Douglas, S.C. “Introduction to Adaptive Filters”
Digital Signal Processing Handbook
Ed. Vijay K. Madisetti and Douglas B. Williams
Boca Raton: CRC Press LLC, 1999

18

Introduction to Adaptive Filters

Scott C. Douglas
University of Utah

18.1 What is an Adaptive Filter?

An *adaptive filter* is a computational device that attempts to model the relationship between two signals in real time in an iterative manner. Adaptive filters are often realized either as a set of program instructions running on an arithmetical processing device such as a microprocessor or DSP chip, or as a set of logic operations implemented in a field-programmable gate array (FPGA) or in a semi-custom or custom VLSI integrated circuit. However, ignoring any errors introduced by numerical precision effects in these implementations, the fundamental operation of an adaptive filter can be characterized independently of the specific physical realization that it takes. For this reason, we shall focus on the mathematical forms of adaptive filters as opposed to their specific realizations in software or hardware. Descriptions of adaptive filters as implemented on DSP chips and on a dedicated integrated circuit can be found in [1, 2, 3], and [4], respectively.

An adaptive filter is defined by four aspects:

1. the *signals* being processed by the filter
2. the *structure* that defines how the output signal of the filter is computed from its input signal
3. the *parameters* within this structure that can be iteratively changed to alter the filter's input-output relationship
4. the *adaptive algorithm* that describes how the parameters are adjusted from one time instant to the next

By choosing a particular adaptive filter structure, one specifies the number and type of parameters that can be adjusted. The adaptive algorithm used to update the parameter values of the system can take on a myriad of forms and is often derived as a form of *optimization procedure* that minimizes an *error criterion* that is useful for the task at hand.

In this section, we present the general adaptive filtering problem and introduce the mathematical notation for representing the form and operation of the adaptive filter. We then discuss several different structures that have been proven to be useful in practical applications. We provide an overview of the many and varied applications in which adaptive filters have been successfully used. Finally, we give a simple derivation of the *least-mean-square (LMS) algorithm*, which is perhaps the most popular method for adjusting the coefficients of an adaptive filter, and we discuss some of this algorithm's properties.

As for the mathematical notation used throughout this section, all quantities are assumed to be real-valued. Scalar and vector quantities shall be indicated by lowercase (e.g., x) and uppercase-bold (e.g., $\mathbf{X}$) letters, respectively. We represent scalar and vector sequences or signals as $x(n)$ and $\mathbf{X}(n)$, respectively, where n denotes the discrete time or discrete spatial index, depending on the application. Matrices and indices of vector and matrix elements shall be understood through the context of the discussion.

18.2 The Adaptive Filtering Problem

Figure 18.1 shows a block diagram in which a sample from a digital *input signal* $x(n)$ is fed into a device, called an *adaptive filter*, that computes a corresponding *output signal* sample $y(n)$ at time n. For the moment, the structure of the adaptive filter is not important, except for the fact that it contains adjustable parameters whose values affect how $y(n)$ is computed. The output signal is compared to a second signal $d(n)$, called the *desired response signal*, by subtracting the two samples at time n. This difference signal, given by

$$e(n) = d(n) - y(n) \,, \tag{18.1}$$

is known as the *error signal*. The error signal is fed into a procedure which alters or *adapts* the parameters of the filter from time n to time $(n+1)$ in a well-defined manner. This process of adaptation is represented by the oblique arrow that pierces the adaptive filter block in the figure. As the time index n is incremented, it is hoped that the output of the adaptive filter becomes a better and better match to the desired response signal through this adaptation process, such that the magnitude of $e(n)$ decreases over time. In this context, what is meant by "better" is specified by the form of the adaptive algorithm used to adjust the parameters of the adaptive filter.

In the adaptive filtering task, adaptation refers to the method by which the parameters of the system are changed from time index n to time index $(n+1)$. The number and types of parameters within this system depend on the computational structure chosen for the system. We now discuss different filter structures that have been proven useful for adaptive filtering tasks.

18.3 Filter Structures

In general, any system with a finite number of parameters that affect how $y(n)$ is computed from $x(n)$ could be used for the adaptive filter in Fig. 18.1. Define the *parameter* or *coefficient vector* $\mathbf{W}(n)$ as

$$\mathbf{W}(n) = [w_0(n)\; w_1(n)\; \cdots\; w_{L-1}(n)]^T \tag{18.2}$$

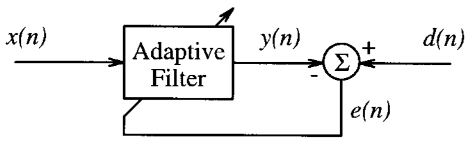

FIGURE 18.1: The general adaptive filtering problem.

where $\{w_i(n)\}$, $0 \leq i \leq L-1$ are the L parameters of the system at time n. With this definition, we could define a general input-output relationship for the adaptive filter as

$$y(n) = f(\mathbf{W}(n),\ y(n-1),\ y(n-2),\ \ldots,\ y(n-N),\ x(n),\ x(n-1),\ \ldots,\ x(n-M+1)), \quad (18.3)$$

where $f(\cdot)$ represents any well-defined linear or nonlinear function and M and N are positive integers. Implicit in this definition is the fact that the filter is *causal*, such that future values of $x(n)$ are not needed to compute $y(n)$. While noncausal filters can be handled in practice by suitably buffering or storing the input signal samples, we do not consider this possibility.

Although (18.3) is the most general description of an adaptive filter structure, we are interested in determining the best *linear relationship* between the input and desired response signals for many problems. This relationship typically takes the form of a *finite-impulse-response* (FIR) or *infinite-impulse-response* (IIR) filter. Figure 18.2 shows the structure of a direct-form FIR filter, also known as a *tapped-delay-line* or *transversal filter*, where z^{-1} denotes the unit delay element and each $w_i(n)$ is a multiplicative gain within the system. In this case, the parameters in $\mathbf{W}(n)$ correspond to the impulse response values of the filter at time n. We can write the output signal $y(n)$ as

$$\begin{aligned} y(n) &= \sum_{i=0}^{L-1} w_i(n) x(n-i) & (18.4)\\ &= \mathbf{W}^T(n)\mathbf{X}(n), & (18.5) \end{aligned}$$

where $\mathbf{X}(n) = [x(n)\,x(n-1)\,\cdots\,x(n-L+1)]^T$ denotes the *input signal vector* and $\cdot^T$ denotes vector transpose. Note that this system requires L multiplies and $L-1$ adds to implement, and these computations are easily performed by a processor or circuit so long as L is not too large and the sampling period for the signals is not too short. It also requires a total of $2L$ memory locations to store the L input signal samples and the L coefficient values, respectively.

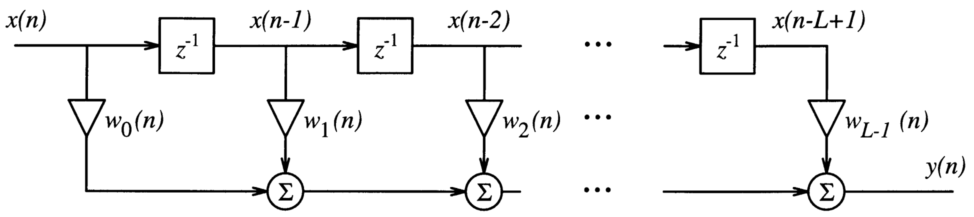

FIGURE 18.2: Structure of an FIR filter.

The structure of a direct-form IIR filter is shown in Fig. 18.3. In this case, the output of the system can be represented mathematically as

$$y(n) = \sum_{i=1}^{N} a_i(n)y(n-i) + \sum_{j=0}^{N} b_j(n)x(n-j) \; , \tag{18.6}$$

although the block diagram does not explicitly represent this system in such a fashion.[1] We could easily write (18.6) using vector notation as

$$y(n) = \mathbf{W}^T(n)\mathbf{U}(n) \; , \tag{18.7}$$

where the $(2N+1)$-dimensional vectors $\mathbf{W}(n)$ and $\mathbf{U}(n)$ are defined as

$$\mathbf{W}(n) = [a_1(n)\, a_2(n) \cdots a_N(n)\, b_0(n)\, b_1(n) \cdots b_N(n)]^T \tag{18.8}$$

$$\mathbf{U}(n) = [y(n-1)\, y(n-2) \cdots y(n-N)\, x(n)\, x(n-1) \cdots x(n-N)]^T , \tag{18.9}$$

respectively. Thus, for purposes of computing the output signal $y(n)$, the IIR structure involves a fixed number of multiplies, adds, and memory locations not unlike the direct-form FIR structure.

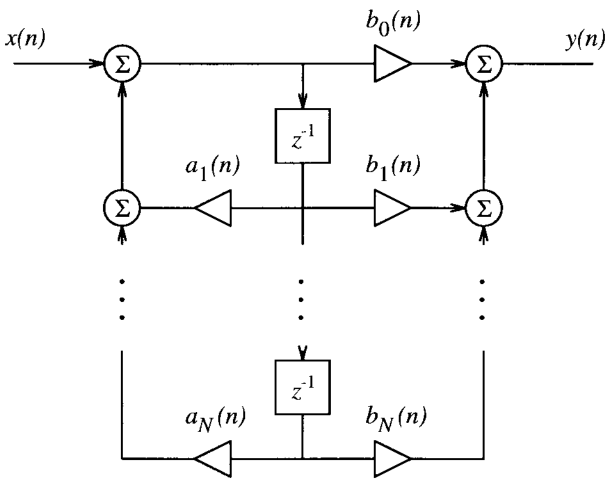

FIGURE 18.3: Structure of an IIR filter.

A third structure that has proven useful for adaptive filtering tasks is the *lattice filter*. A lattice filter is an FIR structure that employs $L-1$ stages of preprocessing to compute a set of auxiliary signals $\{b_i(n)\}$, $0 \leq i \leq L-1$ known as *backward prediction errors*. These signals have the special property that they are *uncorrelated*, and they represent the elements of $\mathbf{X}(n)$ through a *linear transformation*. Thus, the backward prediction errors can be used in place of the delayed input signals in a structure similar to that in Fig. 18.2, and the uncorrelated nature of the prediction errors can provide improved convergence performance of the adaptive filter coefficients with the proper choice of algorithm. Details of the lattice structure and its capabilities are discussed in [6].

[1] The difference between the *direct form II* or *canonical form* structure shown in Fig. 18.3 and the *direct form I* implementation of this system as described by (18.6) is discussed in [5].

A critical issue in the choice of an adaptive filter's structure is its computational complexity. Since the operation of the adaptive filter typically occurs in real time, all of the calculations for the system must occur during one sample time. The structures described above are all useful because $y(n)$ can be computed in a finite amount of time using simple arithmetical operations and finite amounts of memory.

In addition to the linear structures above, one could consider *nonlinear systems* for which the principle of superposition does not hold when the parameter values are fixed. Such systems are useful when the relationship between $d(n)$ and $x(n)$ is not linear in nature. Two such classes of systems are the *Volterra* and *bilinear* filter classes that compute $y(n)$ based on polynomial representations of the input and past output signals. Algorithms for adapting the coefficients of these types of filters are discussed in [7]. In addition, many of the nonlinear models developed in the field of *neural networks*, such as the multilayer perceptron, fit the general form of (18.3), and many of the algorithms used for adjusting the parameters of neural networks are related to the algorithms used for FIR and IIR adaptive filters. For a discussion of neural networks in an engineering context, the reader is referred to [8].

18.4 The Task of an Adaptive Filter

When considering the adaptive filter problem as illustrated in Fig. 18.1 for the first time, a reader is likely to ask, "If we already have the desired response signal, what is the point of trying to match it using an adaptive filter?" In fact, the concept of "matching" $y(n)$ to $d(n)$ with some system obscures the subtlety of the adaptive filtering task. Consider the following issues that pertain to many adaptive filtering problems:

- *In practice, the quantity of interest is not always $d(n)$.* Our desire may be to represent in $y(n)$ a certain component of $d(n)$ that is contained in $x(n)$, or it may be to isolate a component of $d(n)$ within the error $e(n)$ that is *not* contained in $x(n)$. Alternatively, we may be solely interested in the values of the parameters in $\mathbf{W}(n)$ and have no concern about $x(n)$, $y(n)$, or $d(n)$ themselves. Practical examples of each of these scenarios are provided later in this chapter.
- *There are situations in which $d(n)$ is not available at all times.* In such situations, adaptation typically occurs only when $d(n)$ is available. When $d(n)$ is unavailable, we typically use our most-recent parameter estimates to compute $y(n)$ in an attempt to *estimate* the desired response signal $d(n)$.
- *There are real-world situations in which $d(n)$ is* never *available.* In such cases, one can use additional information about the characteristics of a "hypothetical" $d(n)$, such as its predicted statistical behavior or amplitude characteristics, to form suitable estimates of $d(n)$ from the signals available to the adaptive filter. Such methods are collectively called *blind adaptation algorithms.* The fact that such schemes even work is a tribute both to the ingenuity of the developers of the algorithms and to the technological maturity of the adaptive filtering field.

It should also be recognized that the relationship between $x(n)$ and $d(n)$ can vary with time. In such situations, the adaptive filter attempts to alter its parameter values to follow the changes in this relationship as "encoded" by the two sequences $x(n)$ and $d(n)$. This behavior is commonly referred to as *tracking.*

18.5 Applications of Adaptive Filters

Perhaps the most important driving forces behind the developments in adaptive filters throughout their history have been the wide range of applications in which such systems can be used. We now discuss the forms of these applications in terms of more-general problem classes that describe the assumed relationship between $d(n)$ and $x(n)$. Our discussion illustrates the key issues in selecting an adaptive filter for a particular task. Extensive details concerning the specific issues and problems associated with each problem genre can be found in the references at the end of this chapter.

18.5.1 System Identification

Consider Fig. 18.4, which shows the general problem of *system identification*. In this diagram, the system enclosed by dashed lines is a "black box," meaning that the quantities inside are not observable from the outside. Inside this box is (1) an unknown system which represents a general input-output relationship and (2) the signal $\eta(n)$, called the *observation noise signal* because it corrupts the observations of the signal at the output of the unknown system.

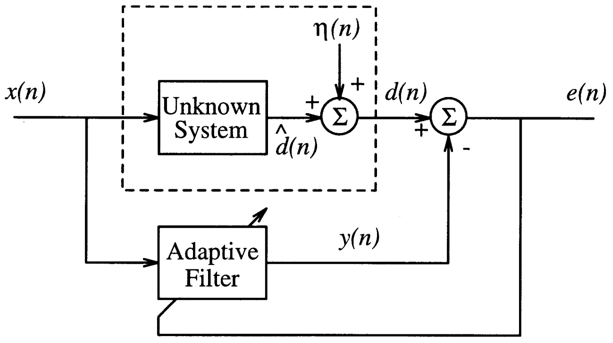

FIGURE 18.4: System identification.

Let $\widehat{d}(n)$ represent the output of the unknown system with $x(n)$ as its input. Then, the desired response signal in this model is

$$d(n) = \widehat{d}(n) + \eta(n)\ . \tag{18.10}$$

Here, the task of the adaptive filter is to accurately represent the signal $\widehat{d}(n)$ at its output. If $y(n) = \widehat{d}(n)$, then the adaptive filter has accurately modeled or identified the portion of the unknown system that is driven by $x(n)$.

Since the model typically chosen for the adaptive filter is a linear filter, the practical goal of the adaptive filter is to determine the best linear model that describes the input-output relationship of the unknown system. Such a procedure makes the most sense when the unknown system is also a linear model of the same structure as the adaptive filter, as it is possible that $y(n) = \widehat{d}(n)$ for some set of adaptive filter parameters. For ease of discussion, let the unknown system and the adaptive filter both be FIR filters, such that

$$d(n) = \mathbf{W}_{opt}^T(n)\mathbf{X}(n) + \eta(n)\ , \tag{18.11}$$

where $\mathbf{W}_{opt}(n)$ is an optimum set of filter coefficients for the unknown system at time n. In this problem formulation, the ideal adaptation procedure would adjust $\mathbf{W}(n)$ such that $\mathbf{W}(n) = \mathbf{W}_{opt}(n)$

as $n \to \infty$. In practice, the adaptive filter can only adjust $\mathbf{W}(n)$ such that $y(n)$ closely approximates $\widehat{d}(n)$ over time.

The system identification task is at the heart of numerous adaptive filtering applications. We list several of these applications here.

Channel Identification

In communication systems, useful information is transmitted from one point to another across a medium such as an electrical wire, an optical fiber, or a wireless radio link. Nonidealities of the transmission medium or *channel* distort the fidelity of the transmitted signals, making the deciphering of the received information difficult. In cases where the effects of the distortion can be modeled as a linear filter, the resulting "smearing" of the transmitted symbols is known as *inter-symbol interference* (ISI). In such cases, an adaptive filter can be used to model the effects of the channel ISI for purposes of deciphering the received information in an optimal manner. In this problem scenario, the transmitter sends to the receiver a sample sequence $x(n)$ that is known to both the transmitter and receiver. The receiver then attempts to model the received signal $d(n)$ using an adaptive filter whose input is the known transmitted sequence $x(n)$. After a suitable period of adaptation, the parameters of the adaptive filter in $\mathbf{W}(n)$ are fixed and then used in a procedure to decode future signals transmitted across the channel.

Channel identification is typically employed when the fidelity of the transmitted channel is severely compromised or when simpler techniques for sequence detection cannot be used. Techniques for detecting digital signals in communication systems can be found in [9].

Plant Identification

In many control tasks, knowledge of the transfer function of a linear plant is required by the physical controller so that a suitable control signal can be calculated and applied. In such cases, we can characterize the transfer function of the plant by exciting it with a known signal $x(n)$ and then attempting to match the output of the plant $d(n)$ with a linear adaptive filter. After a suitable period of adaptation, the system has been adequately modeled, and the resulting adaptive filter coefficients in $\mathbf{W}(n)$ can be used in a control scheme to enable the overall closed-loop system to behave in the desired manner.

In certain scenarios, continuous updates of the plant transfer function estimate provided by $\mathbf{W}(n)$ are needed to allow the controller to function properly. A discussion of these *adaptive control* schemes and the subtle issues in their use is given in [10, 11].

Echo Cancellation for Long-Distance Transmission

In voice communication across telephone networks, the existence of junction boxes called *hybrids* near either end of the network link hampers the ability of the system to cleanly transmit voice signals. Each hybrid allows voices that are transmitted via separate lines or channels across a long-distance network to be carried locally on a single telephone line, thus lowering the wiring costs of the local network. However, when small impedance mismatches between the long distance lines and the hybrid junctions occur, these hybrids can reflect the transmitted signals back to their sources, and the long transmission times of the long-distance network—about 0.3 s for a trans-oceanic call via a satellite link—turn these reflections into a noticeable echo that makes the understanding of conversation difficult for both callers. The traditional solution to this problem prior to the advent of the adaptive filtering solution was to introduce significant loss into the long-distance network so that echoes would decay to an acceptable level before they became perceptible to the callers. Unfortunately, this solution also reduces the transmission quality of the telephone link and makes the task of connecting long distance calls more difficult.

An adaptive filter can be used to cancel the echoes caused by the hybrids in this situation. Adaptive filters are employed at each of the two hybrids within the network. The input $x(n)$ to each adaptive filter is the speech signal being received prior to the hybrid junction, and the desired response signal $d(n)$ is the signal being sent out from the hybrid across the long-distance connection. The adaptive filter attempts to model the transmission characteristics of the hybrid junction as well as any echoes that appear across the long-distance portion of the network. When the system is properly designed, the error signal $e(n)$ consists almost totally of the local talker's speech signal, which is then transmitted over the network. Such systems were first proposed in the mid-1960s [12] and are commonly used today. For more details on this application, see [13, 14].

Acoustic Echo Cancellation

A related problem to echo cancellation for telephone transmission systems is that of acoustic echo cancellation for conference-style speakerphones. When using a speakerphone, a caller would like to turn up the amplifier gains of both the microphone and the audio loudspeaker in order to transmit and hear the voice signals more clearly. However, the feedback path from the device's loudspeaker to its input microphone causes a distinctive *howling* sound if these gains are too high. In this case, the culprit is the room's response to the voice signal being broadcast by the speaker; in effect, the room acts as an extremely poor hybrid junction, in analogy with the echo cancellation task discussed previously. A simple solution to this problem is to only allow one person to speak at a time, a form of operation called *half-duplex transmission.* However, studies have indicated that half-duplex transmission causes problems with normal conversations, as people typically overlap their phrases with others when conversing.

To maintain *full-duplex transmission,* an acoustic echo canceller is employed in the speakerphone to model the acoustic transmission path from the speaker to the microphone. The input signal $x(n)$ to the acoustic echo canceller is the signal being sent to the speaker, and the desired response signal $d(n)$ is measured at the microphone on the device. Adaptation of the system occurs continually throughout a telephone call to model any physical changes in the room acoustics. Such devices are readily available in the marketplace today. In addition, similar technology can and is used to remove the echo that occurs through the combined radio/room/telephone transmission path when one places a call to a radio or television talk show. Details of the acoustic echo cancellation problem can be found in [14].

Adaptive Noise Cancelling

When collecting measurements of certain signals or processes, physical constraints often limit our ability to cleanly measure the quantities of interest. Typically, a signal of interest is linearly mixed with other extraneous noises in the measurement process, and these extraneous noises introduce unacceptable errors in the measurements. However, if a linearly related *reference* version of any one of the extraneous noises can be cleanly sensed at some other physical location in the system, an adaptive filter can be used to determine the relationship between the noise reference $x(n)$ and the component of this noise that is contained in the measured signal $d(n)$. After adaptively subtracting out this component, what remains in $e(n)$ is the signal of interest. If several extraneous noises corrupt the measurement of interest, several adaptive filters can be used in parallel as long as suitable noise reference signals are available within the system.

Adaptive noise cancelling has been used for several applications. One of the first was a medical application that enabled the electroencephalogram (EEG) of the fetal heartbeat of an unborn child to be cleanly extracted from the much-stronger interfering EEG of the maternal heartbeat signal. Details of this application as well as several others are described in the seminal paper by Widrow and his colleagues [15].

18.5.2 Inverse Modeling

We now consider the general problem of *inverse modeling*, as shown in Fig. 18.5. In this diagram, a *source signal* $s(n)$ is fed into an unknown system that produces the input signal $x(n)$ for the adaptive filter. The output of the adaptive filter is subtracted from a desired response signal that is a delayed version of the source signal, such that

$$d(n) = s(n - \Delta) , \tag{18.12}$$

where Δ is a positive integer value. The goal of the adaptive filter is to adjust its characteristics such that the output signal is an accurate representation of the delayed source signal.

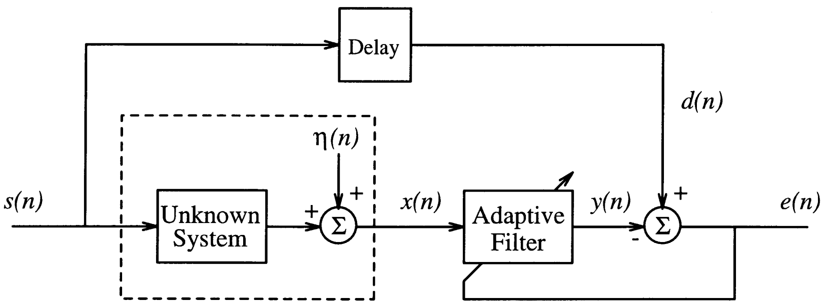

FIGURE 18.5: Inverse modeling.

The inverse modeling task characterizes several adaptive filtering applications, two of which are now described.

Channel Equalization

Channel equalization is an alternative to the technique of channel identification described previously for the decoding of transmitted signals across nonideal communication channels. In both cases, the transmitter sends a sequence $s(n)$ that is known to both the transmitter and receiver. However, in equalization, the received signal is used as the input signal $x(n)$ to an adaptive filter, which adjusts its characteristics so that its output closely matches a delayed version $s(n - \Delta)$ of the known transmitted signal. After a suitable adaptation period, the coefficients of the system either are fixed and used to decode future transmitted messages or are adapted using a crude estimate of the desired response signal that is computed from $y(n)$. This latter mode of operation is known as *decision-directed adaptation*.

Channel equalization was one of the first applications of adaptive filters and is described in the pioneering work of Lucky [16]. Today, it remains as one of the most popular uses of an adaptive filter. Practically every computer telephone modem transmitting at rates of 9600 *baud* (bits per second) or greater contains an adaptive equalizer. Adaptive equalization is also useful for wireless communication systems. Qureshi [17] provides a tutorial on adaptive equalization. A related problem to equalization is *deconvolution*, a problem that appears in the context of geophysical exploration [18]. Equalization is closely related to *linear prediction*, a topic that we shall discuss shortly.

Inverse Plant Modeling

In many control tasks, the frequency and phase characteristics of the plant hamper the convergence behavior and stability of the control system. We can use a system of the form in Fig. 18.5 to compensate for the nonideal characteristics of the plant and as a method for adaptive control. In this case, the signal $s(n)$ is sent at the output of the controller, and the signal $x(n)$ is the signal measured at the output of the plant. The coefficients of the adaptive filter are then adjusted so that the cascade of the plant and adaptive filter can be nearly represented by the pure delay $z^{-\Delta}$. Details of the adaptive algorithms as applied to control tasks in this fashion can be found in [11].

18.5.3 Linear Prediction

A third type of adaptive filtering task is shown in Fig. 18.6. In this system, the input signal $x(n)$ is derived from the desired response signal as

$$x(n) = d(n - \Delta) , \tag{18.13}$$

where Δ is an integer value of delay. In effect, the input signal serves as the desired response signal, and for this reason it is always available. In such cases, the linear adaptive filter attempts to predict future values of the input signal using past samples, giving rise to the name *linear prediction* for this task.

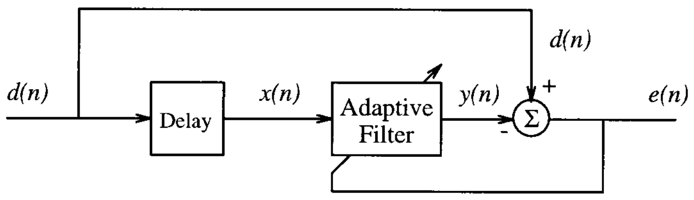

FIGURE 18.6: Linear prediction.

If an estimate of the signal $x(n + \Delta)$ at time n is desired, a copy of the adaptive filter whose input is the current sample $x(n)$ can be employed to compute this quantity. However, linear prediction has a number of uses besides the obvious application of forecasting future events, as described in the following two applications.

Linear Predictive Coding

When transmitting digitized versions of real-world signals such as speech or images, the temporal correlation of the signals is a form of redundancy that can be exploited to code the waveform in a smaller number of bits than are needed for its original representation. In these cases, a linear predictor can be used to model the signal correlations for a short block of data in such a way as to reduce the number of bits needed to represent the signal waveform. Then, essential information about the signal model is transmitted along with the coefficients of the adaptive filter for the given data block. Once received, the signal is synthesized using the filter coefficients and the additional signal information provided for the given block of data.

When applied to speech signals, this method of signal encoding enables the transmission of understandable speech at only 2.4 kb/s, although the reconstructed speech has a distinctly synthetic quality. Predictive coding can be combined with a quantizer to enable higher-quality speech encoding at higher data rates using an *adaptive differential pulse-code modulation* (ADPCM) scheme. In both of these methods, the lattice filter structure plays an important role because of the way in which it parameterizes the physical nature of the vocal tract. Details about the role of the lattice filter in the linear prediction task can be found in [19].

Adaptive Line Enhancement

In some situations, the desired response signal $d(n)$ consists of a sum of a broadband signal and a nearly periodic signal, and it is desired to separate these two signals without specific knowledge about the signals (such as the fundamental frequency of the periodic component).

In these situations, an adaptive filter configured as in Fig. 18.6 can be used. For this application, the delay Δ is chosen to be large enough such that the broadband component in $x(n)$ is uncorrelated with the broadband component in $x(n - \Delta)$. In this case, the broadband signal cannot be removed by the adaptive filter through its operation, and it remains in the error signal $e(n)$ after a suitable period of adaptation. The adaptive filter's output $y(n)$ converges to the narrowband component, which is easily predicted given past samples. The name line enhancement arises because periodic signals are characterized by lines in their frequency spectra, and these spectral lines are enhanced at the output of the adaptive filter.

For a discussion of the adaptive line enhancement task using LMS adaptive filters, the reader is referred to [20].

18.5.4 Feedforward Control

Another problem area combines elements of both the inverse modeling and system identification tasks and typifies the types of problems encountered in the area of adaptive control known as *feedforward control.* Figure 18.7 shows the block diagram for this system, in which the output of the adaptive filter passes through a plant before it is subtracted from the desired response to form the error signal. The plant hampers the operation of the adaptive filter by changing the amplitude and phase characteristics of the adaptive filter's output signal as represented in $e(n)$. Thus, knowledge of the plant is generally required in order to adapt the parameters of the filter properly.

An application that fits this particular problem formulation is *active noise control,* in which unwanted sound energy propagates in air or a fluid into a physical region in space. In such cases, an electroacoustic system employing microphones, speakers, and one or more adaptive filters can be used to create a secondary sound field that interferes with the unwanted sound, reducing its level in the region via destructive interference. Similar techniques can be used to reduce vibrations in solid media. Details of useful algorithms for the active noise and vibration control tasks can be found in [21, 22].

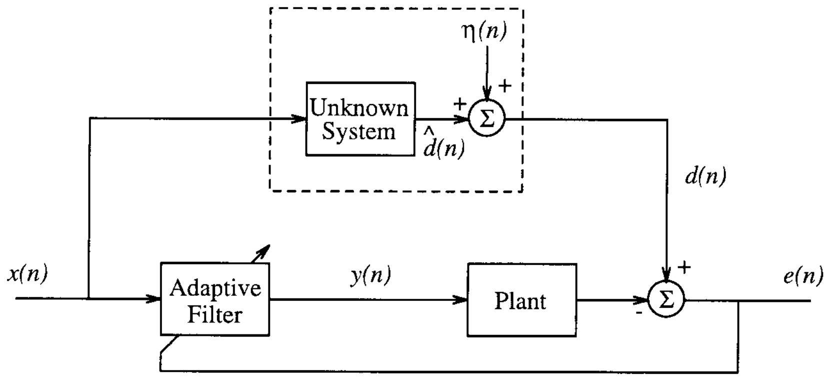

FIGURE 18.7: Feedforward control.

18.6 Gradient-Based Adaptive Algorithms

An adaptive algorithm is a procedure for adjusting the parameters of an adaptive filter to minimize a cost function chosen for the task at hand. In this section, we describe the general form of many adaptive FIR filtering algorithms and present a simple derivation of the LMS adaptive algorithm. In our discussion, we only consider an adaptive FIR filter structure, such that the output signal $y(n)$ is given by (18.5). Such systems are currently more popular than adaptive IIR filters because (1) the input-output stability of the FIR filter structure is guaranteed for any set of fixed coefficients, and (2) the algorithms for adjusting the coefficients of FIR filters are more simple in general than those for adjusting the coefficients of IIR filters.

18.6.1 General Form of Adaptive FIR Algorithms

The general form of an adaptive FIR filtering algorithm is

$$\mathbf{W}(n+1) = \mathbf{W}(n) + \mu(n)\mathbf{G}(e(n),\ \mathbf{X}(n),\ \mathbf{\Phi}(n)), \tag{18.14}$$

where $\mathbf{G}(\cdot)$ is a particular vector-valued nonlinear function, $\mu(n)$ is a *step size* parameter, $e(n)$ and $\mathbf{X}(n)$ are the error signal and input signal vector, respectively, and $\mathbf{\Phi}(n)$ is a vector of states that store pertinent information about the characteristics of the input and error signals and/or the coefficients at previous time instants. In the simplest algorithms, $\mathbf{\Phi}(n)$ is not used, and the only information needed to adjust the coefficients at time n are the error signal, input signal vector, and step size.

The step size is so called because it determines the magnitude of the change or "step" that is taken by the algorithm in iteratively determining a useful coefficient vector. Much research effort has been spent characterizing the role that $\mu(n)$ plays in the performance of adaptive filters in terms of the statistical or frequency characteristics of the input and desired response signals. Often, success or failure of an adaptive filtering application depends on how the value of $\mu(n)$ is chosen or calculated to obtain the best performance from the adaptive filter. The issue of choosing $\mu(n)$ for both stable and accurate convergence of the LMS adaptive filter is addressed in Chapter 19 of this Handbook.

18.6.2 The Mean-Squared Error Cost Function

The form of $\mathbf{G}(\cdot)$ in (18.14) depends on the cost function chosen for the given adaptive filtering task. We now consider one particular cost function that yields a popular adaptive algorithm.

Define the *mean-squared error* (MSE) cost function as

$$\begin{aligned} J_{MSE}(n) &= \frac{1}{2}\int_{-\infty}^{\infty} e^2(n)p_n(e(n))de(n) && (18.15)\\ &= \frac{1}{2}E\{e^2(n)\}\ , && (18.16)\end{aligned}$$

where $p_n(e)$ represents the probability density function of the error at time n and $E\{\cdot\}$ is shorthand for the *expectation integral* on the right-hand side of (18.15). The MSE cost function is useful for adaptive FIR filters because

- $J_{MSE}(n)$ has a well-defined minimum with respect to the parameters in $\mathbf{W}(n)$;
- the coefficient values obtained at this minimum are the ones that minimize the power in the error signal $e(n)$, indicating that $y(n)$ has approached $d(n)$; and

- $J_{MSE}(n)$ is a smooth function of each of the parameters in $\mathbf{W}(n)$, such that it is differentiable with respect to each of the parameters in $\mathbf{W}(n)$.

The third point is important in that it enables us to determine both the optimum coefficient values given knowledge of the statistics of $d(n)$ and $x(n)$ as well as a simple iterative procedure for adjusting the parameters of an FIR filter.

18.6.3 The Wiener Solution

For the FIR filter structure, the coefficient values in $\mathbf{W}(n)$ that minimize $J_{MSE}(n)$ are well-defined if the statistics of the input and desired response signals are known. The formulation of this problem for continuous-time signals and the resulting solution was first derived by Wiener [23]. Hence, this optimum coefficient vector $\mathbf{W}_{MSE}(n)$ is often called the *Wiener solution* to the adaptive filtering problem. The extension of Wiener's analysis to the discrete-time case is attributed to Levinson [24].

To determine $\mathbf{W}_{MSE}(n)$, we note that the function $J_{MSE}(n)$ in (18.16) is quadratic in the parameters $\{w_i(n)\}$, and the function is also differentiable. Thus, we can use a result from optimization theory that states that the derivatives of a smooth cost function with respect to each of the parameters is zero at a minimizing point on the cost function error surface. Thus, $\mathbf{W}_{MSE}(n)$ can be found from the solution to the system of equations

$$\frac{\partial J_{MSE}(n)}{\partial w_i(n)} = 0, \qquad 0 \le i \le L-1. \tag{18.17}$$

Taking derivatives of $J_{MSE}(n)$ in (18.16) and noting that $e(n)$ and $y(n)$ are given by (18.1) and (18.5), respectively, we obtain

$$\begin{aligned}\frac{\partial J_{MSE}(n)}{\partial w_i(n)} &= E\left\{e(n)\frac{\partial e(n)}{\partial w_i(n)}\right\} && (18.18)\\ &= -E\left\{e(n)\frac{\partial y(n)}{\partial w_i(n)}\right\} && (18.19)\\ &= -E\{e(n)x(n-i)\} && (18.20)\\ &= -\left(E\{d(n)x(n-i)\} - \sum_{j=0}^{L-1} E\{x(n-i)x(n-j)\}w_j(n)\right). && (18.21)\end{aligned}$$

where we have used the definitions of $e(n)$ and of $y(n)$ for the FIR filter structure in (18.1) and (18.5), respectively, to expand the last result in (18.21).

By defining the matrix $\mathbf{R}_{\mathrm{xx}}(n)$ and vector $\mathbf{P}_{\mathrm{dx}}(n)$ as

$$\mathbf{R}_{\mathrm{xx}} = E\{\mathbf{X}(n)\mathbf{X}^T(n)\} \text{ and } \mathbf{P}_{\mathrm{dx}}(n) = E\{d(n)\mathbf{X}(n)\}, \tag{18.22}$$

respectively, we can combine (18.17) and (18.21) to obtain the system of equations in vector form as

$$\mathbf{R}_{\mathrm{xx}}(n)\mathbf{W}_{MSE}(n) - \mathbf{P}_{\mathrm{dx}}(n) = \mathbf{0}, \tag{18.23}$$

where $\mathbf{0}$ is the zero vector. Thus, so long as the matrix $\mathbf{R}_{\mathrm{xx}}(n)$ is invertible, the optimum Wiener solution vector for this problem is

$$\mathbf{W}_{MSE}(n) = \mathbf{R}_{\mathrm{xx}}^{-1}(n)\mathbf{P}_{\mathrm{dx}}(n). \tag{18.24}$$

18.6.4 The Method of Steepest Descent

The method of steepest descent is a celebrated optimization procedure for minimizing the value of a cost function $J(n)$ with respect to a set of adjustable parameters $\mathbf{W}(n)$. This procedure adjusts each parameter of the system according to

$$w_i(n+1) = w_i(n) - \mu(n)\frac{\partial J(n)}{\partial w_i(n)} . \tag{18.25}$$

In other words, the ith parameter of the system is altered according to the derivative of the cost function with respect to the ith parameter. Collecting these equations in vector form, we have

$$\mathbf{W}(n+1) = \mathbf{W}(n) - \mu(n)\frac{\partial J(n)}{\partial \mathbf{W}(n)} , \tag{18.26}$$

where $\partial J(n)/\partial \mathbf{W}(n)$ is a vector of derivatives $\partial J(n)/\partial w_i(n)$.

For an FIR adaptive filter that minimizes the MSE cost function, we can use the result in (18.21) to explicitly give the form of the steepest descent procedure in this problem. Substituting these results into (18.25) yields the update equation for $\mathbf{W}(n)$ as

$$\mathbf{W}(n+1) = \mathbf{W}(n) + \mu(n)(\mathbf{P}_{\mathrm{d}\mathbf{X}}(n) - \mathbf{R}_{\mathbf{XX}}(n)\mathbf{W}(n)) . \tag{18.27}$$

However, this steepest descent procedure depends on the statistical quantities $E\{d(n)x(n-i)\}$ and $E\{x(n-i)x(n-j)\}$ contained in $\mathbf{P}_{\mathrm{d}\mathbf{X}}(n)$ and $\mathbf{R}_{\mathbf{XX}}(n)$, respectively. In practice, we only have measurements of both $d(n)$ and $x(n)$ to be used within the adaptation procedure. While suitable estimates of the statistical quantities needed for (18.27) could be determined from the signals $x(n)$ and $d(n)$, we instead develop an approximate version of the method of steepest descent that depends on the signal values themselves. This procedure is known as the *LMS algorithm*.

18.6.5 The LMS Algorithm

The cost function $J(n)$ chosen for the steepest descent algorithm of (18.25) determines the coefficient solution obtained by the adaptive filter. If the MSE cost function in (18.16) is chosen, the resulting algorithm depends on the statistics of $x(n)$ and $d(n)$ because of the expectation operation that defines this cost function. Since we typically only have measurements of $d(n)$ and of $x(n)$ available to us, we substitute an alternative cost function that depends only on these measurements. One such cost function is the least-squares cost function given by

$$J_{LS}(n) = \sum_{k=0}^{n} \alpha(k)(d(k) - \mathbf{W}^T(n)\mathbf{X}(k))^2 . \tag{18.28}$$

where $\alpha(n)$ is a suitable weighting sequence for the terms within the summation. This cost function, however, is complicated by the fact that it requires numerous computations to calculate its value as well as its derivatives with respect to each $w_i(n)$, although efficient recursive methods for its minimization can be developed. See Chapter 21 for more details on these methods.

Alternatively, we can propose the simplified cost function $J_{LMS}(n)$ given by

$$J_{LMS}(n) = \frac{1}{2}e^2(n) . \tag{18.29}$$

This cost function can be thought of as an *instantaneous estimate* of the MSE cost function, as $J_{MSE}(n) = E\{J_{LMS}(n)\}$. Although it might not appear to be useful, the resulting algorithm obtained when $J_{LMS}(n)$ is used for $J(n)$ in (18.25) is extremely useful for practical applications. Taking derivatives of $J_{LMS}(n)$ with respect to the elements of $\mathbf{W}(n)$ and substituting the result into (18.25), we obtain the LMS adaptive algorithm given by

$$\mathbf{W}(n+1) = \mathbf{W}(n) + \mu(n)e(n)\mathbf{X}(n) \ . \tag{18.30}$$

Note that this algorithm is of the general form in (18.14). It also requires only multiplications and additions to implement. In fact, the number and type of operations needed for the LMS algorithm is nearly the same as that of the FIR filter structure with fixed coefficient values, which is one of the reasons for the algorithm's popularity.

The behavior of the LMS algorithm has been widely studied, and numerous results concerning its adaptation characteristics under different situations have been developed. For discussions of some of these results, the reader is referred to Chapters 19 and 20 in this Handbook. For now, we indicate its useful behavior by noting that the solution obtained by the LMS algorithm near its convergent point is related to the Wiener solution. In fact, analyses of the LMS algorithm under certain statistical assumptions about the input and desired response signals show that

$$\lim_{n\to\infty} E\{\mathbf{W}(n)\} = \mathbf{W}_{MSE} \ , \tag{18.31}$$

when the Wiener solution $\mathbf{W}_{MSE}(n)$ is a fixed vector. Moreover, the average behavior of the LMS algorithm is quite similar to that of the steepest descent algorithm in (18.27) that depends explicitly on the statistics of the input and desired response signals. In effect, the iterative nature of the LMS coefficient updates is a form of time-averaging that smooths the errors in the instantaneous gradient calculations to obtain a more reasonable estimate of the true gradient.

18.6.6 Other Stochastic Gradient Algorithms

The LMS algorithm is but one of an entire family of algorithms that are based on instantaneous approximations to steepest descent procedures. Such algorithms are known as *stochastic gradient algorithms* because they use a stochastic version of the gradient of a particular cost function's error surface to adjust the parameters of the filter. As an example, we consider the cost function

$$J_{SA}(n) = |e(n)| \ , \tag{18.32}$$

where $|\cdot|$ denotes absolute value. Like $J_{LMS}(n)$, this cost function also has a unique minimum at $e(n) = 0$, and it is differentiable everywhere except at $e(n) = 0$. Moreover, it is the instantaneous value of the *mean absolute error* cost function $J_{MAE}(n) = E\{J_{SA}(n)\}$. Taking derivatives of $J_{SA}(n)$ with respect to the coefficients $\{w_i(n)\}$ and substituting the results into (18.25) yields the *sign-error* algorithm as[2]

$$\mathbf{W}(n+1) = \mathbf{W}(n) + \mu(n)\mathrm{sgn}(e(n))\mathbf{X}(n) \ , \tag{18.33}$$

where

$$\mathrm{sgn}(e) = \begin{cases} 1 & \text{if } e > 0 \\ 0 & \text{if } e = 0 \\ -1 & \text{if } e < 0 \end{cases} \ . \tag{18.34}$$

This algorithm is also of the general form in (18.14).

The sign error algorithm is a useful adaptive filtering procedure because the terms $\mathrm{sgn}(e(n))x(n-i)$ can be computed easily in dedicated digital hardware. Its convergence properties differ from those of the LMS algorithm, however. Discussions of this and other algorithms based on non-MSE criteria can be found in [25].

[2]Here, we have specified $\partial|e|/\partial e = 0$ for $e = 0$, although the derivative of this function does not exist at this point.

18.6.7 Finite-Precision Effects and Other Implementation Issues

In all digital hardware and software implementations of the LMS algorithm in (18.30), the quantities $e(n)$, $d(n)$, and $\{x(n-i)\}$ are represented by finite-precision quantities with a certain number of bits. Small numerical errors are introduced in each of the calculations within the coefficient updates in these situations. The effects of these numerical errors are usually less severe in systems that employ *floating-point arithmetic*, in which all numerical values are represented by both a mantissa and exponent, as compared to systems that employ *fixed-point arithmetic*, in which a mantissa-only numerical representation is used. The effects of the numerical errors introduced in these cases can be characterized; see [26] for a discussion of these issues.

While knowledge of the numerical effects of finite-precision arithmetic are necessary for obtaining the best performance from the LMS adaptive filter, it can be generally stated that the LMS adaptive filter performs robustly in the presence of these numerical errors. In fact, the apparent robustness of the LMS adaptive filter has led to the development of approximate implementations of (18.30) that are more easily implemented in dedicated hardware. The general form of these implementations is

$$w_i(n+1) = w_i(n) + \mu(n) g_1(e(n)) g_2(x(n-i)) \ , \tag{18.35}$$

where $g_1(\cdot)$ and $g_2(\cdot)$ are odd-symmetric nonlinearities that are chosen to simplify the implementation of the system. Some of the algorithms described by (18.35) include the *sign-data* $\{g_1(e) = e$, $g_2(x) = \mathrm{sgn}(x)\}$, *sign-sign* or *zero-forcing* $\{g_1(e) = \mathrm{sgn}(e)$, $g_2(x) = \mathrm{sgn}(x)\}$, and *power-of-two quantized* algorithms, as well as the sign error algorithm introduced previously. A presentation and comparative analysis of the performance of many of these algorithms can be found in [27].

18.6.8 System Identification Example

We now illustrate the actual behavior of the LMS adaptive filter through a system identification example in which the impulse response of a small audio loudspeaker in a room is estimated. A Gaussian-distributed signal with a flat frequency spectrum over the usable frequency range of the loudspeaker is generated and sent through an audio amplifier to the loudspeaker. This same Gaussian signal is sent to a 16-bit analog-to-digital (A/D) converter which samples it at an 8 kHz rate. The sound produced by the loudspeaker propagates to a microphone located several feet away from the loudspeaker, where it is collected and digitized by a second A/D converter also sampling at an 8 kHz rate. Both signals are stored to a computer file for subsequent processing and analysis. The goal of the analysis is to determine the combined impulse response of the loudspeaker/room/microphone sound propagation path. Such information is useful if the loudspeaker and microphone are to be used in the active noise control task described previously, and the general task also resembles that of acoustic echo cancellation for speakerphones.

We process these signals using a computer program that implements the LMS adaptive filter within the MATLAB[3] signal manipulation environment. In this case, we have normalized the powers of both the Gaussian input signal and desired response signal collected at the microphone to unity, and we have highpass-filtered the microphone signal using a filter with transfer function $H(z) = (1 - z^{-1})/(1 - 0.95z^{-1})$ to remove any DC offset in this signal. For this task, we have chosen an $L = 100$-coefficient FIR filter adapted using the LMS algorithm in (18.30) with a fixed step size of $\mu = 0.0005$ to obtain an accurate estimate of the impulse response of the loudspeaker and room. Figure 18.8 shows the convergence of the error signal in this situation. After about 4000 samples (0.5 s), the error signal has been reduced to a power that is about 1/15 (-12 dB) below that of the

[3]MATLAB is a registered trademark of The MathWorks, Newton, MA.

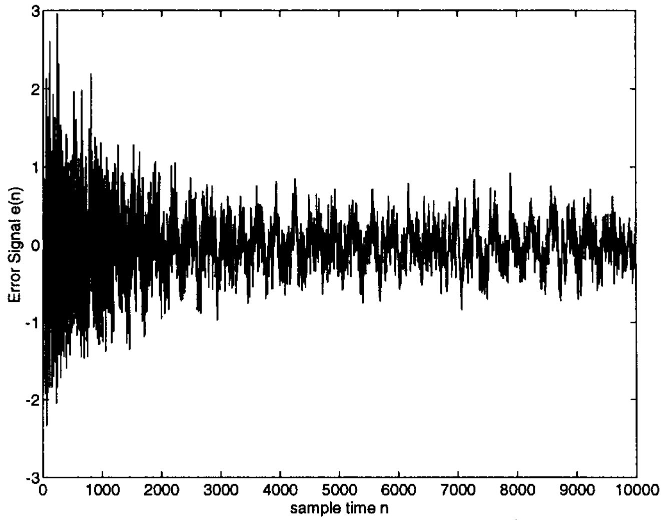

FIGURE 18.8: Convergence of the error signal in the loudspeaker identification experiment.

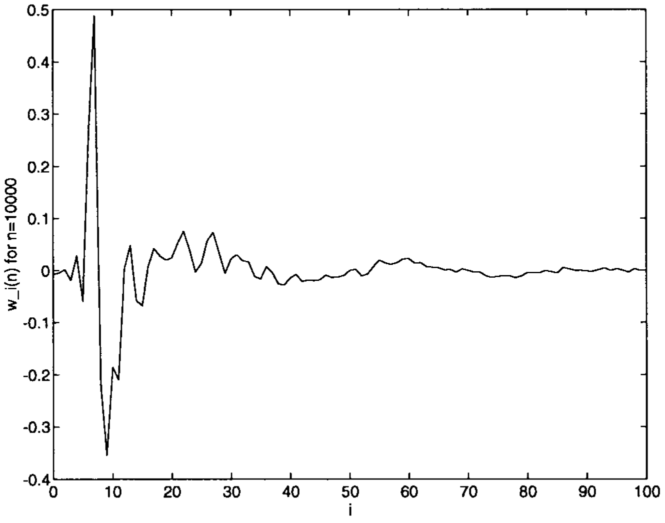

FIGURE 18.9: The adaptive filter coefficients obtained in the loudspeaker identification experiment.

microphone signal, indicating that the filter has converged. Figure 18.9 shows the coefficients of the adaptive filter at iteration $n = 10000$. The impulse response of the loudspeaker/room/microphone path consists of a large pulse corresponding to the direct sound propagation path as well as numerous smaller pulses caused by reflections of sounds off walls and other surfaces in the room.

18.7 Conclusions

In this section, we have presented an overview of adaptive filters, emphasizing the applications and basic algorithms that have already proven themselves to be useful in practice. Despite the many contributions in the field, research efforts in adaptive filters continue at a strong pace, and it is likely that new applications for adaptive filters will be developed in the future. To keep abreast of these advances, the reader is urged to consult journals such as the *IEEE Transactions on Signal Processing* as well as the proceedings of yearly conferences and workshops in the signal processing and related fields.

References

[1] Kuo, S. and Chen, C., Implementation of adaptive filters with the TMS320C25 or the TMS320C30, in *Digital Signal Processing Applications with the TMS320 Family*, Papamichalis, P., Ed., Prentice-Hall, Englewood Cliffs, NJ, 1991, 191–271.

[2] Analog Devices, Adaptive Filters, in *ADSP-21000 Family Application Handbook*, vol. 1, Analog Devices, 1994, 157–203.

[3] El-Sharkawy, M., Designing adaptive FIR filters and implementing them on the DSP56002 processor, in *Digital Signal Processing Applications with Motorola's DSP56002 Processor*, Prentice-Hall, Upper Saddle River, NJ, 1996, 319–342.

[4] Borth, D.E., Gerson, I.A., Haug, J.R., and Thompson, C.D., A flexible adaptive FIR filter VLSI IC, *IEEE J. Sel. Areas Commun.*, 6(3), 494–503, April 1988.

[5] Oppenheim, A.V. and Schafer, A.W., *Discrete-Time Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1989.

[6] Friedlander, B., Lattice filters for adaptive processing, *Proc. IEEE*, 70(8), 829–867, Aug. 1982.

[7] Mathews, V.J., Adaptive polynomial filters, *IEEE Signal Processing Mag.*, 8(3), 10–26, July 1991.

[8] Haykin, S., *Neural Networks: A Comprehensive Foundation*, Macmillan, New York, 1994.

[9] Proakis, J.G. and Salehi, M., *Communication Systems Engineering*, Prentice-Hall, Englewood Cliffs, NJ, 1994.

[10] Åström, K.G. and Wittenmark, B., *Adaptive Control*, Addison-Wesley, Reading, MA, 1989.

[11] Widrow, B. and Walach, E., *Adaptive Inverse Control*, Prentice-Hall, Upper Saddle River, NJ, 1996.

[12] Sondhi, M.M., An adaptive echo canceller, *Bell Sys. Tech. J.*, 46, 497–511, March 1967.

[13] Messerschmitt, D.G., Echo cancellation in speech and data transmission, *IEEE J. Sel. Areas Commun.*, SAC-2(2), 283–297, March 1984.

[14] Murano, K., Unagami, S., and Amano, F., Echo cancellation and applications, *IEEE Commun. Mag.*, 28(1), 49–55, Jan. 1990.

[15] Widrow, B., Glover, J.R., Jr., McCool, J.M., Kaunitz, J., Williams, C.S., Hearn, R.H., Zeidler, J.R., Dong, E., Jr., and Goodlin, R.C., Adaptive noise cancelling: principles and applications, *Proc. IEEE*, 63(12), 1692–1716, Dec. 1975.

[16] Lucky, R.W., Techniques for adaptive equalization of digital communication systems, *Bell Sys. Tech. J.*, 45, 255–286, Feb. 1966.

[17] Qureshi, S.U.H., Adaptive equalization, *Proc. IEEE*, 73(9), 1349–1387, Sept. 1985.

[18] Robinson, E.A. and Durrani, T., *Geophysical Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1986.

[19] Makhoul, J., Linear prediction: A tutorial review, *Proc. IEEE*, 63(4), 561–580, April 1975.

[20] Zeidler, J.R., Performance analysis of LMS adaptive prediction filters, *Proc. IEEE*, 78(12), 1781–1806, Dec. 1990.

[21] Kuo, S.M. and Morgan, D.R., *Active Noise Control Systems: Algorithms and DSP Implementations*, John Wiley & Sons, New York, 1996.

[22] Fuller, C.R., Elliott, S.J., and Nelson, P.A., *Active Control of Vibration*, Academic Press, London, 1996.

[23] Wiener, N., *Extrapolation, Interpolation, and Smoothing of Stationary Time Series, with Engineering Applications*, MIT Press, Cambridge, MA, 1949.

[24] Levinson, N., The Wiener RMS (root-mean-square) error criterion in filter design and prediction, *J. Math Phys.*, 25, 261–278, 1947.

[25] Douglas, S.C. and Meng, T.H.-Y., Stochastic gradient adaptation under general error criteria, *IEEE Trans. Signal Processing*, 42(6), 1335–1351, June 1994.

[26] Caraiscos, C. and Liu, B., A roundoff error analysis of the LMS adaptive algorithm, *IEEE Trans. Acoust., Speech, Signal Processing*, ASSP-32(1), 34–41, Feb. 1984.

[27] Duttweiler, D.L., Adaptive filter performance with nonlinearities in the correlation multiplier, *IEEE Trans. Acoust., Speech, Signal Processing*, ASSP-30(4), 578–586, Aug. 1982.