is also known as the *sigmoid function* in the neural network field.

a. Derive a stochastic gradient algorithm for adjusting the ith coefficient of the artificial neuron to approximately minimize the mean-squared error $J(n) = E\{e^2(n)\}$, where $e(n) = d(n) - y(n)$. Express your answer in vector form.

b. From your result in part a, is the update for $w_i(n)$ linear in the instantaneous values of the parameters $\{w_i(n)\}$?

4.11. *The Complex LMS Adaptive Filter:* The generalization of the LMS adaptive filter to complex-valued signals and coefficients is useful in communication systems, where the complex signal representation is used to describe the in-phase and quadrature components of the received signal.

Let $x(n)$ be defined as

$$x(n) = x_R(n) + jx_I(n),$$

where $x_R(n)$ and $x_I(n)$ are the real and imaginary components of the input signal. Similarly, let

$$w_i(n) = w_{R,i}(n) + jw_{I,i}(n)$$

denote the ith complex-valued filter coefficient. Then, the output of the system is defined as

$$y(n) = \sum_{i=0}^{L-1} w_i(n)x(n-i)$$

as before. Define the error signal $e(n)$ as

$$\begin{aligned} e(n) &= d(n) - y(n) \\ &= (d_R(n) - y_R(n)) + j(d_I(n) - y_I(n)). \end{aligned}$$

Show that the stochastic gradient algorithm for adjusting the coefficient vector $\mathbf{W}(n)$ to approximately minimize the mean-squared value of the *absolute value* of the error, given by $E\{|e(n)|^2\}$, is

$$\mathbf{W}(n+1) = \mathbf{W}(n) + \mu e(n)\mathbf{X}^*(n), \tag{4.40}$$

where the ith element of $\mathbf{X}^*(n)$ is the complex conjugate of $x(n-i+1)$. In this case, the differentiation of a real-valued function $f(u)$ with respect to its complex-valued argument $u = u_R + ju_I$ is defined as

$$\frac{\partial f(u)}{\partial u} = \frac{\partial f(u)}{\partial u_R} + j\frac{\partial f(u)}{\partial u_I}. \tag{4.41}$$

4.12. *The Filtered-X LMS Adaptive Filter:* Consider the block diagram of the system in Figure 4.27, where the output of an adaptive filter is passed through another fixed FIR filter with impulse response vector $\mathbf{H} = [h_0\ h_1\ \cdots\ h_{M-1}]^T$. Such a block diagram often arises in adaptive control systems. The error signal $e(n)$ in this situation is given by

$$e(n) = d(n) - \sum_{m=0}^{M-1} h_m y(n-m).$$

a. Develop a version of the LMS adaptive filter that minimizes the mean-squared error cost function $E\{e^2(n)\}$. In your derivation, assume that

$$\frac{\partial y(n-m)}{\partial \mathbf{W}(n)} \approx \frac{\partial y(n-m)}{\partial \mathbf{W}(n-m)} = \mathbf{X}(n-m).$$

b. Draw a block diagram of the resulting system that uses the fewest number of multiplications and additions possible

Hint: The minimum number of mutiplications necessary is $2L + M + 1$ per iteration.

c. What are the implications of the assumption that you used in part a to derive the algorithm on the choice of step size μ for this system?

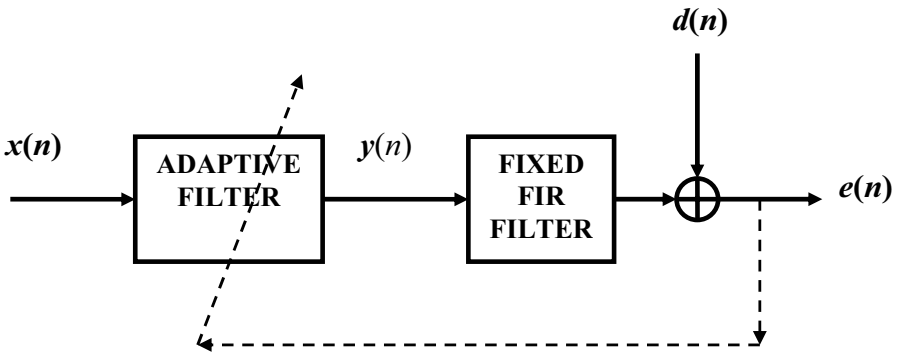

Figure 4.27: LMS adaptive filter for adaptive control.

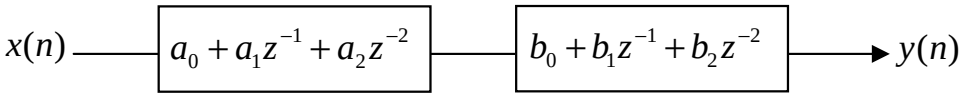

Figure 4.28: Cascade-form LMS adaptive filter

.

4.13. *Cascade-Form LMS Adaptive Filter:* Consider the cascade form structure of the system model shown in Figure 4.28. Develop an LMS adaptive filter that attempts to minimize the squared estimation error at each time instant for the parameters $a_0(n)$, $a_1(n)$, $a_2(n)$, $b_1(n)$, and $b_2(n)$. Does the mean-square error surface for this problem have a unique minimum?

Hint: Consider the approximation used in Problem 4.12 above.

4.14. *Optimum MMSE Solution for Nonstationary Channel Equalization:*

a. Show through direct solution of the equation $\mathbf{R}_{\mathbf{xx}}(n)\mathbf{W}_{opt}(n) = \mathbf{P}_{\mathrm{d}\mathbf{x}}$ that the minimum mean-square-error solution for the nonstationary channel equalization problem in Example 4.3 is given by

$$\mathbf{W}_{opt}(n) = \begin{bmatrix} 1 \\ -a(n) \end{bmatrix}. \tag{4.42}$$

b. Does this result hold if $\{s(n)\}$ is a nonstationary i.i.d. random sequence? Explain.

4.15. *The Continuous-Time LMS Adaptive Filter:* Consider the continuous-time system defined as

$$y(t) = \int_{-\infty}^{\infty} w(s)x(t-s), \tag{4.43}$$

where $w(t)$ is the impulse response of the continuous-time filter. Determine a differential equation "update" for $w(t)$ of the form

$$\frac{dw(t)}{dt} = \mu \frac{de^2(t)}{dw(t)}, \tag{4.44}$$

where $e(t) = d(t) - y(t)$.

4.16. *Computing Assignment on Adaptive Prediction:* This assignment evaluates the performance of the LMS adaptive filter in a prediction problem. For this, we consider an input signal that is generated using the model

$$x(n) = 0.44\xi(n) + 1.5x(n-1) - x(n-2) + 0.25x(n-3),$$

where $\xi(n)$ is a zero-mean, i.i.d, Gaussian-distributed random process with unit variance.

a. Obtain an expression for the power spectral density of $x(n)$.

b. Find the coefficients of the MMSE, one-step linear predictor for $x(n)$ that employs four coefficients.

c. Develop an adaptive LMS predictor employing four coefficients for $x(n)$.

d. Evaluate the 4×4-element autocorrelation matrix and the 4-element cross-correlation vector for this prediction problem. Derive the evolution equations for the mean values of each adaptive predictor coefficient for $\mu = 0.01$, and zero initial coefficient values. Find the steady-state misadjustment for this step size.

e. Generate a 2000-sample sequence using the model for $x(n)$ described earlier. Evaluate the mean coefficient behavior using fifty independent experiments. Compare the empirical averages with the theoretical equations of part d.

f. Plot the mean-squared prediction error obtained by averaging the squared prediction errors of the fifty experiments. If the steady state appears to have been reached, evaluate the mean-square prediction error as the ensemble average of the time average of the last one hundred samples of the squared errors in each run over the fifty experiments. Compare the empirical misadjustment with its theoretical value.

g. Explain the possible reasons for the differences between the theoretical and empirical results.

4.17. *Computing Assignment on Adaptive Interference Cancellation:* One significant problem that occurs in test equipments such as electro-cardiographs (ECG) and electro-encephalographs (EEG) is the inability to completely isolate the devices from line voltages. Since the measurements made by these machines typically range in the microvolts, even a small leakage of the line voltage can completely obscure the desired measurements. Fortunately, the source of interference is known in this case and we can use this information to cancel the interference adaptively. A block diagram of the system one would employ for this application is shown in Figure 4.29. The desired response signal contains the signals $f(n)$ that we want extracted. The interference signal is different from the input signal by an unknown initial phase and an unknown amplitude value as shown in the figure. Assuming that $f(n)$ is uncorrelated with the source of interference $x(n)$, we can argue that the estimate of $d(n)$ using $x(n)$ will estimate only the interference and, therefore, the estimation error signal is a cleaner version of the signal $f(n)$.

a. Develop an adaptive interference canceller using the ideas described above.

b. To simulate an ECG signal, generate a triangular waveform $f(n)$ with period twenty samples and a peak value of 0.1 volt. Also generate a sinusoidal signal $x(n)$ with amplitude 1 volt and frequency 60 Hz and sampled at a rate of 200 samples/second. Generate 2000 samples of each signal. You can simulate the corrupted signal using the model

$$d(n) = f(n) + 0.5 \sin\left(\frac{120\pi}{200}(n - 0.25)\right).$$

Figure 4.29: Block diagram of an adaptive interference canceller.

By trial and error, as well as your understanding about the predictability of sinusoids, find a good choice for the number of coefficients for the adaptive filter and the step size. Plot the enhanced version of $f(n)$ obtained as the estimation error of the adaptive filter. Comment on the performance of the interference canceller you developed.

4.18. *Computing Assignment on Adaptive Frequency Tracking:* In this assignment, we consider the problem of tracking the instantaneous frequency of an FM signal modeled as

$$x(t) = \cos\left(2\pi 0.25t - 0.025\cos(4\pi t)\right) + \eta(n),$$

where $\eta(n)$ is an additive noise signal that is uncorrelated with the sinusoidal components. We can compute the instantaneous frequency of this signal by finding the derivative of the instantaneous phase function given by

$$\theta(t) = 2\pi 0.25t - 0.025\cos(4\pi t).$$

Our approach to finding the instantaneous frequency is to use a L-coefficient predictor for $x(t)$ after sampling it and evaluating the frequency corresponding to the peak of the autoregressive spectrum estimate obtained from the coefficients at each time. See Example 2.15 for a description of the autoregressive spectrum estimation technique.

a. Generate 2000 samples of a discrete version of the input FM signal by sampling it at a rate of 1000 samples/second. The noise component may be modeled as a white, Gaussian process with zero mean value and variance $\sigma_\eta^2 = 0.01$. Develop an adaptive predictor for this signal.

b. By trial and error, find good choices of the step size and coefficient length for this adaptive filter to track the frequencies. You can use the model of the instantaneous frequency to guide you in your selection process. Estimate the instantaneous frequency by calculating the autoregressive spectrum estimate every ten samples. After compensating for the normalization of the frequency variable during sampling, plot the estimated instantaneous frequencies against their true values.

c. Document your observations on this experiment.

4.19. *Computing Assignment on FIR Filter Design.* This assignment guides you through the design of time-invariant FIR filters from specifications using the LMS adaptive filter. Consider the problem of designing a linear phase FIR filter that meets the following specifications:

$$\begin{array}{rcccccccl} 0.9 & \leq & |H(\omega)| & \leq & 1.1 & ; & 0 & \leq & |\omega| & \leq & \pi/4 \\ & & |H(\omega)| & \leq & 0.01 & ; & \pi/2 & \leq & |\omega| & < & \pi/4. \end{array}$$

We can design this filter using the adaptive filter by creating the appropriate input and desired response signals for the adaptive filter. Create an input signal as

$$x(n) = \sum_{i=1}^{K} A_1 \cos(\omega_i n + \phi_i),$$

where the frequencies are uniformly sampled from the passband and stop band of the desired filter response and the phase values ϕ_i's are uncorrelated with each other and uniformly distributed in the range $[-\pi, pi)$. Let the ideal filter response be

$$H_I(\omega) = \begin{cases} e^{j\theta_I(\omega)} & ; \quad 0 \leq |\omega| \leq \pi/4 \\ 0 \ ; & \text{otherwise} \end{cases} .$$

(What should $\theta_I(\omega)$ be for an L-coefficient filter to have linear phase characteristics?) Since the input is a sum of sinusoids, we can easily find the output of the ideal filter and use it as the desired response signal for the adaptive filter. Use an adaptive filter with the above input signal and desired response signal and find the coefficients of the adaptive filter when it reaches the steady-state. You may even average the coefficients over a long duration of time after the adaptive filter has effectively converged. You must verify that the approximation obtained using the adaptive filter meets the specifications.

When you perform the experiments, keep the following points in mind. The number of sinusoidal components in the input signal should be fairly large. The amplitude values A_i's may all be chosen to be the same. If they are differently chosen, you are weighting the sinusoids differently, thereby emphasizing the specifications in certain

regions more than those at other regions. Remember that the acceptable values of the step size depends on the input signal power. You may have to run the adaptive filter for a long time. Consider impulse response lengths upto 128 samples. Choose the design that employs the minimum number of coefficients and still meets the specifications.