CHAPTER 5

Method of Stochastic Gradient Descent

In the preceding chapter, we studied the method of steepest descent for formulating recursive computation of the Wiener filter under the following two assumptions:

1. Joint stationarity of the environment, from which the regressor (i.e., input vector) $u(n)$ and the corresponding desired response $d(n)$ are picked for $n = 1, 2, \ldots$
2. Information about the environment, which is made up of the correlation matrix, $R$, of $u(n)$ and the cross-correlation vector, $p$, between $u(n)$ and $d(n)$.

Insofar as formulation of the cost function for the Wiener filter is concerned, namely, $J(w(n))$, where $w(n)$ is the unknown tap-weight vector, the information made up of $R$ and $p$ is perfect for the discussion at hand [see Eq. (4.8)]. As pointed out previously in Section 4.5, the method of steepest descent is deterministic, in the sense that this information is all that is required to determine the gradient (i.e., the search direction) from one adaptation cycle to the next.

Unfortunately, in many situations encountered in practice, the needed information about the environment contained in $R$ and $p$ is not available. We therefore have to look to a new class of algorithms with the built-in capability to adapt to statistical variations in the unknown environment. Such algorithms are collectively called adaptive filtering algorithms. There are two different methods for deriving these algorithms:

1. One based on the method of stochastic gradient descent, which is discussed in this chapter.
2. The other one based on the method of least squares, which is deferred to Chapter 9.

5.1 PRINCIPLES OF STOCHASTIC GRADIENT DESCENT

"Stochastic" is of Greek origin; it is a generic term, which signifies that we have a "random choice" in finding a gradient for an adaptive filtering algorithm to iterate from one adaptation cycle to the next. As different as they both are, the methods of stochastic gradient descent and steepest descent do share a common property: They are both local methods of filtering.

---

1The method of stochastic gradient descent has a long history. It was originally introduced into the statistics literature by Robbins and Monro (1951), where the method was used for solving certain sequential parameter estimation problems. For detailed treatments of the method, the reader is referred to Kushner and Clark (1978), Kushner (1984), and Spall (2003).
Over and above its application to the widely used least-mean-square (LMS) algorithm and its variants, the method of stochastic gradient descent has been applied to stochastic control (Stengel, 1986), self-organized maximum eigenfiltering (Oja, 1982), nearest-neighbor clustering (Duda et al., 2001), back-propagation algorithm for supervised training of multilayer perceptrons (Rumelhart & McLelland, 1986), and reinforcement learning (Sutton, 1992).

**Optimization and Complexity**

In a rigorous sense, the method of stochastic gradient descent does not qualify to be viewed as an optimization algorithm for the following simple reason:

*On account of its stochastic nature, stochastic gradient descent can never reach the desired optimum solution of a convex optimization problem; rather, once it reaches the local neighborhood of the optimum solution, it continually wanders around that solution in a random-walk fashion and therefore never settles down to an equilibrium point.*

As such, the method of stochastic gradient descent is *suboptimal*.

The method makes up for this computational deficiency, however, by offering the simplest possible form of complexity: *linear law of scaling* with respect to adjustable parameters. It is therefore popular in practice whenever computational complexity (among other issues) is of interest. This is particularly so when dealing with the processing of information-bearing data whose size grows continually over time.

**Efficiency**

Another key issue in the study of adaptive filtering algorithms is that of *efficiency*, which may be viewed as a representation of the cost involved in finding a satisfactory solution. There are various ways of measuring efficiency: the amount of computer run time, number of algorithmic adaption cycles, and rate of convergence. In what follows, the latter measure will be used for the following reason:

*The rate of convergence not only involves statistical learning theory but also encompasses the Wiener solution as the frame of reference for linear adaptive filtering algorithms, exemplified by the LMS and recursive least-squares (RLS) algorithms, operating in a stationary environment.*

**Robustness**

Yet another key issue in the study of adaptive filtering algorithms is that of *robustness*, which provides a criterion for assessing how a particular adaptive filtering algorithm maintains a satisfactory performance in the face of unknown disturbances that are typically expected to arise in practice. The classical approach used to evaluate robustness is a deterministic one, rooted in *H*\(^*\) *theory*, which was first formulated in control theory (Zames, 1981); detailed treatment of this property is deferred to Chapter 11. For the present, it suffices to say that robustness and efficiency address two conflicting aspects of an adaptive filtering algorithm's behavior, more on which will also be said in Chapter 11.
Curse of Dimensionality

In general, the volume of a search space, inside which the gradient of a cost function is picked at each adaptation cycle of an optimization algorithm, may grow exponentially with dimensionality (i.e., number of adjustable parameters) of the algorithm. This kind of behavior is attributed to the curse of dimensionality problem (Bellman, 1961). In the case of stochastic gradient descent algorithms, which feature prominently throughout this book, algorithmic complexity follows a linear law, as pointed out at the beginning of this section. Fortunately, therefore, the curse of dimensionality is of no concern to the study of linear adaptive filtering algorithms rooted in stochastic gradient descent.

Time-Varying Problems

In many practical applications of adaptive filtering algorithms, the relevant environment is nonstationary. Therefore, its statistical characterization continually varies over time. In situations of this kind, the “best” solution to an adaptive filtering problem computed now may not be the best—or even a good—solution in the future. For an adaptive filtering algorithm to operate successfully in a nonstationary environment, it must have the capability to track statistical variations in the environment continually over time.

Monte Carlo Simulations

Given the realities just described, how do we study the performance of an adaptive filtering algorithm experimentally? The answer to this important question lies in the use of Monte Carlo simulations, which provide the experimenter the means not only to gain insight about that algorithm but also to compare it with a different adaptive filtering algorithm under various conditions. For this computer-oriented approach to be informative, it is essential that we do the following:

- First, use a number of independent Monte Carlo runs in the simulations large enough (at least 100 if not more) for the simulation results to be statistically reliable.
- Second, whenever possible, combine together both theory and numerical results in assessing the insights derived from the simulations.

With the material covered thus far on the method of stochastic gradient descent, the stage is set for two applications of the method to linear adaptive filtering.

5.2 APPLICATION 1: LEAST-MEAN-SQUARE (LMS) ALGORITHM

For our first application of the method of stochastic gradient descent, we have chosen the highly popular adaptive filtering algorithm, the LMS algorithm, which was pioneered by Widrow and Hoff (1960). Distinctive features of this algorithm can be summarized as follows:

1. The LMS algorithm is simple, meaning that computational complexity of the algorithm scales linearly with the dimensionality of the finite-duration impulse response (FIR) filter, around which the algorithm operates.
2. Unlike the Wiener filter, the algorithm does not require knowledge of statistical characteristics of the environment in which it operates.

3. The algorithm is robust in a deterministic sense (i.e., single realization of the algorithm) in the face of unknown environmental disturbances.

4. Last but by no means least, the algorithm does not require inversion of the correlation matrix of the regressor (i.e., input vector), which, therefore, makes it simpler than its counterpart, namely, the RLS algorithm.

Given this important set of properties, it is not surprising that the LMS algorithm is one of the most popular adaptive filtering algorithms in use.

**Structural Description of the LMS Algorithm**

Figure 5.1, depicted in three parts, addresses different perspectives of the underlying structure of the LMS algorithm.

The overall block diagram of Fig. 5.1(a) shows the three components that constitute the algorithm:

1. **FIR filter**, which operates on the regressor (input vector) \( u(n) \) to produce an estimate of the desired response, denoted by \( \hat{d}(n | u_n) \), where \( u_n \) denotes the space in which the input vector \( u(n) \) resides.

2. **Comparator**, which subtracts the estimate \( \hat{d}(n | u_n) \) from the desired response, \( d(n) \), applied to the FIR filter at its output; the resultant is the estimation error (also referred to as the error signal), denoted by \( e(n) \).

3. **Adaptive weight-control mechanism**, the function of which is to control the incremental adjustments applied to the individual tap weights of the FIR filter by exploiting information contained in the estimation error \( e(n) \).

Details of the FIR filter are presented in Fig. 5.1(b). The tap inputs \( u(n), u(n-1), \ldots, u(n-M+1) \) form the elements of the \( M \)-by-1 tap-input vector \( u(n) \), where \( M-1 \) is the number of delay elements; these inputs span a multidimensional space denoted by \( u_n \). Correspondingly, the tap weights \( \hat{w}_0(n), \hat{w}_1(n), \ldots, \hat{w}_{M-1}(n) \) form the elements of the \( M \)-by-1 tap-weight vector \( \hat{w}(n) \). The value computed for this vector using the LMS algorithm represents an estimate whose expected value may come close to the Wiener solution \( w_o \) (for a wide-sense stationary environment) as the number of adaptation cycles, \( n \), approaches infinity.

Figure 5.1(c) presents details of the adaptive weight-control mechanism. Specifically, a scalar version of the inner product of the estimation error \( e(n) \) and the tap input \( u(n-k) \) is computed for \( k = 0, 1, 2, \ldots, M-2, M-1 \). The result so obtained defines the correction \( \delta \hat{w}_k(n) \) applied to the tap weight \( \hat{w}_k(n) \) at adaptation cycle \( n+1 \). The scaling factor used in this computation is denoted by a positive \( \mu \) in Fig. 5.1(c) called the step-size parameter, which is real-valued.

Comparing the control mechanism of Fig. 5.1(c) for the LMS algorithm with that of Fig. 4.2 for the method of steepest descent, we see that the LMS algorithm uses the product \( u(n-k)e^*(k) \) as an estimate of element \( k \) in the gradient vector \( \nabla J(n) \) that characterizes the method of steepest descent. In other words, the expectation operator is removed from all the paths in Fig. 5.1(c). Accordingly, the recursive computation of each tap weight in the LMS algorithm suffers from gradient noise.
FIGURE 5.1 (a) Block diagram of adaptive FIR filter. (b) Detailed structure of the FIR filter component. (c) Detailed structure of the weight-control mechanism.
Throughout this chapter, it is assumed that the tap-input vector \( \mathbf{u}(n) \) and the desired response \( d(n) \) are drawn from a jointly wide-sense stationary environment, In particular, the desired response \( d(n) \) is \textit{linearly} related to the input vector (i.e., regressor) \( \mathbf{u}(n) \) by a multiple linear regression model whose parameter vector is unknown—hence the need for adaptive filtering. For such an environment, we know—rather than terminating on the Wiener solution, the tap-weight vector \( \mathbf{w}(n) \) computed by the LMS algorithm moves in the form of a \textit{random walk} around the minimum point of the error-performance surface—that is, the Wiener solution.

In Eq. (2.3) of Chapter 2 on the Wiener filter, which is reproduced here for convenience, we defined the cost function as the mean-square value of the estimation error, as shown by

\[
J = \mathbb{E}[|e(n)|^2],
\]  

(5.1)

where \( \mathbb{E} \) is the statistical expectation operator. Therein, it was assumed that the Wiener filter operates on a wide-sense stationary environment, which results in a cost function, \( J \), that is independent of time \( n \), as shown in Eq. (5.1). To this end:

\textit{The expectation operator performs ensemble averaging over a large number of statistically independent realizations of the instantaneous value of the square estimation error, } \( |e(n)|^2 \), \textit{which is performed at time } \( n \).

Unfortunately, in practical applications of adaptive filtering, the use of ensemble averaging in the manner just described is not feasible. We say so because the whole motivation behind the deployment of adaptive filtering is to \textit{adapt} to statistical variations of an unknown environment in an \textit{on-line manner}, based on a single realization of the estimation error, \( e(n) \), as it evolves across time \( n \). Doing so is precisely what the method of stochastic gradient descent is all about. We may therefore proceed by ignoring the expectation operator in Eq. (5.1), thereby working on the following simplified cross-function:

\[
J_s(n) = |e(n)|^2 = e(n)e^*(n),
\]  

(5.2)

where the subscript \( s \) in \( J_s(n) \) is intended to differentiate it from its ensemble-average counterpart, \( J \). With the estimation error \( e(n) \) being the \textit{sample function} of a stochastic process, it follows that the cost function \( J_s(n) \) is itself the sample value of a stochastic process. The derivative of \( J_s(n) \) with respect to the \( k \)th tap weight of the FIR filter, \( w_k(n) \), is therefore also stochastic, which is how it should be in the method of stochastic gradient descent.

Following the procedure described in Appendix B based on the Wirtinger calculus for computing gradients of complex data, the first use of which was discussed in
Chapter 2 on Wiener filters, we may express the partial derivative of the cost function \( J_k(n) \) with respect to the complex conjugate of \( w_k(n) \) as

\[
\nabla J_{s,k}(n) = \frac{\partial J_k(n)}{\partial w_k^*(n)}
\]

\[
= -2u(n-k)e^*(n), \quad k = 0, 1, \ldots, M - 1,
\]

which is exactly the same as that of Eq. (2.10) in Chapter 2, except for the expectation operator \( \mathbb{E} \). Using the stochastic gradient of Eq. (5.3), we may now formulate the updating rule for the LMS algorithm as follows:

\[
\hat{w}_k(n + 1) = \hat{w}_k(n) - \frac{1}{\mu} \nabla J_{s,k}(n),
\]

where the scaling factor \( 1/\mu \) has been introduced merely for mathematical convenience. Thus, substituting Eq. (5.3) into Eq. (5.4), we obtain

\[
\hat{w}_k(n + 1) = \hat{w}_k(n) + \mu u(n-k)e^*(n), \quad k = 0, 1, \ldots, M - 1,
\]

where \( \mu \) is a fixed positive step-size parameter.

Equation (5.5) is the scalar form of the LMS algorithm. To express it in vector form using matrix notation, let

\[
\hat{\mathbf{w}}(n) = [\hat{w}_0(n), \hat{w}_1(n), \ldots, \hat{w}_{M-1}(n)]^T,
\]

where the superscript \( T \) denotes transposition, and

\[
\mathbf{u}(n) = [u(n), u(n-1), \ldots, u(n-M-1)]^T.
\]

We may thus rewrite Eq. (5.5) in the following compact form:

\[
\hat{\mathbf{w}}(n + 1) = \hat{\mathbf{w}}(n) + \mu \mathbf{u}(n)e^*(n),
\]

where the asterisk denotes complex conjugation.

By definition, we have

\[
e(n) = d(n) - \hat{d}(n | \mathcal{U}_n),
\]

where

\[
\hat{d}(n | \mathcal{U}_n) = \hat{\mathbf{w}}^H(n)\mathbf{u}(n),
\]

where the superscript \( H \) denotes Hermitian transposition (i.e., transposition combined with complex conjugation). Thus Eqs. (5.6) to (5.8), put together, define the LMS algorithm.

**Another Way of Deriving the LMS Algorithm**

The updated formula of Eq. (5.6) may also be obtained exactly from Eq. (4.10), describing iterative computation of the Wiener filter using the method of steepest descent that was covered in Chapter 4. Specifically, all that we have to do is to replace the correlation
CHAPTER 6

The Least-Mean-Square (LMS) Algorithm

In this relatively long chapter, we build on the derivation of the least-mean-square (LMS) algorithm as an application of the method of stochastic gradient descent that was presented in the preceding chapter. Specifically, we will expand on why the LMS algorithm is of fundamental importance in linear adaptive filtering in theory as well as application.

We begin the chapter with a signal-flow graph representation of the LMS algorithm, which clearly exhibits the fact that the LMS algorithm is basically a nonlinear feedback control system. With feedback known to be a “double-edged sword,” it can work for us or against us. It is not surprising, therefore, to find that with the control mechanism being directly dependent on how the step-size parameter, $\mu$, is chosen. This parameter plays a critical role in assuring convergence of the algorithm (i.e., its stability when viewed as a feedback control system) or in failing to do so.

The study of convergence is carried out under the umbrella of statistical learning theory of the LMS algorithm, which occupies a good part of the chapter. Although, indeed, the LMS algorithm is simple to formulate, its mathematical analysis is very difficult to carry out. Nevertheless, ideas related to efficiency of the algorithm that are derived from this theory are supported through the use of Monte Carlo simulations.

6.1 SIGNAL-FLOW GRAPH

For convenience of presentation, we reproduce the LMS algorithm summarized in Table 5.1, as follows:

\begin{align}
y(n) &= \hat{w}(n)^H u(n), \\
e(n) &= d(n) - y(n),
\end{align}

and

\begin{align}
\hat{w}(n + 1) &= \hat{w}(n) + \mu u(n)e^*(n),
\end{align}

where $u(n)$ is the input vector (regressor), $d(n)$ is the corresponding desired response, and $\hat{w}(n)$ is an estimate of the unknown tap-weight vector, $w(n)$, of the linear multiple regression model used to represent the environment from which $u(n)$ and $d(n)$ are jointly picked. The superscript H denotes Hermitian transposition (i.e., transposition combined with complex conjugation), and the asterisk denotes complex conjugation.
FIGURE 6.1 Signal-flow graph representation of the LMS algorithm, where I is the identity matrix and $z^{-1}$ is the unit-time delay operator.

Given this set of equations, we may construct a signal-flow graph of the LMS algorithm as shown in Fig. 6.1. Based on this diagram, we see that the LMS algorithm embodies two basic processes that continually build on each other:

1. **Filtering process.** This first process involves two operations:
   - one computing the output, $y^*(n)$, of the finite-duration impulse response (FIR) filter within the algorithm, in response to the input signal $u(n)$, and
   - the other one generating the estimation error, $e^*(n)$, by subtracting $y^*(n)$ from $d^*(n)$.

2. **Adaptation process.** This second process involves updating the present value of the estimated weight vector $\hat{w}(n)$ by an “incremental” amount equal to the product term $\mu u(n)e^*(n)$ to produce $\hat{w}(n + 1)$, where the incrementality is assured by choosing a small value for the step-size parameter, $\mu$.

These two processes are identified on the right- and left-hand sides of Fig. 6.1, respectively. Henceforth, each complete cycle of the LMS algorithm is referred to as an adaptation cycle.

From Fig. 6.1, we readily find that:

Computational complexity of the LMS algorithm scales linearly with dimensionality of the estimate $\hat{w}(n)$.

This statement reemphasizes simplicity of the LMS algorithm that was pointed out in Chapter 5.

---

1 The multiplier on the left in Fig. 6.1 signifies ordinary multiplication, and the multiplier on the right signifies inner-product formulation.
6.2 OPTIMALITY CONSIDERATIONS

There are two ways in which optimality of the LMS algorithm may be considered. The first involves data-induced perturbation, for which the LMS algorithm is optimal in a localized sense. The second is an overall sense, for which, as expected, the LMS algorithm is suboptimal.

Localized Optimality

In Chapter 5, we pointed out that the method of stochastic gradient descent is of a "local" kind, meaning that the search for a gradient vector is chosen not only randomly but also in a localized manner. Despite its stochastic behavior, the LMS algorithm is capable of exhibiting localized optimality in a Euclidean sense, provided that the data-induced perturbation from one adaptation cycle to the next one is small enough (Sayeed, 2003). In a way, this requirement works in the best interests of the algorithm as an adaptive filter.

To explain what we have in mind here, we consider two different scenarios:

1. **Current estimation error.** Given the supervised-training set \( \{u(n), d(n)\} \), we write

   \[
   e(n) = d(n) - w^H(n)u(n),
   \]

   where \( w(n) \) is an unknown tap-weight vector to be optimized in a localized Euclidean sense.

2. **Posterior estimation error.** In this second scenario, the requirement is to optimize the updated tap-weight vector \( w(n + 1) \), for which we write

   \[
   r(n) = d(n) - w^H(n + 1)u(n).
   \]

   Here, it should be noted that the error \( r(n) \) is different from the one-step prediction error that involves the product term \( w^H(n)u(n + 1) \).

   For the localized perturbation to be small, we require that the step-size parameter, \( \mu \), is small enough to satisfy the following condition:

   \[
   |1 - \mu \|u(n)\|^2| < 1 \quad \text{for all } u(n),
   \]

   where \( \|u(n)\|^2 \) is the squared Euclidean norm of the input vector \( u(n) \). Equivalently, we may express this condition as follows:

   \[
   0 < \frac{1}{2} \mu \|u(n)\|^2 < 1 \quad \text{for all } u(n).
   \]

   We may now state the problem at hand:

   Find the optimum value of \( w(n + 1) \) that minimizes the squared Euclidean between it and its past value \( w(n) \), subject to the condition described in Eq. (6.6).

In effect, the localized problem of interest is a constrained optimization problem.

To solve this problem, we use the well-known method of Lagrange multipliers, described in Appendix C. Specifically, we begin by introducing the Lagrangian function:
Section 6.2  Optimality Considerations

\[ L(n) = \frac{1}{2} \| \mathbf{w}(n+1) - \mathbf{w}(n) \|^2 + \lambda(n) \left( r(n) - \left( 1 - \frac{\mu}{2} \| \mathbf{u}(n) \|^2 \right) e(n) \right), \]  

(6.7)

where \( \lambda(n) \) is a time-dependent Lagrangian multiplier; \( e(n) \) and \( r(n) \) are defined in Eqs. (6.4) and (6.5). Applying Wirtinger calculus of Appendix B to differentiate \( L(n) \) with respect to \( \mathbf{w}^H(n+1) \) and formally treating \( \mathbf{w}(n) \) as a constant, we obtain the partial derivative

\[ \frac{\partial L(n)}{\partial \mathbf{w}^H(n+1)} = \mathbf{w}(n+1) - \mathbf{w}(n) - \lambda(n) \mathbf{u}(n). \]

Let \( \hat{\mathbf{w}}(n+1) \) and \( \hat{\mathbf{w}}(n) \) be the tap-weight vectors for which this partial derivative is zero. Accordingly, we obtain

\[ \hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \lambda(n) \mathbf{u}(n). \]  

(6.8)

To find the corresponding value of the Lagrange multiplier, we substitute the solution of Eq. (6.8) in Eq. (6.7). Then, cancelling common terms in the resulting equation, we obtain

\[ \lambda(n) = -\mu e^*(n). \]

for which the Lagrangian function \( L(n) \) attains its minimum value of zero. Recognizing that the parameter \( \mu \) is a positive constant, it follows that the Lagrangian multiplier is complex valued for all \( n \).

Finally, using the \( \lambda(n) \) in Eq. (6.8), we get the desired solution to our constrained optimization problem, as shown by

\[ \hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \mu \mathbf{u}(n) e^*(n), \]

which is a duplicate of the update formula previously described in Eq. (6.3), and with it, localized optimality of the LMS algorithm is justified.

**Suboptimality Compared to the Wiener Solution**

In Chapter 4, we demonstrated that the steepest-descent algorithmic representation of the Wiener filter approaches the Wiener solution, \( \mathbf{w}_n \), in the limit as the number of adaptation cycles, \( n \), approaches infinity, provided that the step-size parameter, \( \mu \), satisfies the condition of Eq. (4.22). Moreover, the learning curve approaches the asymptotic value \( J_{\min} \).

In Section 6.4, we will show that, in contrast with the Wiener filter, the corresponding learning curve of the LMS algorithm approaches an asymptotic value \( J(\infty) \) that exceeds \( J_{\min} \) by an amount referred to as the excess mean-square error. It follows, therefore, that compared to the Wiener solution, the LMS algorithm is suboptimal.

In conceptual statistical terms, we may differentiate between these two learning curves as follows:

1. In the method of steepest descent, ensemble averaging is performed before computing the learning curve, as illustrated in Fig. 6.2(a); thereby, perfect information about the environment is contained in the correlation matrix \( \mathbf{R} \) and cross-correlation vector \( \mathbf{p} \), provided that the environment is wide-sense stationary. The learning curve for the method of steepest descent is therefore deterministic in nature.
2. On the other hand, in the LMS case, ensemble averaging is performed after computing the "noisy" learning curves of an ensemble of independently adapted FIR filters, as illustrated in Fig. 6.2(b); the noise is attributed to gradient noise. The learning curve of the LMS is therefore statistical in nature.

This important difference in ensemble averaging is what accounts for the excess mean-square error experienced in the LMS algorithm and therefore its suboptimality compared to the Wiener solution.

6.3 APPLICATIONS

Before proceeding further with a convergence analysis of the LMS algorithm, it is instructive to develop an appreciation for the versatility of this important signal-processing algorithm. We do that by presenting six widely different applications of the LMS algorithm.

Application 1: Canonical Model of the Complex LMS Algorithm

The LMS algorithm described in Eqs. (6.1) through (6.3) is complex in the sense that the input and output data as well as the tap weights are all complex valued. To emphasize the complex nature of the algorithm, henceforth we use the following complex notations:

**Tap-input vector:**

\[ \mathbf{u}(n) = u_r(n) + j u_Q(n). \]  

\[ (6.9) \]
Desired response:

\[ d(n) = d_I(n) + j d_Q(n). \]  \hspace{1cm} (6.10)

Tap-weight vector:

\[ \hat{w}(n) = \hat{w}_I(n) + j \hat{w}_Q(n). \]  \hspace{1cm} (6.11)

FIR filter output:

\[ y(n) = y_I(n) + j y_Q(n). \]  \hspace{1cm} (6.12)

Estimation error:

\[ e(n) = e_I + j e_Q(n). \]  \hspace{1cm} (6.13)

The subscripts \( I \) and \( Q \) denote "in-phase" and "quadrature" components, respectively—that is, real and imaginary parts, respectively. Using these definitions in Eqs. (6.1) through (6.3), expanding terms, and then equating real and imaginary parts, we get

\[ y_I(n) = \hat{w}_I^T(n) u_I(n) - \hat{w}_Q^T(n) u_Q(n), \]  \hspace{1cm} (6.14)

\[ y_Q(n) = \hat{w}_I^T(n) u_Q(n) + \hat{w}_Q^T(n) u_I(n), \]  \hspace{1cm} (6.15)

\[ e_I(n) = d_I(n) - y_I(n), \]  \hspace{1cm} (6.16)

\[ e_Q(n) = d_Q(n) - y_Q(n), \]  \hspace{1cm} (6.17)

\[ \hat{w}_I(n + 1) = \hat{w}_I(n) + \mu [e_I(n) u_I(n) - e_Q(n) u_Q(n)], \]  \hspace{1cm} (6.18)

and

\[ \hat{w}_Q(n + 1) = \hat{w}_Q(n) + \mu [e_I(n) u_Q(n) + e_Q(n) u_I(n)], \]  \hspace{1cm} (6.19)

where the superscript \( T \) denotes transposition. Equations (6.14) through (6.17), defining the error and output signals, are represented by the cross-coupled signal-flow graph shown in Fig. 6.3(a). The updated Eqs. (6.18) and (6.19) are likewise represented by the cross-coupled signal-flow graph shown in Fig. 6.3(b). The combination of this pair of signal-flow graphs constitutes the canonical model of the complex LMS algorithm. This model clearly illustrates that a complex LMS algorithm is equivalent to a set of four real LMS algorithms with cross-coupling between them, hence the computing power of the complex LMS algorithm.

The need for the canonical model of the complex LMS algorithm may arise, for example, in the adaptive equalization of a communication system for the transmission of binary data over a dispersive channel. To facilitate data transmission over the channel, some form of modulation is used, so that the spectral content of the transmitted signal resides inside the passband of the channel. Moreover, for spectral efficiency, modulation techniques such as quadrature phase-shift keying (QPSK) or \( M \)-ary quadrature amplitude modulation (QAM) are used, in which case the baseband version of the channel output assumes a complex form, which is the reason for the complex LMS algorithm. In any event, data transmission through the channel is limited by two factors:

- **Intersymbol interference** (ISI), which is caused mainly by dispersion in the channel.
- **Thermal noise**, which is generated at the channel output (i.e., receiver input).
For bandwidth-limited channels, we typically find that ISI is the chief determining factor in the design of high-data transmission systems. The adaptive equalizer is customarily placed in the receiver. With the channel output as the source of excitation applied to the equalizer, its free parameters are adjusted by means of the complex LMS algorithm to provide an estimate of each symbol transmitted. Provision for the desired response is made locally in the receiver. Specifically, during the training mode, a replica of the desired response is stored in the receiver. Naturally, the generator of this stored reference has to be synchronized with the known training sequence that is transmitted prior to data transmission. A widely used training sequence is a pseudonoise (PN) sequence, which has broadband noiselike properties; in reality, it is a deterministic waveform that repeats periodically. The PN sequence is generated by means of a feedback shift register that consists of a number of consecutive two-state memory stages (flip-flops) regulated
by a single timing clock. A feedback signal consisting of the modulo-two sum of the outputs of the memory stages is applied to the first memory stage of the shift register and thereby prevents it from emptying. Once the training mode is completed, data transmission over the channel begins. To allow the adaptive equalizer to track statistical variations of the channel during data transmission, the equalizer is switched to its decision-directed mode, more on which is said in Chapter 17.

Application 2: Adaptive Deconvolution for Processing of Time-Varying Seismic Data

In exploration seismology, we usually think of a layered model of the earth. In order to collect (i.e., record) seismic data for the purpose of characterizing such a model and thereby unraveling the complexities of the earth's surface, it is customary to use the method of reflection seismology, which involves the following:

- A source of seismic energy (e.g., dynamite or an air gun), which is typically activated on the earth's surface.
• The propagation of seismic waves from the interfaces between the earth's geological layers.
• Picking up and recording the seismic returns (i.e., reflections of seismic waves from the interfaces), which carry information about the subsurface structure of the earth.

An important issue in exploration seismology is the interpretation of seismic returns from the different geological layers of the earth. This interpretation is fundamental to the identification of crusted regions such as rocks, layers of sand, or sedimentary layers. The sedimentary layers are of particular interest because they may contain hydrocarbon reservoirs. The idea of a layered-earth model plays a key role here.

Figure 6.4 depicts an FIR model for a layered earth: the model provides a local parameterization of the propagation (scattering) phenomenon in the earth's subsurface. According to the real-valued model shown in the figure, the input (downgoing) seismic wave \( s(n) \) and the output (upgoing) seismic wave \( u(n) \) are related by the convolution sum (assuming a model of length \( M - 1 \))

\[
u(n) = \sum_{k=0}^{M-1} w_k s(n - k).
\]

where the sequence of tap weights, denoted by \( w_0, w_1, \ldots, w_{M-1} \), represents the spatial mapping of the weighting or impulse response of the medium. The problem to be solved is one of system identification:

Given a seismogram (i.e., a record of seismic returns) denoted by \( u(n) \), estimate the impulse response \( w_n \) of the medium.

This estimation, called seismic deconvolution, removes the effect of convolving \( s(n) \) with \( w_n \). However, the problem is complicated by the fact that the input seismic

![FIR model of layered earth.](image-url)
wave \( s(n) \) is actually unknown. To overcome this difficulty, we may use one of two procedures:

1. **Predictive deconvolution**,\(^2\) which is so called because the procedure relies on linear prediction theory.
2. **Blind deconvolution**, which accounts for valuable phase information contained in the reflection seismogram—information that is ignored in predictive deconvolution.

In this section, we describe predictive convolution; discussion of blind deconvolution is deferred to Chapter 17. In both procedures, the LMS algorithm features prominently. Predictive deconvolution rests on two hypotheses (Robinson & Durrani, 1986):

1. **The feedback hypothesis**, which treats \( w_n \) as the impulse response of an autoregressive (AR) model; the implication of this hypothesis is that the layered-earth model is minimum phase.
2. **The random hypothesis**, according to which the reflectivity function (i.e., the result of the deconvolution) is assumed to have the properties of white noise, at least within certain time gates.

Given the real-valued seismogram \( u(n) \), we may use the real LMS algorithm to solve the predictive deconvolution problem by proceeding as follows (Griffiths et al., 1977):

- An \( M \)-dimensional operator \( \hat{w}(n) \) is used to generate a predicted trace from the data; that is,
  \[
  u(n + \Delta) = \hat{w}^T(n)u(n),
  \]  
  \[
  (6.20)
  \]

  where
  \[
  \hat{w}(n) = [\hat{w}_0(n), \hat{w}_1(n), \ldots, \hat{w}_{M-1}(n)]^T,
  \]
  \[
  u(n) = [u(n), u(n - 1), \ldots, u(n - M + 1)]^T,
  \]

  and \( \Delta \geq 1 \) is the **prediction depth**, or **decorrelation delay**, measured in units of the sampling period.

- The deconvolved trace \( y(n) \) defining the difference between the input and predicted samples is evaluated:
  \[
  y(n) = u(n) - \hat{u}(n).
  \]

- The operator \( \hat{w}(n) \) is updated:
  \[
  \hat{w}(n + 1) = \hat{w}(n) + \mu[u(n + \Delta) - \hat{u}(n + \Delta)]u(n).
  \]
  \[
  (6.21)
  \]

Equations (6.20) and (6.21) constitute the LMS-based adaptive seismic deconvolution algorithm. The adaptation is begun with an initial guess \( \hat{w}(0) \).

\(^2\)A critique of predictive deconvolution is given by Schneider (1978, p. 29):

[A] work horse of statistical wavelet deconvolution for [several] decades has been the predictive deconvolution approach, which assumes the reflectivity function is statistically white and the convolutional wavelet to be minimum phase. To say that this has not been an effective tool is to condemn hundreds of thousands of miles of seismic processing and to deny untold millions of barrels of oil discovered from these data.
Application 3: Instantaneous Frequency Measurement

In this example, we study the use of the LMS algorithm as the basis for estimating the frequency content of a narrowband signal characterized by a rapidly varying power spectrum (Griffiths, 1975). In so doing, we illustrate the linkage between three basic ideas: an autoregressive (AR) model for describing a stochastic process, studied in Chapter 1; a linear predictor for analyzing the process, studied in Chapter 3; and the LMS algorithm for estimating the AR parameters.

By a narrowband signal, we mean a signal whose bandwidth $\Omega$ is small compared with the midband angular frequency $\omega_c$, as illustrated in Fig. 6.5. A frequency-modulated (FM) signal is an example of a narrowband signal, provided that the carrier frequency is high enough. The instantaneous frequency (defined as the derivative of phase with respect to time) of an FM signal varies linearly with the modulating signal. Consider, then, a narrowband process $u(n)$ generated by a time-varying AR model of order $M$, as shown by the difference equation (assuming real data)

$$u(n) = -\sum_{k=1}^{M} a_k(n)u(n-k) + \nu(n),$$  \hfill (6.22)

where the $a_k(n)$ are the time-varying model parameters and $\nu(n)$ is a zero-mean white-noise process of time-varying variance $\sigma^2(n)$. The time-varying AR (power) spectrum of the narrowband process $u(n)$ is given by [see Eq. (3.101)]

$$S_{AR}(\omega; n) = \frac{\sigma^2(n)}{1 + \sum_{k=1}^{M} a_k(n)e^{-j\omega k}}^2, \quad -\pi < \omega \leq \pi.$$  \hfill (6.23)

![Diagram of spectrum with 3 dB point and bandwidth $\Omega$](image)

**FIGURE 6.5** Definition of a narrowband signal in terms of its spectrum.
Note that an AR process whose poles form a cluster near the unit circle in the $z$-plane has the characteristics of a narrowband process.

To estimate the model parameters, we use an adaptive FIR filter employed as a linear predictor of order $M$. Let the tap weights of the predictor be denoted by $\hat{w}_k(n)$, $k = 1, 2, \ldots, M$. The tap weights are adapted continuously as the input signal $u(n)$ is received. In particular, we use the following LMS algorithm for adapting the tap weights:

$$\hat{w}_k(n + 1) = \hat{w}_k(n) + \mu u(n - k)f_M(n), \quad k = 1, 2, \ldots, M. \quad (6.24)$$

In this equation,

$$f_M(n) = u(n) - \sum_{k=1}^{M} \hat{w}_k(n)u(n - k) \quad (6.25)$$

is the prediction error. The tap weights of the adaptive predictor are related to the AR model parameters as follows:

$$-\hat{w}_k(n) = \text{estimate of } a_k(n) \text{ at adaptation cycle } n, \quad \text{for } k = 1, 2, \ldots, M.$$ 

Moreover, the average power of the prediction error $f_M(n)$ provides an estimate of the noise variance $\sigma^2(n)$. Our interest is in locating the frequency of a narrowband signal. Accordingly, in what follows, we ignore the estimation of $\sigma^2(n)$. Specifically, we use only the tap weights of the adaptive predictor to define the time-varying frequency function

$$F(\omega; n) = \frac{1}{1 - \sum_{k=1}^{M} \hat{w}_k(n)e^{-j\omega k}}. \quad (6.26)$$

Given the relationship between $\hat{w}_k(n)$ and $a_k(n)$, we see that the essential difference between the frequency function $F(\omega; n)$ in Eq. (6.26) and the AR power spectrum $S_{\text{AR}}(\omega; n)$ in Eq. (6.23) lies in their numerator scale factors. The numerator of $F(\omega; n)$ is a constant equal to unity, whereas that of $S_{\text{AR}}(\omega; n)$ is a time-varying constant equal to $\sigma^2(n)$. The advantages of $F(\omega; n)$ over $S_{\text{AR}}(\omega; n)$ are twofold: First, the 0/0 indeterminacy inherent in the narrowband spectrum of Eq. (6.23) is replaced by a “computationally tractable” limit of 1/0 in Eq. (6.26); second, the frequency function $F(\omega; n)$ is not affected by amplitude scale changes in the input signal $u(n)$, with the result that the peak value of $F(\omega; n)$ is related directly to the spectral width of the input signal.

We may use the function $F(\omega; n)$ to measure the instantaneous frequency of a frequency-modulated signal $u(n)$, provided that the following assumptions are justified (Griffiths, 1975):

- The adaptive predictor has been in operation sufficiently long, so as to ensure that any transients caused by the initialization of the tap weights have died out.
- The step-size parameter $\mu$ is chosen correctly for the prediction error $f_M(n)$ to be small for all $n$.
- The modulating signal is essentially constant over the sampling range of the adaptive predictor, which extends from adaptation cycle $(n - M)$ to adaptation cycle $(n - 1)$. 
Given the validity of these assumptions, we find that the frequency function $F(\omega; n)$ has a peak at the instantaneous frequency of the input signal $u(n)$. Moreover, the LMS algorithm will track the time variation of the instantaneous frequency.

**Application 4: Adaptive Noise Cancelling Applied to a Sinusoidal Interference**

The traditional method of suppressing a sinusoidal interference corrupting an information-bearing signal is to use a fixed notch filter tuned to the frequency of the interference. To design the filter, we naturally need to know the precise frequency of the interference. But what if the notch is required to be very sharp and the interfering sinusoid is known to drift slowly? Clearly, then, we have a problem that calls for an adaptive solution. One such solution is provided by the use of adaptive noise cancelling, an application that is different from the previous three in that it is not based on a stochastic excitation.

Figure 6.6 shows the block diagram of a dual-input adaptive noise canceller. The primary input supplies an information-bearing signal and a sinusoidal interference that are uncorrelated with each other. The reference input supplies a correlated version of the sinusoidal interference. For the adaptive filter, we may use an FIR filter whose tap weights are adapted by means of the LMS algorithm. The filter uses the reference input to provide (at its output) an estimate of the sinusoidal interfering signal contained in the primary input. Thus, by subtracting the adaptive filter output from the primary input, the effect of the sinusoidal interference is diminished. In particular, an adaptive noise canceller using the LMS algorithm has two important characteristics (Widrow et al., 1976; Glover, 1977):

1. The canceller behaves as an adaptive notch filter whose null point is determined by the angular frequency $\omega_0$ of the sinusoidal interference. Hence, the canceller is tunable, and the tuning frequency moves with $\omega_0$.

2. The notch in the frequency response of the canceller can be made very sharp at precisely the frequency of the sinusoidal interference by choosing a small enough value for the step-size parameter $\mu$.

Thus, unlike the situation with an ordinary notch filter, we have control over the frequency response of the adaptive noise canceller.

![Block diagram of adaptive noise canceller](image-url)
In the application considered here, the input data are assumed to be real valued:

- **Primary input:**
  \[ d(n) = s(n) + A_0 \cos(\omega_0 n + \phi_0), \]  
  \( (6.27) \)

  where \( s(n) \) is an information-bearing signal, \( A_0 \) is the amplitude of the sinusoidal interference, \( \omega_0 \) is the *normalized* angular frequency, and \( \phi_0 \) is the phase.

- **Reference input:**
  \[ u(n) = A \cos(\omega_0 n + \phi), \]  
  \( (6.28) \)

  where the amplitude \( A \) and the phase \( \phi \) are different from those in the primary input, but the angular frequency \( \omega_0 \) is the same.

Using the real form of the LMS algorithm, we describe the tap-weight update by means of the equations

\[ y(n) = \sum_{i=0}^{M-1} \hat{w}_i(n) u(n - i), \]  
\( (6.29) \)

\[ e(n) = d(n) - y(n), \]  
\( (6.30) \)

and

\[ \hat{w}_i(n + 1) = \hat{w}_i(n) + \mu e(n - i) e(n), \quad i = 0, 1, \ldots, M - 1. \]  
\( (6.31) \)

where \( M \) is the length of the FIR filter and the constant \( \mu \) is the step-size parameter. Note that the sampling period in the input data and in all other signals in the LMS algorithm is assumed to be unity for convenience of presentation; as mentioned previously, this practice is indeed followed throughout the book.

With a sinusoidal excitation as the input of interest, we restructure the block diagram of the adaptive noise canceller as in Fig. 6.7(a). According to this new representation, we may lump the sinusoidal input \( u(n) \), the FIR filter, and the weight-update equation of the LMS algorithm into a single (open-loop) system. The adaptive system with input \( e(n) \) and output \( y(n) \) varies with time and cannot be represented by a transfer function. We may get around this difficulty as follows: With \( z = e^{j\omega} \) and \( z_0 = e^{j\omega_0} \), let the adaptive system be excited with \( e(n) = z^n \). Then, the output \( y(n) \) consists of three components: one proportional to \( z^n \), the second proportional to \( z^n(z_0^{2n})^* \), and the third proportional to \( z^n(z_0^{2n}) \). The first component represents a time-invariant system with the transfer function \( G(z) \), which denotes the proportionality factor characterizing that component. The task at hand is to find \( G(z) \).

To do so, we use the detailed signal-flow graph representation of the LMS algorithm depicted in Fig. 6.7(b) (Glover, 1977). In this diagram, we have singled out the \( i \)th tap weight for specific attention. The corresponding value of the tap input is

\[ u(n - i) = A \cos[\omega_0(n - i) + \phi] \]

\[ = \frac{A}{2} [e^{i(\omega_0 n + \phi_i)} + e^{-i(\omega_0 n + \phi_i)}], \]  
\( (6.32) \)
FIGURE 6.7  (a) New representation of adaptive noise canceller. (continued on the next page.)
FIGURE 6.7 (b) Signal-flow graph representation of adaptive noise canceller, singling out the $i$th tap weight for detailed attention.
where
\[ \phi_i = \phi - \omega_0 i. \]

In Fig. 6.7(b), the input \( u(n-i) \) is multiplied by the estimation error \( e(n) \). Hence, taking the z-transform of the product \( u(n-i)e(n) \) and using \( z[\cdot] \) to denote this operation, we obtain
\[ z[u(n-i)e(n)] = \frac{A}{2} e^{j\phi} E(z e^{-j\omega_0}) + \frac{A}{2} e^{-j\phi} E(z e^{j\omega_0}). \]
(6.33)

where \( E(z e^{-j\omega_0}) \) is the z-transform \( E(z) \) of \( e(n) \) rotated counterclockwise around the unit circle through the angle \( \omega_0 \). Similarly, \( E(z e^{j\omega_0}) \) represents a clockwise rotation through \( \omega_0 \).

Next, taking the z-transform of \( \hat{w}(z) \), we get
\[ z\hat{W}(z) = \hat{W}(z) + \mu z[u(n-i)e(n)], \]
(6.34)

where \( \hat{W}(z) \) is the z-transform of \( \hat{w}(n) \). Solving Eq. (6.34) for \( \hat{W}(z) \) and using the z-transform given in Eq. (6.33), we get
\[ \hat{W}(z) = \frac{\mu A}{2} \frac{1}{z-1} [e^{j\phi} E(z e^{-j\omega_0}) + e^{-j\phi} E(z e^{j\omega_0})]. \]
(6.35)

We turn next to Eq. (6.29), which defines the adaptive filter output \( y(n) \). Substituting Eq. (6.32) into Eq. (6.29), we obtain
\[ y(n) = \frac{A}{2} \sum_{i=0}^{M-1} \hat{W}(n)[e^{i(\omega_0 n + \phi_i)} + e^{-j(\omega_0 n + \phi_i)}]. \]

Evaluating the z-transform of \( y(n) \) then yields
\[ Y(z) = \frac{A}{2} \sum_{i=0}^{M-1} [e^{i\phi} \hat{W}(z e^{-j\omega_0}) + e^{i\phi} \hat{W}(z e^{j\omega_0})]. \]
(6.36)

Thus, substituting Eq. (6.35) into Eq. (6.36), we obtain an expression for \( Y(z) \) that consists of the sum of two components (Glover, 1977):

1. A time-invariant component defined by
\[ \frac{\mu A^2}{4} \left( \frac{1}{z e^{-j\omega_0} - 1} + \frac{1}{z e^{j\omega_0} - 1} \right), \]
which is independent of the phase \( \phi_i \) and, therefore, the time index \( i \).

2. A time-varying component that is dependent on the phase \( \phi_i \) and hence on the variation with time \( i \). This second component is scaled in amplitude by the factor
\[ \beta(\omega_0, M) = \frac{\sin(M \omega_0)}{\sin \omega_0}. \]

For a given angular frequency \( \omega_0 \), we assume that the total number of tap weights \( M \) in the FIR filter is large enough to satisfy the following approximation:
\[ \frac{\beta(\omega_0, M)}{M} \approx 0. \]
(6.37)
Accordingly, we may justifiably ignore the time-varying component of the z-transform and so approximate \( Y(z) \) by retaining the time-invariant component only:

\[
Y(z) = \frac{\mu MA^2}{4} E(z) \left( \frac{1}{ze^{-j\omega_0} - 1} + \frac{1}{ze^{j\omega_0} - 1} \right).
\]  

(6.38)

The open-loop transfer function, relating \( y(n) \) to \( e(n) \), is therefore

\[
G(z) = \frac{Y(z)}{E(z)} = \frac{\mu MA^2}{4} \left( \frac{1}{ze^{-j\omega_0} - 1} + \frac{1}{ze^{j\omega_0} - 1} \right)
\]  

(6.39)

\[
= \frac{\mu MA^2}{2} \left( \frac{z \cos \omega_0 - 1}{z^2 - 2z \cos \omega_0 + 1} \right).
\]

The transfer function \( G(z) \) has two complex-conjugate poles on the unit circle at \( z = e^{\pm j\omega_0} \) and a real zero at \( z = 1/\cos \omega_0 \), as illustrated in Fig. 6.8(a). In other words, the adaptive noise canceller has a null point determined by the angular frequency \( \omega_0 \) of the sinusoidal interference, as stated previously. (See Characteristic 1 on page 278.) Indeed, according to Eq. (6.39), we may view \( G(z) \) as a pair of integrators that have been rotated by \( \pm \omega_0 \). In actuality, we see from Fig. 6.7(b) that it is the input that is first shifted in frequency by an amount \( \pm \omega_0 \) due to the first multiplication by the reference sinusoid \( u(n) \), digitally integrated at zero frequency, and then shifted back again by the second multiplication. This overall operation is similar to a well-known technique in communications for obtaining a resonant filter that involves the combined use of two low-pass filters and heterodyning with sine and cosine at the resonant frequency (Wozencraft & Jacobs, 1965; Glover, 1977).

![Figure 6.8](image_url)

**FIGURE 6.8** Approximate pole-zero patterns of (a) the open-loop transfer function \( G(z) \) and (b) the closed-loop transfer function \( H(z) \).
Chapter 6  The Least-Mean-Square (LMS) Algorithm

The model of Fig. 6.7(a) is recognized as a closed-loop feedback system whose transfer function $H(z)$ is related to the open-loop transfer function $G(z)$ via the equation

$$H(z) = \frac{E(z)}{D(z)} = \frac{1}{1 + G(z)} \quad (6.40)$$

where $E(z)$ is the $z$-transform of the system output $e(n)$ and $D(z)$ is the $z$-transform of the system input $d(n)$. Accordingly, substituting Eq. (6.39) into Eq. (6.40), we get the approximate result

$$H(z) \approx \frac{z^2 - 2z \cos \omega_0 + 1}{z^2 - 2(1 - \mu MA^2/4)z \cos \omega_0 + (1 - \mu MA^2/2)} \quad (6.41)$$

Equation (6.41) is the transfer function of a second-order digital notch filter with a notch at the normalized angular frequency $\omega_0$. The zeros of $H(z)$ are at the poles of $G(z)$; that is, they are located on the unit circle at $z = e^{\pm j \omega_0}$. For a small value of the step-size parameter $\mu$ (i.e., a slow adaptation rate), such that

$$\frac{\mu MA^2}{4} \ll 1,$$

we find that the poles of $H(z)$ are located at

$$z \approx \left(1 - \frac{\mu MA^2}{4}\right)e^{\pm j \omega_0} \quad (6.42)$$

In other words, the two poles of $H(z)$ lie inside the unit circle, a radial distance approximately equal to $\mu MA^2/4$ behind the zeros, as indicated in Fig. 6.8(b). The fact that the poles of $H(z)$ lie inside the unit circle means that the adaptive noise canceller is stable, as it should be for practical use in real time.

Figure 6.8(b) also includes the half-power points of $H(z)$. Since the zeros of $H(z)$ lie on the unit circle, the adaptive noise canceller has (in theory) a notch of infinite depth (in dB) at $\omega = \omega_0$. The sharpness of the notch is determined by the closeness of the poles of $H(z)$ to its zeros. The 3-dB bandwidth $B$ is determined by locating the two half-power points on the unit circle that are $\sqrt{2}$ times as far from the poles as they are from the zeros. Using this geometric approach, we find that the 3-dB bandwidth of the adaptive noise canceller is

$$B \approx \frac{\mu MA^2}{2} \text{ radians.} \quad (6.43)$$

Therefore, the smaller we make $\mu$, the smaller the bandwidth $B$ is, and the sharper the notch is. This confirms Characteristic 2 of the adaptive noise canceller that was mentioned on page 278. Its analysis is thereby completed.