CHAPTER

Adaptive Filters and Applications 10

CHAPTER OUTLINE

OBJECTIVES

This chapter introduces principles of adaptive filters and the adaptive least mean square algorithm and illustrates how to apply the adaptive filters to solve the real-world application problems such as adaptive noise cancellation, system modeling, adaptive line enhancement, and telephone echo cancellation.

10.1 INTRODUCTION TO LEAST MEAN SQUARE ADAPTIVE FINITE IMPULSE RESPONSE FILTERS

An *adaptive filter* is a digital filter that has self-adjusting characteristics. It is capable of adjusting its filter coefficients automatically to adapt the input signal via an adaptive algorithm. Adaptive filters play an important role in modern digital signal processing (DSP) products in areas such as telephone echo cancellation, noise cancellation, equalization of communications channels, biomedical signal enhancement, active noise control, and adaptive control systems. Adaptive filters work generally for adaptation of signal-changing environments, spectral overlap between noise and signal, and unknown or time-varying noise. For example, when the interference noise is strong and its spectrum overlaps that of the desired signal, removing the interference using a traditional filter such as a notch filter with fixed filter coefficients will fail to preserve the desired signal spectrum, as shown in Figure 10.1.

However, an adaptive filter will do the job. Note that adaptive filtering, with its applications, has existed for more than two decades in the research community and is still active there. This chapter can

Digital Signal Processing. http://dx.doi.org/10.1016/B978-0-12-415893-1.00010-X

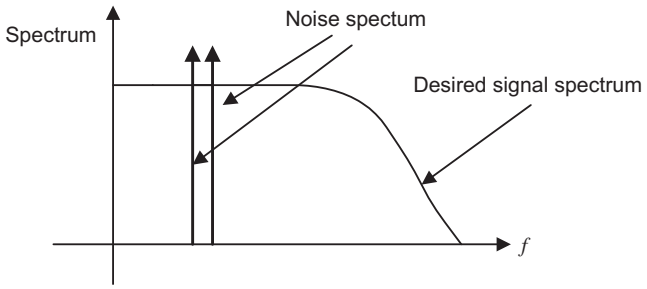

FIGURE 10.1

Spectrum illustration for using adaptive filters.

only introduce some fundaments of the subject, that is, adaptive finite impulse response (FIR) filters with a simple and popular least mean square (LMS) algorithm. Further exploration into adaptive infinite impulse response (IIR) filters, adaptive lattice filters, their associated algorithms and applications, and so on, can be found in comprehensive texts by Haykin (1991), Stearns (2003), and Widrow and Stearns (1985).

To understand the concept of adaptive filtering, we will first look at an illustrative example of the simplest noise canceller to see how it works before diving into detail. The block diagram for such a noise canceller is shown in Figure 10.2.

As shown in Figure 10.2, first, the DSP system consists of two analog-to-digital conversion (ADC) channels. The first microphone with ADC is used to capture the desired speech $s(n)$. However, due to a noisy environment, the signal is contaminated and the ADC channel produces a signal with the noise; that is, $d(n) = s(n) + n(n)$. The second microphone is placed where only noise is picked up and the second ADC channel captured noise $x(n)$, which is fed to the adaptive filter.

Note that the corrupting noise $n(n)$ in the first channel is uncorrelated to the desired signal $s(n)$, so that separation between them is possible. The noise signal $x(n)$ from the second channel is correlated

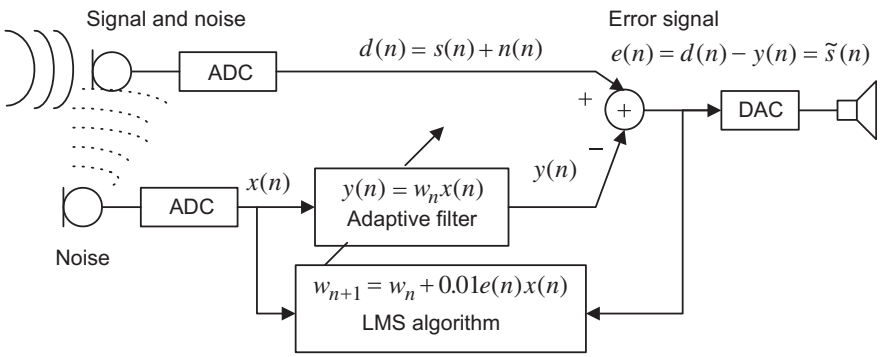

FIGURE 10.2

Simplest noise canceller using a one-tap adaptive filter.

to the corrupting noise $n(n)$ in the first channel, since both come from the same noise source. Similarly, the noise signal $x(n)$ is not correlated to the desired speech signal $s(n)$.

We assume that the corrupting noise in the first channel is a linear filtered version of the second-channel noise, since it has a different physical path from the second-channel noise, and the noise source is time varying, so that we can estimate the corrupting noise $n(n)$ using an adaptive filter. The adaptive filter contains a digital filter with adjustable coefficient(s) and the LMS algorithm to modify the value(s) of coefficient(s) for filtering each sample. The adaptive filter then produces an estimate of noise $y(n)$, which will be subtracted from the corrupted signal $d(n) = s(n) + n(n)$. When the noise estimate $y(n)$ equals or approximates the noise $n(n)$ in the corrupted signal, that is, $y(n) \approx n(n)$, the error signal $e(n) = s(n) + n(n) - y(n) \approx \tilde{s}(n)$ will approximate the clean speech signal $s(n)$. Hence, the noise is cancelled.

In our illustrative numerical example, the adaptive filter is set to be one-tap FIR filter to simplify numerical algebra. The filter adjustable coefficient w_n is adjusted based on the LMS algorithm (discussed later in detail) in the following:

$$w_{n+1} = w_n + 0.01 \cdot e(n) \cdot x(n)$$

where w_n is the coefficient used currently, while w_{n+1} is the coefficient obtained from the LMS algorithm and will be used for the next coming input sample. The value of 0.01 controls the speed of the coefficient change. To illustrate the concept of the adaptive filter in Figure 10.2, the LMS algorithm has the initial coefficient set to $w_0 = 0.3$ and leads to

$$\begin{aligned} y(n) &= w_n x(n) \\ e(n) &= d(n) - y(n) \\ w_{n+1} &= w_n + 0.01 e(n) x(n) \end{aligned}$$

The corrupted signal is generated by adding noise to a sine wave. The corrupted signal and noise reference are shown in Figure 10.3, and their first 16 values are listed in Table 10.1.

Let us perform adaptive filtering for several samples using the values for the corrupted signal and reference noise in Table 10.1. We see that

$$\begin{aligned} n = 0, \quad y(0) &= w_0 x(0) = 0.3 \times (-0.5893) = -0.1768 \\ e(0) &= d(0) - y(0) = -0.2947 - (-0.1768) = -0.1179 = \tilde{s}(0) \\ w_1 &= w_0 + 0.01 e(0) x(0) = 0.3 + 0.01 \times (-0.1179) \times (-0.5893) = 0.3007 \end{aligned}$$

$$\begin{aligned} n = 1, \quad y(1) &= w_1 x(1) = 0.3007 \times 0.5893 = 0.1772 \\ e(1) &= d(1) - y(1) = 1.0017 - 0.1772 = 0.8245 = \tilde{s}(1) \\ w_2 &= w_1 + 0.01 e(1) x(1) = 0.3007 + 0.01 \times 0.8245 \times 0.5893 = 0.3056 \end{aligned}$$

$$\begin{aligned} n = 2, \quad y(2) &= w_2 x(2) = 0.3056 \times 3.1654 = 0.9673 \\ e(2) &= d(2) - y(2) = 2.5827 - 0.9673 = 1.6155 = \tilde{s}(2) \\ w_3 &= w_2 + 0.01 e(2) x(2) = 0.3056 + 0.01 \times 1.6155 \times 3.1654 = 0.3567 \end{aligned}$$

$$n = 3, \ \cdots$$

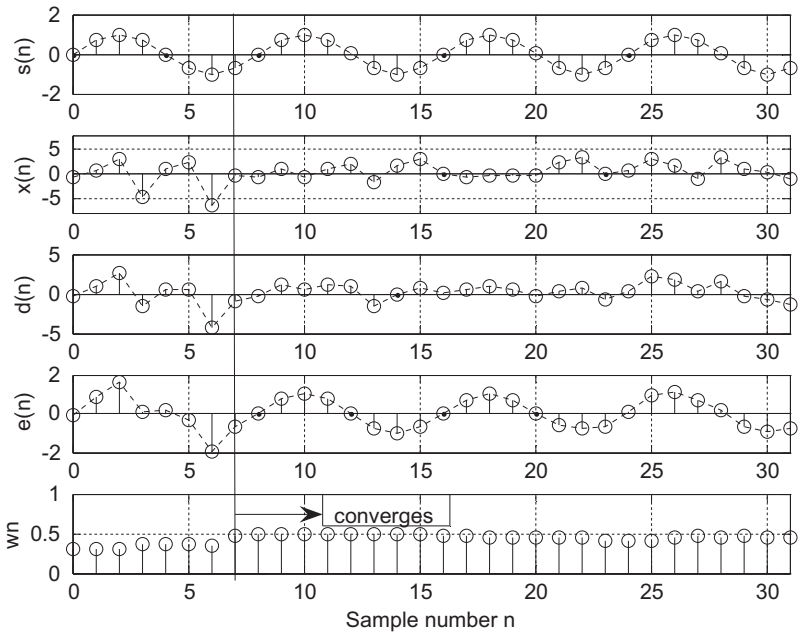

FIGURE 10.3

Original signal, reference noise, corrupted signal, enhanced signal, and adaptive coefficient in the noise cancellation.

TABLE 10.1 Adaptive Filtering Results for the Simplest Noise Canceller Example

n	$d(n)$	$x(n)$	$\tilde{s}(n) = e(n)$	Original $s(n)$	w_{n+1}
0	−0.2947	−0.5893	−0.1179	0	0.3000
1	1.0017	0.5893	0.8245	0.7071	0.3007
2	2.5827	3.1654	1.6155	1.0000	0.3056
3	−1.6019	−4.6179	0.0453	0.7071	0.3567
4	0.5622	1.1244	0.1635	0.0000	0.3546
5	0.4456	2.3054	−0.3761	−0.7071	0.3564
6	−4.2674	−6.5348	−1.9948	−1.0000	0.3478
7	−0.8418	−0.2694	−0.7130	−0.7071	0.4781
8	−0.3862	−0.7724	−0.0154	−0.0000	0.4800
9	1.2274	1.0406	0.7278	0.7071	0.4802
10	0.6021	−0.7958	0.9902	1.0000	0.4877
11	1.1647	0.9152	0.7255	0.7071	0.4799
12	0.9630	1.9260	0.0260	0.0000	0.4865
13	−1.5065	−1.5988	−0.7279	−0.7071	0.4870
14	−0.1329	1.7342	−0.9976	−1.0000	0.4986
15	0.8146	3.0434	−0.6503	−0.7071	0.4813

For comparison, results of the first 16 processed output samples, original samples, and filter coefficient values are also included in Table 10.1. Figure 10.3 also shows the original signal samples, reference noise samples, corrupted signal samples, enhanced signal samples, and filter coefficient values for each incoming sample, respectively.

As shown in Figure 10.3, after seven adaptations, the adaptive filter learns noise characteristics and cancels the noise in the corrupted signal. The adaptive coefficient is close to the optimal value of 0.5. The processed output is close to the original signal. The first 16 processed values for corrupted signal, reference noise, clean signal, original signal, and adaptive filter coefficient used at each step are listed in Table 10.1.

Clearly, the enhanced signal samples look much like the sinusoid input samples. Now our simplest one-tap adaptive filter works for this particular case. In general, an FIR filter with multiple taps is used and has the following format:

$$y(n) = \sum_{i=0}^{N-1} w_n(i)x(n-i) = w_n(0)x(n) + w_n(1)x(n-1) + \cdots + w_n(N-1)x(n-N+1) \quad (10.1)$$

The LMS algorithm for the adaptive FIR filter will be developed next.

10.2 BASIC WIENER FILTER THEORY AND LEAST MEAN SQUARE ALGORITHM

Many adaptive algorithms can be viewed as approximations of the discrete Wiener filter shown in Figure 10.4, where the Wiener filter output y(n) is a sum of its N weighted inputs, that is,

$$y(n) = w(0)x(n) + w(1)x(n-1) + \ldots + w(N-1)x(n-N+1).$$

The Wiener filter adjusts its weight(s) to produce a desired filter output $y(n)$ which is close to the noise $n(n)$ contained in the corrupted signal $d(n)$. At the subtracted output, the noise $n(n)$ is cancelled or attenuated. Hence, the output $e(n)$ contains a clean signal.

Consider a single-weight case of $y(n) = wx(n)$, and note that the error signal $e(n)$ is given by

$$e(n) = d(n) - wx(n) \quad (10.2)$$

Now let us determine the best weight w^*. Taking the square or enhanced the output error leads to

$$e^2(n) = (d(n) - wx(n))^2 = d^2(n) - 2d(n)wx(n) + w^2x^2(n) \quad (10.3)$$

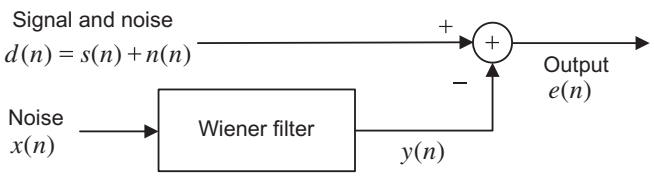

FIGURE 10.4

Wiener filter for noise cancellation.

Taking the statistical expectation of Equation (10.3), we have

$$E\left(e^2(n)\right) = E\left(d^2(n)\right) - 2wE(d(n)x(n)) + w^2E\left(x^2(n)\right) \tag{10.4}$$

Using the notations in statistics, we define

$$\begin{aligned} J &= E\left(e^2(n)\right) = MSE = \textit{mean squared error} \\ \sigma^2 &= E\left(d^2(n)\right) = \textit{power of corrupted signal} \\ P &= E(d(n)x(n)) = \textit{cross-correlation between } d(n) \textit{ and } x(n) \\ R &= E\left(x^2(n)\right) = \textit{autocorrelation} \end{aligned}$$

We can view the statistical expectation as an average of the N signal terms, each being a product of two individual samples

$$E\left(e^2(n)\right) = \frac{e^2(0) + e^2(1) + \cdots + e^2(N-1)}{N}$$

or

$$E(d(n)x(n)) = \frac{d(0)x(0) + d(1)x(1) + \cdots + d(N-1)x(N-1)}{N}$$

for a sufficiently large sample number of N. We can write Equation (10.4) as

$$J = \sigma^2 - 2wP + w^2R \tag{10.5}$$

Since σ^2, P, and R are constants, J is a quadratic function of w that may be plotted as shown in Figure 10.5.

The best weight (optimal) w^* is at the location where the minimum MSE $J_{\min}$ is achieved. To obtain w^*, taking a derivative of J and setting it to zero leads to

$$\frac{dJ}{dw} = -2P + 2wR = 0 \tag{10.6}$$

Solving Equation (10.6), we get the best weight solution as

$$w^* = R^{-1}P \tag{10.7}$$

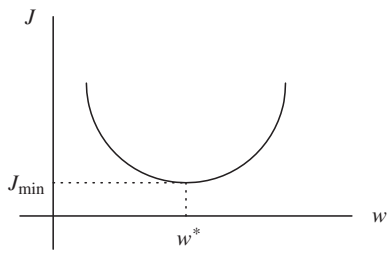

FIGURE 10.5

Mean square error quadratic function.

EXAMPLE 10.1

Consider the following quadratic MSE function for the Wiener filter:

$$J = 40 - 20w + 10w^2$$

Find the optimal solution for w^* to achieve the minimum MSE $J_{\min}$ and determine $J_{\min}$.

Solution:
Taking a derivative of the MSE function and setting it to zero, we have

$$\frac{dJ}{dw} = -20 + 10 \times 2w = 0$$

Solving the equation leads to

$$w^* = 1$$

Finally, substituting $w^* = 1$ into the MSE function, we get the minimum $J_{\min}$ as

$$J_{\min} = J|_{w=w^*} = 40 - 20w + 10w^2|_{w=1} = 40 - 20 \times 1 + 10 \times 1^2 = 30$$

Notice that a few points need to be clarified for Equation (10.7):

1. Optimal coefficient (s) can be different for every block of data, since the corrupted signal and reference signal are unknown. The autocorrelation and cross-correlation may vary.
2. If a larger number of coefficients (weights) are used, the inverse matrix of R^{-1} may require a larger number of computations and may become ill-conditioned. This will make real-time implementation impossible.
3. The optimal solution is based on the statistics, assuming that the size of the data block, N, is sufficient long. This will cause a long processing delay that will make real-time implementation impossible.

As we pointed out, solving the Wiener solution, Equation (10.7), requires a lot of computations, including matrix inversion for a general multiple-tap FIR filter. The well-known textbook authored by Widrow and Stearns (1985) described a powerful LMS algorithm by using the steepest descent algorithm to minimize the MSE sample by sample to locate the filter coefficient(s). We first study the steepest descent algorithm illustrated in the following:

$$w_{n+1} = w_n - \mu \frac{dJ}{dw} \tag{10.8}$$

where μ = constant controlling speed of convergence.

The illustration of the steepest decent algorithm for solving the optimal coefficient(s) is described in Figure 10.6.

As shown in the first plot in Figure 10.6, if $\frac{dJ}{dw} < 0$, notice that $-\mu \frac{dJ}{dw} > 0$. The new coefficient w_{n+1} will be increased to approach the optimal value w^* by Equation (10.8). On the other hand, if $\frac{dJ}{dw} > 0$, as shown in the second plot in Figure 10.6, we see that $-\mu \frac{dJ}{dw} < 0$. The new

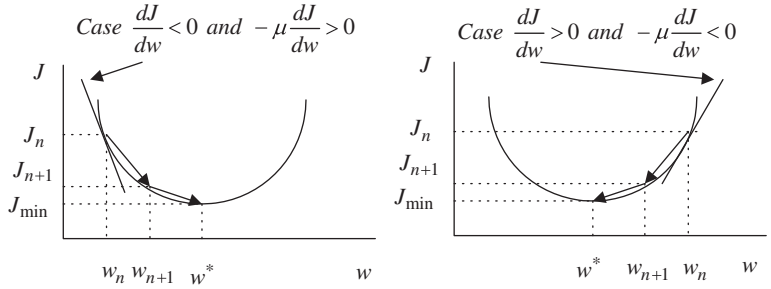

FIGURE 10.6

Illustration of the steepest descent algorithm.

coefficient w_{n+1} will be decreased to approach the optimal value w^*. When $\frac{dJ}{dw} = 0$, the best coefficient w_{n+1} is reached.

EXAMPLE 10.2

Consider the following quadratic MSE function for the Wiener filter:

$$J = 40 - 20w + 10w^2$$

Use the steepest decent method with an initial guess of $w_0 = 0$ and $\mu = 0.04$ to find the optimal solution for w^* and determine $J_{\min}$ by iterating three times.

Solution:

Taking a derivative of the MSE function, we have

$$\frac{dJ}{dw} = -20 + 10 \times 2w_n$$

When $n = 0$, we calculate

$$\mu\frac{dJ}{dw} = 0.04 \times (-20 + 10 \times 2w_0)\big|_{w_0=0} = -0.8$$

Applying the steepest decent algorithm, it follows that

$$w_1 = w_0 - \mu\frac{dJ}{dw} = 0 - (-0.8) = 0.8$$

Similarly for $n = 1$, we get

$$\mu\frac{dJ}{dw} = 0.04 \times (-20 + 10 \times 2w_1)\big|_{w_1=0.8} = -0.16$$

$$w_2 = w_1 - \mu\frac{dJ}{dw} = 0.8 - (-0.16) = 0.96$$

and for $n = 2$, it follows that

$$\mu\frac{dJ}{dw} = 0.04 \times (-20 + 10 \times 2w_2)\big|_{w_2=0.96} = -0.032$$

$$w_3 = w_2 - \mu \frac{dJ}{dw} = 0.96 - (-0.032) = 0.992$$

Finally, substituting $w^* \approx w_3 = 0.992$ into the MSE function, we get the minimum $J_{\min}$ as

$$J_{\min} \approx 40 - 20w + 10w^2\big|_{w=0.992} = 40 - 20 \times 0.992 + 10 \times 0.992^2 = 30.0006$$

As we can see, after three iterations, the filter coefficient and minimum MSE values are very close to the theoretical values obtained in Example 10.1.

Application of the steepest descent algorithm still needs an estimation of the derivative of the MSE function that could include statistical calculation of a block of data. To change the algorithm to do sample-based processing, an LMS algorithm must be used. To develop the LMS algorithm in terms of sample-based processing, we take the statistical expectation out of J and then take the derivative to obtain an approximation of $\frac{dJ}{dw}$, that is,

$$J = e^2(n) = (d(n) - wx(n))^2 \tag{10.9}$$

$$\frac{dJ}{dw} = 2(d(n) - wx(n))\frac{d(d(n) - wx(n))}{dw} = -2e(n)x(n) \tag{10.10}$$

Substituting $\frac{dJ}{dw}$ into the steepest descent algorithm in Equation (10.8), we achieve the LMS algorithm for updating a single-weight case as

$$w_{n+1} = w_n + 2\mu e(n)x(n) \tag{10.11}$$

where μ is the convergence parameter controlling speed of convergence. For example, let us choose $2\mu = 0.01$. In general, with an adaptive FIR filter of length N, we extend the single-tap LMS algorithm without going through derivation, as shown in the following equations:

$$y(n) = w_n(0)x(n) + w_n(1)x(n-1) + \cdots + w_n(N-1)x(n-N+1) \tag{10.12}$$

$$\begin{aligned} &\text{for } i = 0, \cdots, N-1 \\ &w_{n+1}(i) = w_n(i) + 2\mu e(n)x(n-i) \end{aligned} \tag{10.13}$$

The convergence factor is chosen to be

$$0 < \mu < \frac{1}{NP_x} \tag{10.14}$$

where P_x is the input signal power. In practice, if the ADC has 16-bit data, the maximum signal amplitude should be $A = 2^{15}$. Then the maximum input power must be less than

$$P_x < \left(2^{15}\right)^2 = 2^{30}$$

Hence, we may make a selection of the convergence parameter as

$$\mu = \frac{1}{N \times 2^{30}} \approx \frac{9.3 \times 10^{-10}}{N} \tag{10.15}$$

We further neglect time index for $w_n(i)$ and use the notation $w(i) = w_n(i)$, since only the current updated coefficients are needed for next sample adaptation. We conclude the implementation of the LMS algorithm with the following steps:

1. Initialize $w(0), w(1), \ldots w(N-1)$ to arbitrary values.
2. Read $d(n)$, $x(n)$, and perform digital filtering:

$$y(n) = w(0)x(n) + w(1)x(n-1) + \cdots + w(N-1)x(n-N+1)$$

3. Compute the output error:

$$e(n) = d(n) - y(n)$$

4. Update each filter coefficient using the LMS algorithm:

$$\begin{aligned} &\text{for } i = 0, \cdots, N-1 \\ &w(i) = w(i) + 2\mu e(n)x(n-i) \end{aligned}$$

We will apply the adaptive filter to solve real-world problems in the next section.

10.3 APPLICATIONS: NOISE CANCELLATION, SYSTEM MODELING, AND LINE ENHANCEMENT

We now examine several applications of the LMS algorithm, such as noise cancellation, system modeling, and line enhancement via application examples. First, we begin with the noise cancellation problem to illustrate operations of the LMS adaptive FIR filter.

10.3.1 Noise Cancellation

The concept of noise cancellation was introduced in the previous section. Figure 10.7 shows the main concept.

The DSP system consists of two ADC channels. The first microphone with ADC captures the noisy speech, $d(n) = s(n) + n(n)$, which contains the clean speech $s(n)$ and noise $n(n)$ due to a noisy environment, while the second microphone with ADC resides where it picks up only the correlated noise and feeds the noise reference $x(n)$ to the adaptive filter. The adaptive filter uses the LMS algorithm to adjust its coefficients to produce the best estimate of noise $y(n) \approx n(n)$, which will be subtracted from the corrupted signal $d(n) = s(n) + n(n)$. The output of the error signal $e(n) = s(n) + n(n) - y(n) \approx \tilde{s}(n)$ is expected to be the best estimate of the clean speech signal. Through digital-to-analog conversion (DAC), the cleaned digital speech becomes analog voltage, which drives the speaker.

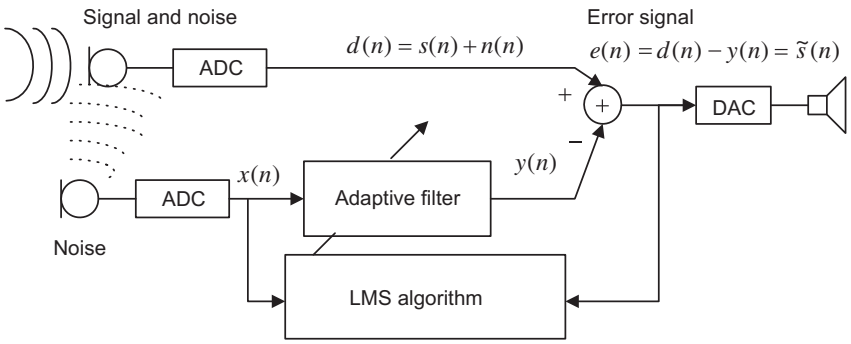

FIGURE 10.7

Simplest noise canceller using a one-tap adaptive filter.

We first study the noise cancellation problem using a simple two-tap adaptive filter via Example 10.3 and assumed data. The purpose of doing so is to become familiar with the setup and operations of the adaptive filter and LMS algorithm. The simulation for real adaptive noise cancellation follows.

EXAMPLE 10.3

Consider the DSP system for the noise cancellation application using an adaptive filter with two coefficients shown in Figure 10.8.

a. Set up the LMS algorithm for the adaptive filter.
b. Perform adaptive filtering to obtain outputs $e(n)$ for $n = 0, 1, 2$ given the following inputs and outputs:

$$x(0) = 1,\ x(1) = 1,\ x(2) = -1,\ d(0) = 2,\ d(1) = 1,\ d(2) = -2$$

The initial weights are $w(0) = w(1) = 0$, and the convergence factor is set to be $\mu = 0.1$.

Solution:

a. The adaptive LMS algorithm is set up as:
Initialization: $w(0) = 0,\ w(1) = 0$
Digital filtering: $y(n) = w(0)x(n) + w(1)x(n-1)$

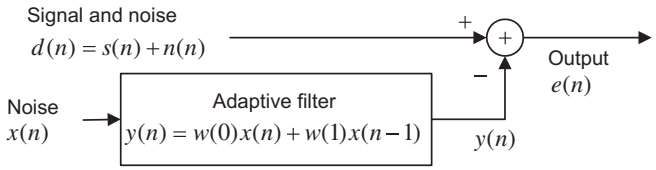

FIGURE 10.8

Noise cancellation in Example 10.3.

Computing the output: $e(n) = d(n) - y(n)$

Updating each weight to be used for the next coming sample:

$$w(i) = w(i) + 2\mu e(n)x(n-i), \quad \text{for} \quad i = 0, 1$$

or

$$w(0) = w(0) + 2\mu e(n)x(n)$$

$$w(1) = w(1) + 2\mu e(n)x(n-1)$$

b. We can see the adaptive filtering operations as follows:

For $n = 0$

Digital filtering:

$$y(0) = w(0)x(0) + w(1)x(-1) = 0 \times 1 + 0 \times 0 = 0$$

Computing the output:

$$e(0) = d(0) - y(0) = 2 - 0 = 2$$

Updating coefficients:

$$w(0) = w(0) + 2 \times 0.1 \times e(0)x(0) = 0 + 2 \times 0.1 \times 2 \times 1 = 0.4$$

$$w(1) = w(1) + 2 \times 0.1 \times e(0)x(-1) = 0 + 2 \times 0.1 \times 2 \times 0 = 0.0$$

For $n = 1$

Digital filtering:

$$y(1) = w(0)x(1) + w(1)x(0) = 0.4 \times 1 + 0 \times 1 = 0.4$$

Computing the output:

$$e(1) = d(1) - y(1) = 1 - 0.4 = 0.6$$

Updating coefficients:

$$w(0) = w(0) + 2 \times 0.1 \times e(1)x(1) = 0.4 + 2 \times 0.1 \times 0.6 \times 1 = 0.52$$

$$w(1) = w(1) + 2 \times 0.1 \times e(1)x(0) = 0 + 2 \times 0.1 \times 0.6 \times 1 = 0.12$$

For $n = 2$

Digital filtering:

$$y(2) = w(0)x(2) + w(1)x(1) = 0.52 \times (-1) + 0.12 \times 1 = -0.4$$

Computing the output:

$$e(2) = d(2) - y(2) = -2 - (-0.4) = -1.6$$

Updating coefficients:

$$w(0) = w(0) + 2 \times 0.1 \times e(2)x(2) = 0.52 + 2 \times 0.1 \times (-1.6) \times (-1) = 0.84$$

$$w(1) = w(1) + 2 \times 0.1 \times e(2)x(1) = 0.12 + 2 \times 0.1 \times (-1.6) \times 1 = -0.2$$

Hence, the adaptive filter outputs for the first three samples are listed as

$$e(0) = 2,\ e(1) = 0.6,\ e(2) = -1.6$$

Next we examine the MSE function assuming the following statistical data for the two-tap adaptive filter $y(n) = w(0)x(n) + w(1)x(n-1)$:

$$\sigma^2 = E\left[d^2(n)\right] = 4,\ E\left[x^2(n)\right] = E\left[x^2(n-1)\right] = 1,\ E[x(n)x(n-1)] = 0$$

$$E[d(n)x(n)] = 1, \text{ and } E[d(n)x(n-1)] = -1$$

We follow Equations (10.2) to (10.5) to achieve the minimum MSE function in two dimensions as

$$J = 4 + w^2(0) + w^2(1) - 2w(0) + 2w(1)$$

Figure 10.9 shows the MSE function versus the weights, where the optimal weights and the minimum MSE are $w^*(0) = 1$, $w^*(1) = -1$, and $J_{\min} = 2$. If the adaptive filter continues to process the data, it will converge to the optimal weights, which locate the minimum MSE. The plot also indicates that the function is quadratic and that there exists only one minimum of the MSE surface.

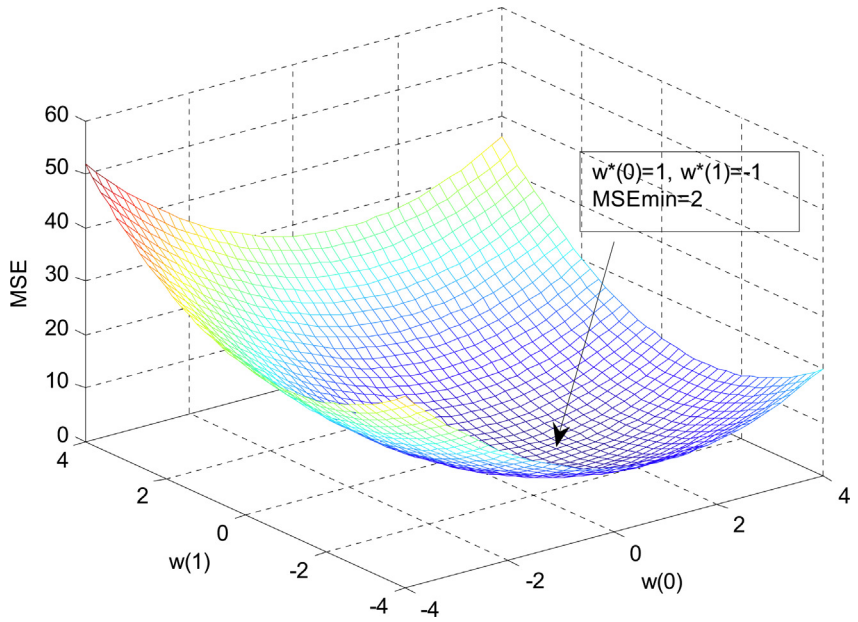

FIGURE 10.9

Plot of the MSE function versus two weights.

Next, a simulation example is given to illustrate this idea and its results. The noise cancellation system is assumed to have the following specifications:

- Sample rate = 8,000 Hz
- Original speech data: wen.dat
- Speech corrupted by Gaussian noise with a power of 1 delayed by 5 samples from the noise reference
- Noise reference containing Gaussian noise with a power of 1
- Adaptive FIR filter used to remove the noise
- Number of FIR filter taps = 21
- Convergence factor for the LMS algorithm is chosen to be 0.01 (<1/21).

The speech waveforms and spectral plots for the original, corrupted, and reference noise and for the cleaned speech are plotted in Figures 10.10A and Figure 10.10B. From the figures, it is observed that the enhanced speech waveform and spectrum are very close to the original ones. The LMS algorithm converges after approximately 400 iterations. The method is a very effective approach for noise canceling. The MATLAB implementation is detailed in Program 10.1.

Program 10.1. MATLAB program for adaptive noise cancellation.

```
close all; clear all
load wen.dat                          % Given by the instructor
fs=8000;                              % Sampling rate
t=0:1:length(wen)-1;                  % Create index array
```

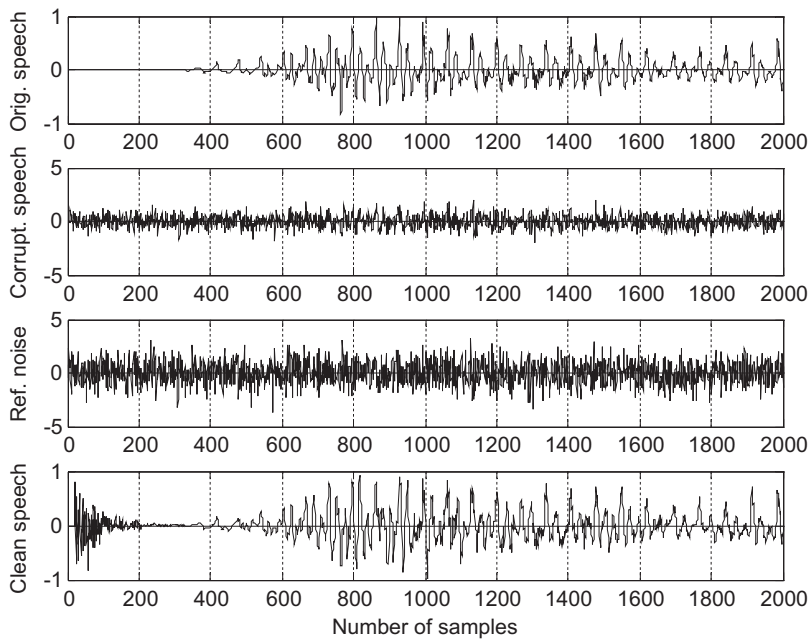

FIGURE 10.10A

Waveforms for original speech, corrupted speech, reference noise, and clean speech.

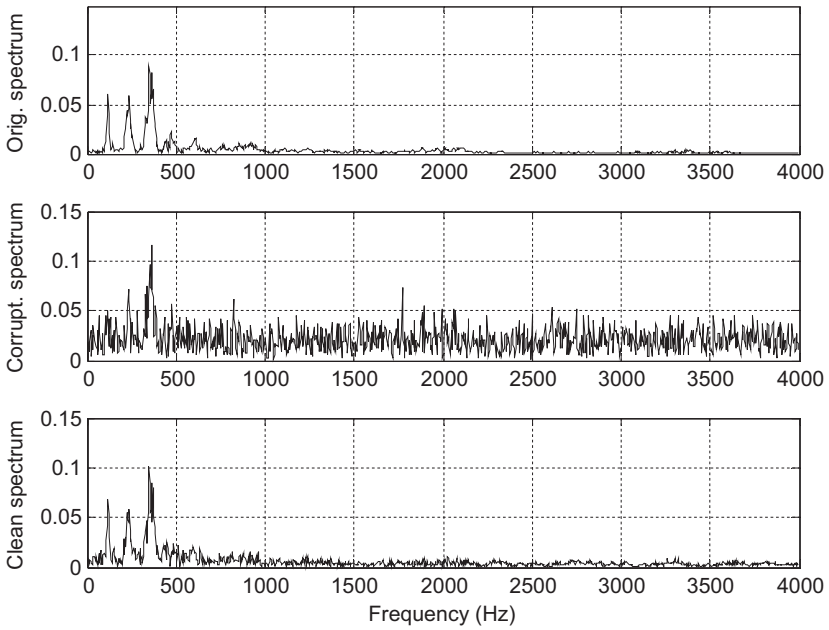

FIGURE 10.10B

Spectrum for original speech, corrupted speech, and clean speech.

```
t=t/fs;                                % Convert indices to time instant
x=randn(1,length(wen));                % Generate random noise
n=filter([ 0 0 0 0 0 0 0.5 ],1,x);     % Generate the corruption noise
d=wen+n;                  % Generate signal plus noise
mu=0.01;                  % Initialize step size
w=zeros(1,21);            % Initialize adaptive filter coefficients
y=zeros(1,length(t));     % Initialize the adaptive filter output array
e=y;                      % Initialize the output array
% Adaptive filtering using LMS algorithm
for m=22:1:length(t)-1
  sum=0;
  for i=1:1:21
  sum=sum+w(i)*x(m-i);
  end
  y(m)=sum;
  e(m)=d(m)-y(m);
  for i=1:1:21
  w(i)=w(i)+2*mu*e(m)*x(m-i);
  end
end
% Calculate the single-sided amplitude spectrum for the original signal
WEN=2*abs(fft(wen))/length(wen);WEN(1)=WEN(1)/2;
```

```
% Calculate the single-sided amplitude spectrum for the corrupted signal
D=2*abs(fft(d))/length(d);D(1)=D(1)/2;
f=[0:1:length(wen)/2]*8000/length(wen);
% Calculate the single-sided amplitude spectrum for the noise-cancelled signal
E=2*abs(fft(e))/length(e);E(1)=E(1)/2;
% Plot signals and spectrums
subplot(4,1,1), plot(wen);grid; ylabel('Orig. speech');
subplot(4,1,2),plot(d);grid; ylabel('Corrupt. speech')
subplot(4,1,3),plot(x);grid;ylabel('Ref. noise');
subplot(4,1,4),plot(e);grid; ylabel('Clean speech');
xlabel('Number of samples');
figure
subplot(3,1,1),plot(f,WEN(1:length(f)));grid
ylabel('Orig. spectrum')
subplot(3,1,2),plot(f,D(1:length(f)));grid; ylabel('Corrupt. spectrum')
subplot(3,1,3),plot(f,E(1:length(f)));grid
ylabel('Clean spectrum'); xlabel('Frequency (Hz)');
```

Other interference cancellations include that of 60-Hz interference cancellation in electrocardiography (ECG) (Chapter 8) and echo cancellation in long-distance telephone circuits, which will be described in Section 10.4.

10.3.2 System Modeling

Another application of the adaptive filter is system modeling. The adaptive filter can keep tracking the behavior of an unknown system by using the unknown system input and output, as depicted in Figure 10.11.

As shown in the figure, after the adaptive filter converges, the adaptive filter output $y(n)$ will approach to the unknown system's output. Since both the unknown system and the adaptive filter respond to the same input, the transfer function of the adaptive filter approximates the transfer function of the unknown system.

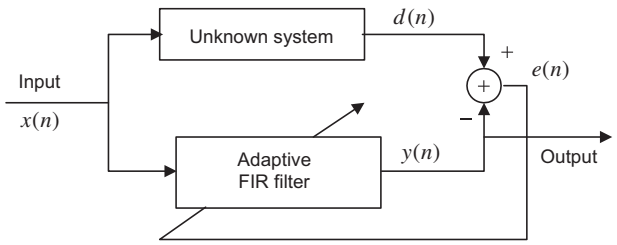

FIGURE 10.11

Adaptive filter for system modeling.

EXAMPLE 10.4

Given the system modeling described in this section and using a single-weight adaptive filter $y(n) = wx(n)$ to perform the system-modeling task,

a. set up the LMS algorithm to implement the adaptive filter assuming that initially $w = 0$ and $\mu = 0.5$;
b. perform adaptive filtering to obtain $y(0)$, $y(1)$, $y(2)$, and $y(3)$, given

$$d(0) = 1,\ d(1) = 2,\ d(2) = -2,\ d(3) = 2,$$

$$x(0) = 0.5,\ x(1) = 1,\ x(2) = -1,\ x(3) = 1$$

Solution:

a. Adaptive filtering equations are set up as

$$w = 0 \quad \text{and} \quad 2\mu = 2 \times 0.5 = 1$$

$$y(n) = wx(n)$$

$$e(n) = d(n) - y(n)$$

$$w = w + e(n)x(n)$$

b. Adaptive filtering:

$$n = 0, \qquad y(0) = wx(0) = 0 \times 0.5 = 0$$

$$e(0) = d(0) - y(0) = 1 - 0 = 1$$

$$w = w + e(0)x(0) = 0 + 1 \times 0.5 = 0.5$$

$$n = 1, \qquad y(1) = wx(1) = 0.5 \times 1 = 0.5$$

$$e(1) = d(1) - y(1) = 2 - 0.5 = 1.5$$

$$w = w + e(1)x(1) = 0.5 + 1.5 \times 1 = 2.0$$

$$n = 2, \qquad y(2) = wx(2) = 2 \times (-1) = -2$$

$$e(2) = d(2) - y(2) = -2 - (-2) = 0$$

$$w = w + e(2)x(2) = 2 + 0 \times (-1) = 2$$

$$n = 3, \qquad y(3) = wx(3) = 2 \times 1 = 2$$

$$e(3) = d(3) - y(3) = 2 - 2 = 0$$

$$w = w + e(3)x(3) = 2 + 0 \times 1 = 2$$

For this particular case, the system is actually a digital amplifier with a gain of 2.

Next, we assume the unknown system is a fourth-order bandpass IIR filter whose 3-dB lower and upper cutoff frequencies are 1,400 Hz and 1,600 Hz operating at 8,000 Hz. We use an input consisting

of tones of 500, 1,500, and 2,500 Hz. The unknown system's frequency responses are shown in Figure 10.12.

The input waveform $x(n)$ with three tones is shown as the first plot in Figure 10.13. We can predict that the output of the unknown system will contain a 1,500 Hz tone only, since the other two tones are rejected by the unknown system. Now, let us look at adaptive filter results. We use an FIR adaptive filter with the number of taps being 21, and a convergence factor set to 0.01. In the time domain, the output waveforms of the unknown system $d(n)$ and adaptive filter output $y(n)$ are almost identical after 70 samples when the LMS algorithm converges. The error signal $e(n)$ is also plotted to show the adaptive filter keeps tracking the unknown system's output with no difference after the first 50 samples.

Figure 10.14 depicts the frequency domain comparisons. The first plot displays the frequency components of the input signal, which clearly shows 500 Hz, 1,500 Hz, and 2,500 Hz. The second plot shows the unknown system's output spectrum, which contains only a 1,500 Hz tone, while the third plot displays the spectrum of the adaptive filter output. As we can see, in the frequency domain, the adaptive filter tracks the characteristics of the unknown system. The MATLAB implementation is given in Program 10.2.

Program 10.2. MATLAB program for adaptive system identification.

```
close all; clear all
%Design unknown system
fs=8000; T=1/fs;          % Sampling rate and sampling period
```

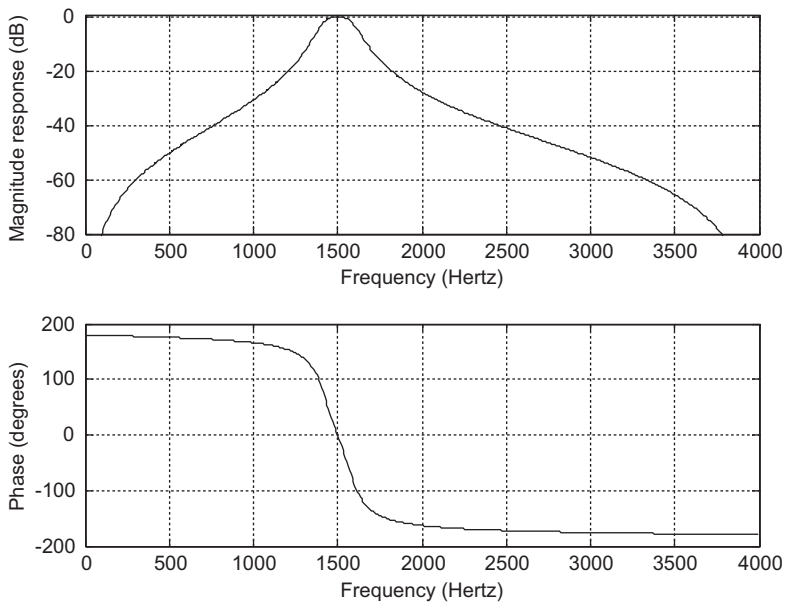

FIGURE 10.12

The unknown system's frequency responses.

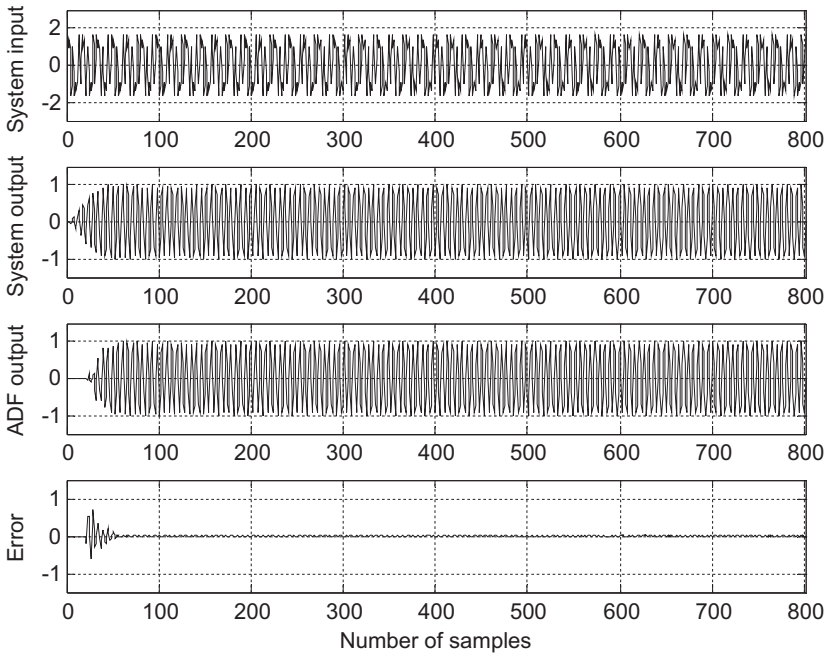

FIGURE 10.13

The waveforms for the unknown system's output, adaptive filter output, and error output.

```
% Bandpass filter design
% for the assumed unknown system using the bilinear transformation
%(BLT) method (see Chapter 8)
wd1=1400*2*pi; wd2=1600*2*pi;
wa1=(2/T)*tan(wd1*T/2); wa2=(2/T)*tan(wd2*T/2);
BW=wa2-wa1;
w0=sqrt(wa2*wa1);
[B,A]=lp2bp([1],[1 1.4141 1],w0,BW);
[b,a]=bilinear(B,A,fs);
freqz(b,a,512,fs); axis([0 fs/2 -80 1]); % Frequency response plots
figure
t=0:T:0.1;    % Generate the time vector
x=cos(2*pi*500*t)+sin(2*pi*1500*t)+cos(2*pi*2500*t+pi/4);
d=filter(b,a,x);  % Produce unknown system output
mu=0.01;    % Convergence factor
w=zeros(1,21); y=zeros(1,length(t)); % Initialize the coefficients and output
e=y;              % Initialize the error vector
% Perform adaptive filtering using LMS algorithm
for m=22:1:length(t)-1
  sum=0;
  for i=1:1:21
```

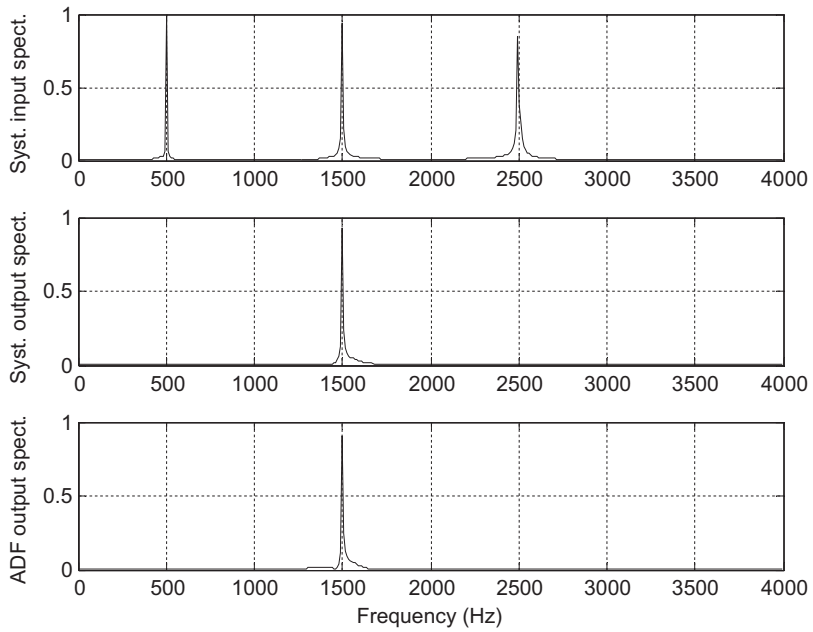

FIGURE 10.14

Spectrum for the input signal, unknown system output, and the adaptive filter output.

```
  sum=sum+w(i)*x(m-i);
  end
  y(m)=sum;
  e(m)=d(m)-y(m);
  for i=1:1:21
  w(i)=w(i)+2*mu*e(m)*x(m-i);
  end
end
% Calculate the single-sided amplitude spectrum for the input
X=2*abs(fft(x))/length(x);X(1)=X(1)/2;
% Calculate the single-sided amplitude spectrum for the unknown system output
D=2*abs(fft(d))/length(d);D(1)=D(1)/2;
% Calculate the single-sided amplitude spectrum for the adaptive filter output
Y=2*abs(fft(y))/length(y);Y(1)=Y(1)/2;
% Map the frequency index to its frequency in Hz
f=[0:1:length(x)/2]*fs/length(x);
% Plot signals and spectra
subplot(4,1,1), plot(x);grid; axis([0 length(x) -3 3]);
ylabel('System input');
subplot(4,1,2), plot(d);grid; axis([0 length(x) -1.5 1.5]);
ylabel('System output');
subplot(4,1,3),plot(y);grid; axis([0 length(y) -1.5 1.5]);
```

```
ylabel('ADF output')
subplot(4,1,4),plot(e);grid; axis([0 length(e) -1.5 1.5]);
ylabel('Error'); xlabel('Number of samples')
figure
subplot(3,1,1),plot(f,X(1:length(f)));grid; ylabel('Syst. input spect.')
subplot(3,1,2),plot(f,D(1:length(f)));grid; ylabel('Syst. output spect.')
subplot(3,1,3),plot(f,Y(1:length(f)));grid
ylabel('ADF output spect.'); xlabel('Frequency (Hz)');
```

10.3.3 Line Enhancement Using Linear Prediction

We study adaptive filtering via another application example: line enhancement. If the signal frequency content is very narrow compared with the bandwidth and changes with time, then the signal can efficiently be enhanced by the adaptive filter, which is line enhancement. Figure 10.15 shows line enhancement using the adaptive filter where the LMS algorithm is used. As illustrated in the figure, the signal $d(n)$ is the sine wave signal corrupted by the white Gaussian noise $n(n)$. The enhanced line consists of the delay element to delay the corrupted signal by Δ samples to produce an input to the adaptive filter. The adaptive filter is actually a linear predictor of the desired narrow band signal. A two-tap FIR adaptive filter can predict one sinusoid (proof is beyond the scope of this text). The value of Δ is usually determined by experiments or experience in practice to achieve the best enhanced signal.

Our simulation example has the following specifications:

- Sampling rate = 8,000 Hz
- Corrupted signal = 500 Hz tone with white Gaussian noise added to the unit amplitude
- Adaptive filter = FIR type, 21 taps
- Convergence factor = 0.001
- Delay value $\Delta = 7$
- LMS algorithm is applied

Figure 10.16 shows the time domain results. The first plot is the noisy signal, while the second plot clearly demonstrates the enhanced signal. Figure 10.17 describes the frequency domain point of view. The spectrum of the noisy signal is shown in the top plot, where we can see the white noise is populated

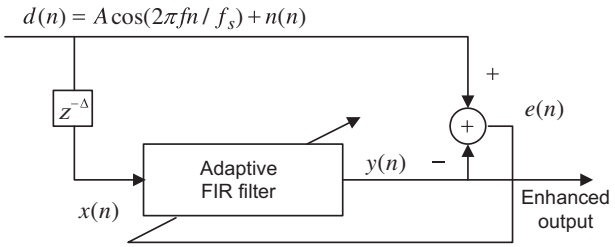

FIGURE 10.15

Line enhancement using an adaptive filter.

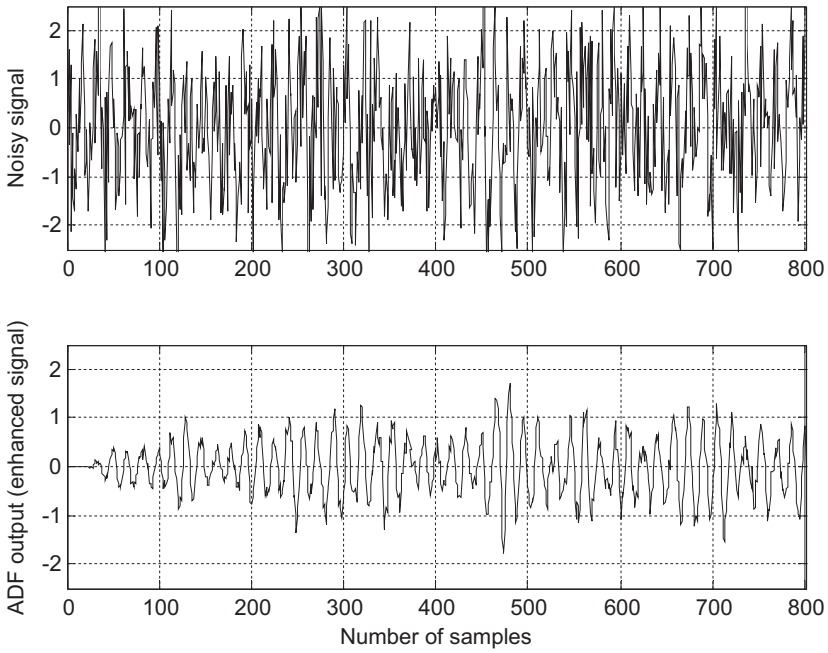

FIGURE 10.16

Noisy signal and enhanced signal.

over the entire bandwidth. The bottom plot is the enhanced signal spectrum. Since the method is adaptive, it is especially effective when the enhanced signal frequency is changing with time. Program 10.3 lists the MATLAB program for this simulation.

Program 10.3. MATLAB program for adaptive line enhancement.

```
close all; clear all
fs=8000; T=1/fs;                        % Sampling rate and sampling period
t=0:T:0.1;                              % 1 second time instant
n=randn(1,length(t));                   % Generate Gaussian random noise
d=cos(2*pi*500*t)+n;                    % Generate 500-Hz tone plus noise
x=filter([ 0 0 0 0 0 0 0 1 ],1,d);      % Delay filter
mu=0.001;                               % Initialize the step size for LMS algorithms
w=zeros(1,21);                          % Initialize the adaptive filter coefficients
y=zeros(1,length(t));                   % Initialize the adaptive filter output
e=y;                                    % Initialize the error vector
% Perform adaptive filtering using the LMS algorithm
for m=22:1:length(t)-1
  sum=0;
  for i=1:1:21
  sum=sum+w(i)*x(m-i);
```

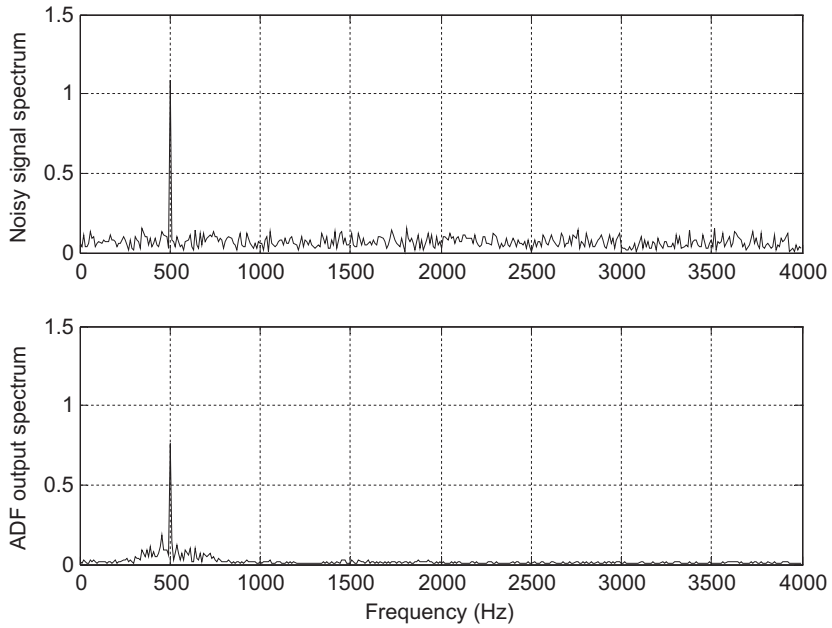

FIGURE 10.17

Spectrum plots for the noisy signal and enhanced signal.

```
  end
  y(m)=sum;
  e(m)=d(m)-y(m);
  for i=1:1:21
  w(i)=w(i)+2*mu*e(m)*x(m-i);
  end
end
% Calculate the single-sided amplitude spectrum for corrupted signal
D=2*abs(fft(d))/length(d);D(1)=D(1)/2;
% Calculate the single-sided amplitude spectrum for enhanced signal
Y=2*abs(fft(y))/length(y);Y(1)=Y(1)/2;
% Map the frequency index to its frequency in Hz
f=[0:1:length(x)/2]*8000/length(x);
% Plot the signals and spectra
subplot(2,1,1), plot(d);grid; axis([0 length(x) -2.5 2.5]); ylabel('Noisy signal');
subplot(2,1,2),plot(y);grid; axis([0 length(y) -2.5 2.5]);
ylabel('ADF output (enhanced signal)'); xlabel('Number of samples')
figure
subplot(2,1,1),plot(f,D(1:length(f)));grid; axis([0 fs/2 0 1.5]);
ylabel('Noisy signal spectrum')
subplot(2,1,2),plot(f,Y(1:length(f)));grid; axis([0 fs/2 0 1.5]);
ylabel('ADF output spectrum'); xlabel('Frequency (Hz)');
```

10.4 OTHER APPLICATION EXAMPLES

This section continues to explore other adaptive filter applications briefly, without showing computer simulations. The topics include periodic interference cancellation, ECG interference cancellation, and echo cancellation in long-distance telephone circuits. Detailed information can also be explored in Haykin (1991), Ifeachor and Jervis (2002), Stearns (2003), and Widrow and Stearns (1985).

10.4.1 Canceling Periodic Interferences Using Linear Prediction

An audio signal may be corrupted by periodic interference with no noise reference available. Such examples include the playback of speech or music with tape hum interference, turntable rumble, or vehicle engine or power line interference. We can use the modified line enhancement structure as shown in Figure 10.18.

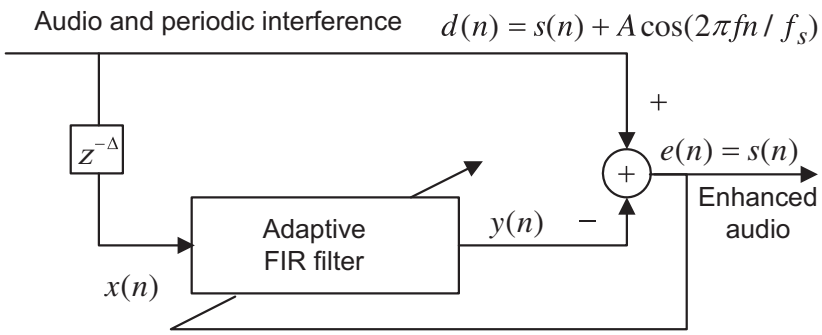

FIGURE 10.18

Canceling periodic interference using the adaptive filter.

The adaptive filter uses the delayed version of the corrupted signal $x(n)$ to predict the periodic interference. The number of delayed samples is selected through experiments that determine the performance of the adaptive filter. Note that a two-tap FIR adaptive filter can predict one sinusoid, as noted earlier. After convergence, the adaptive filter would predict the interference as

$$y(n) = \sum_{i=0}^{N-1} w(i)x(n-i) \approx A\cos(2\pi fn/f_s) \tag{10.16}$$

Therefore, the error signal contains only the desired audio signal

$$e(n) \approx s(n) \tag{10.17}$$

10.4.2 Electrocardiography Interference Cancellation

As we discussed in Chapters 1 and 8, in recording of electrocardiograms (ECG), there often exists unwanted 60-Hz interference, along with its harmonics, in the recorded data. This interference comes

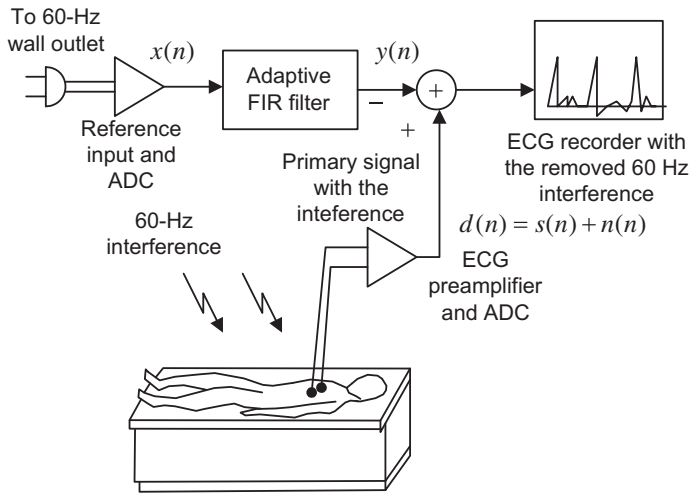

FIGURE 10.19

Illustration of canceling 60-Hz interference in ECG.

from the power line, including effects from magnetic induction, displacement currents in leads or in the body of the patient, and equipment interconnections and imperfections.

Figure 10.19 illustrates the application of adaptive noise canceling in ECG. The primary input is taken from the ECG preamplifier, while a 60-Hz reference input is taken from a wall outlet with proper attenuation. After proper signal conditioning, the digital interference $x(n)$ is acquired by the digital signal (DS) processor. The digital adaptive filter uses this reference input signal to produce an estimate, which approximates the 60-Hz interference $n(n)$ sensed from the ECG amplifier:

$$y(n) \approx n(n) \tag{10.18}$$

Here, an FIR adaptive filter with N taps and the LMS algorithm can be used for this application:

$$y(n) = w(0)x(n) + w(1)x(n-1) + \cdots + w(N-1)x(n-N+1) \tag{10.19}$$

Then after convergence of the adaptive filter, the estimated interference is subtracted from the primary signal of the ECG preamplifier to produce the output signal $e(n)$, in which the 60-Hz interference is cancelled:

$$e(n) = d(n) - y(n) = s(n) + n(n) - x(n) \approx s(n) \tag{10.20}$$

With enhanced ECG recording, doctors in clinics can give more accurate diagnoses for patients.

Canceling the maternal ECG in fetal monitoring is another important application. The block diagram is shown in Figure 10.20(a). Fetal ECG plays an important role in monitoring the condition of the baby before or during birth. However, the ECG acquired from the mother's abdomen is contaminated by noise such as muscle activity and fetal motion, as well as the mother's own ECG. In order to reduce the effect of the mother's ECG, four (or more) chest leads

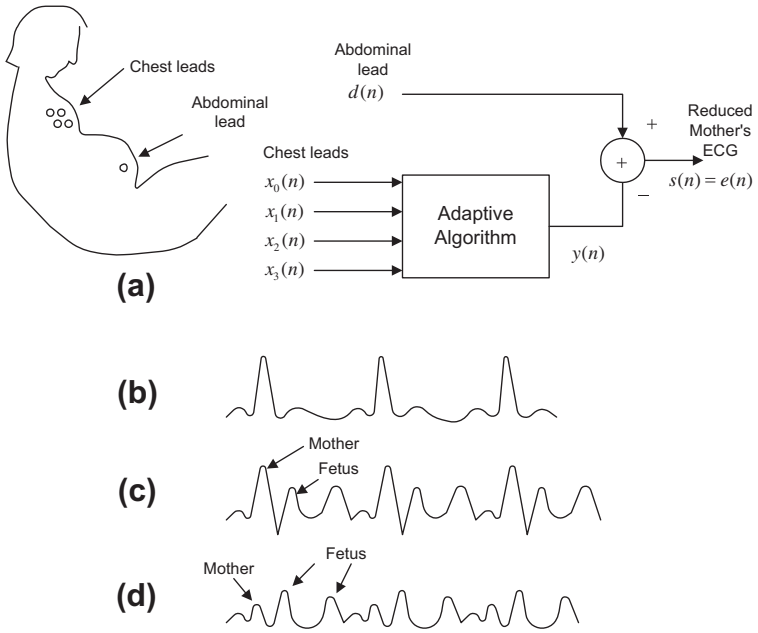

FIGURE 10.20

Canceling the maternal ECG in fetal monitoring.

(electrodes) are used to acquire the reference inputs: $x_0(n)$, $x_1(n)$, $x_2(n)$, and $x_3(n)$, with the assumption that these channels only contain the mother's ECG (see Figure 10.20(b)). One lead (electrode) placed on the mother's abdomen is used to capture the fetal information $d(n)$, which may be corrupted by the mother's ECG as shown in Figure 10.20(c). An adaptive filter uses its references to predict the mother ECG, which will be subtracted from the corrupted fetus signal. Then the fetal ECG with the reduced mother's ECG is obtained, as depicted in Figure 10.20(d). One possible LMS algorithm is listed below:

For $k = 0,1,2,3$

$$y_k(n) = w_k(0)x_k(n) + w_k(1)x_k(n-1) + \cdots + w_k(N-1)x_k(n-N+1)$$

$$y(n) = y_0(n) + y_1(n) + y_2(n) + y_3(n)$$

$$s(n) = e(n) = d(n) - y(n)$$

For $k = 0,1,2,3$

$$w_k(n-i) = w_k(n-i) + 2\mu e(n)x_k(n-i), \text{ for } i = 0, 1, \cdots N-1$$

10.4.3 Echo Cancellation in Long-Distance Telephone Circuits

Long-distance telephone transmission often suffers from impedance mismatches. This occurs primarily at the hybrid circuit interface. Balancing electric networks within the hybrid can never perfectly match the hybrid to the subscriber loop due to temperature variations, degradation of transmission lines, and so on. As a result, a small portion of the received signal is leaked for transmission. For example, in Figure 10.21A, if speaker B talks, the speech indicated as $x_B(n)$ will pass the transmission line to reach user A, and a portion of $x_B(n)$ at site A is leaked and transmitted back to the user B, forcing caller B to hear his or her own voice. This is known as an echo for speaker B. A similar echo illustration can be conducted for speaker A. When the telephone call is made over a long distance (more than 1,000 miles, such as geostationary satellites), the echo can be delayed by as much as 540 milliseconds. The echo impairment can be annoying to the customer and increases with distance.

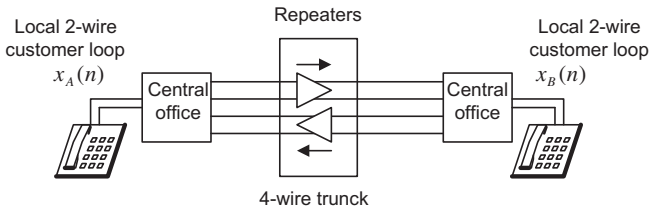

FIGURE 10.21A

Simplified long-distance circuit.

To circumvent the problem of echo in long-distance communications, an adaptive filter is applied at each end of the communication system, as shown in Figure 10.21B. Let us examine the adaptive filter installed at the speaker A site. The incoming signal is $x_B(n)$ from speaker B, while the outgoing signal contains the speech from the speaker A and a portion of leakage from the hybrid circuit

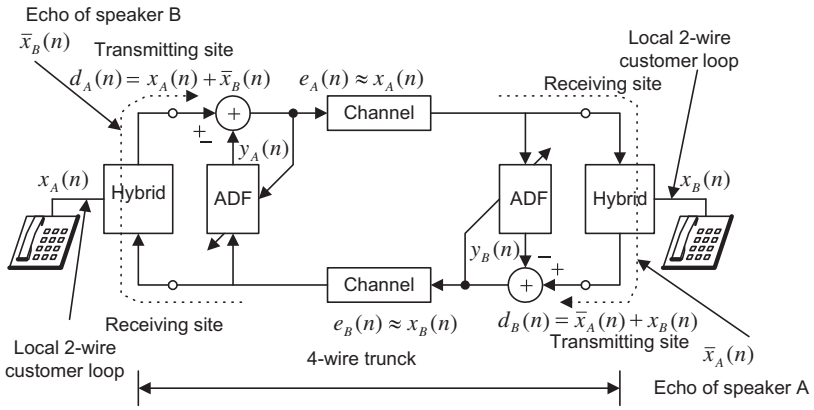

FIGURE 10.21B

Adaptive echo cancellers.

$d_A(n) = x_A(n) + \bar{x}_B(n)$. If the leakage $\bar{x}_B(n)$ returns back to speaker B, it becomes an annoying echo. To prevent the echo, the adaptive filter at the speaker A site uses the incoming signal from speaker B as an input and makes its output approximate to the leaked speaker B signal by adjusting its filter coefficients; that is,

$$y_A(n) = \sum_{i=0}^{N-1} w(i)x_B(n-i) \approx \bar{x}_B(n) \tag{10.21}$$

As shown in Figure 10.21B, the estimated echo $y_A(n) \approx \bar{x}_B(n)$ is subtracted from the outgoing signal, thus producing the signal that contains only speech A; that is, $e_A(n) \approx x_A(n)$. As a result, the echo of speaker B is removed. We can illustrate similar operations for the adaptive filter used at the speaker B site. In practice, an FIR adaptive filter with several hundred coefficients or more is commonly used to effectively cancel the echo. If nonlinearities are concerned in the echo path, a corresponding nonlinear adaptive canceller can be used to improve the performance of the echo cancellation.

Other forms of adaptive filters and other applications are beyond the scope of this book. The reader is referred to the references for further development.

10.5 LABORATORY EXAMPLES USING THE TMS320C6713 DSK

The implementation for system modeling in Section 10.4.3 is shown in Figure 10.22, where the input is fed from a function generator. The unknown system is a bandpass filter with a lower cutoff frequency of 1,400 Hz and upper cutoff frequency of 1,600 Hz. As shown in Figure 10.22, the left input channel (Left Line In [LCI]) is used for the input while the left output channel (Left Line Out [LCO]) and the right output channel (Right Line Out [RCO]) are designated as the system output and error output,

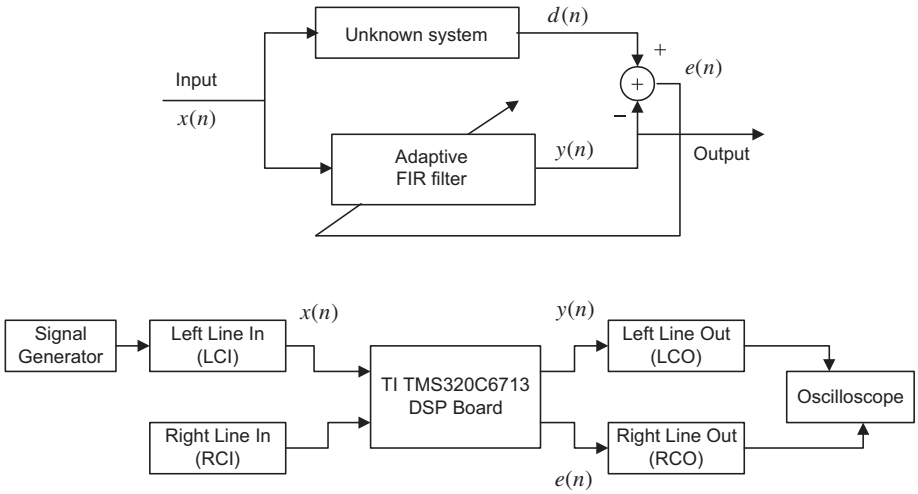

FIGURE 10.22

Setup for system modeling using the LMS adaptive filter.

respectively. Note that the right input channel (Right Line In [RCI]) is not used. When the input frequency is swept from 200 Hz to 3,000 Hz, the output shows a maximum peak when the input frequency is dialed to around 1,500 Hz. Hence, the adaptive filter acts like the unknown system. Program 10.4 gives the sample program segment.

Program 10.4. Program segment for system modeling.

```
/*Numerator coefficients */
/*for the bandpass filter (unknown system) with fL=1.4 kHz, fH=1.6 kHz*/
float b[5]={ 0.005542761540433, 0.000000000000002, -0.011085523080870,
     0.000000000000003 0.005542761540431};
/*Denominator coefficients */
/*for the bandpass filter (unknown system) with fL=1.4 kHz, fH=1.6 kHz*/
float a[5]={ 1.000000000000000, -1.450496619180500. 2.306093105231476,
     -1.297577189144526 0.800817049322883};
float x[40]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}; /*Reference input buffer*/
float w[40]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}; /*Adaptive filter coefficients*/
float d[5]={0.0,0.0,0.0,0.0,0.0}; /*Unknown system output */
float y[1]={0.0}; /*Adaptive filter output */
float e[1]={0.0}; /*Error signal */
float mu=0.000000000002; /*Adaptive filter convergence factor*/
interrupt void c_int11()
{
  float lc; /*left channel input */
  float rc; /*right channel input */
  float lcnew; /*left channel output */
  float rcnew; /*right channel output */
  int i;
//Left channel and right channel inputs
  AIC23_data.combo=input_sample();
  lc=(float) (AIC23_data.channel[LEFT]);
  rc= (float) (AIC23_data.channel[RIGHT]);
// Insert DSP algorithm below
  for(i=39;i>0;i--) /*Update the input buffer*/
  { x[i]=x[i-1]; }
  x[0]=lc;
  for(i=4; i>0; i--)
  { d[i]=d[i-1]; }
  d[0]=b[0]*x[0]+ b[1]*x[1]+ b[2]*x[2]+ b[3]*x[3]+ b[4]*x[4]
    -a[1]*d[1]-a[2]*d[2]- a[3]*d[3]- a[4]*d[4]; /*Unknown system output*/
// Adaptive filter
  y[0]=0;
  for(i=0;i<40; i++)
  { y[0]=y[0]+w[i]*x[i];}
  e[0]=d[0]-y[0]; /* Error output */
  for(i=0;i<40; i++)
  { w[i]=w[i]+2*mu*e[0]*x[i];} /* LMS algorithm */
```

```
// End of the DSP algorithm
  lcnew=y[0]; /* Send the tracked output */
  rcnew=e[0]; /* Send the error signal*/
  AIC23_data.channel[LEFT]=(short) lcnew;
  AIC23_data.channel[RIGHT]=(short) rcnew;
  output_sample(AIC23_data.combo);
}
```

With the advantage of the stereo input and output channels, we can conduct system modeling for an unknown analog system illustrated in Figure 10.23, where RCI is used to feed the unknown analog system output to the DSK. The program segment is listed in Program 10.5.

Program 10.5. Program segment for modeling an analog system.

```
float x[40]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}; /*Reference input buffer*/
float w[40]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}; /*Adaptive filter coefficients*/
float d[1]={0.0}; /*Unknown system output */
float y[1]={0,0}; /*Adaptive filter output */
float e[1]={0.0}; /*Error signal */
float mu=0.000000000002; /*Adaptive filter convergence factor*/
interrupt void c_int11()
{
  float lc; /*left channel input */
  float rc; /*right channel input */
  float lcnew; /*left channel output */
  float rcnew; /*right channel output */
  int i;
//Left channel and right channel inputs
  AIC23_data.combo=input_sample();
  lc=(float) (AIC23_data.channel[LEFT]);
  rc= (float) (AIC23_data.channel[RIGHT]);
// Insert DSP algorithm below
  for(i=39;i>0;i--) /*Update the input buffer*/
```

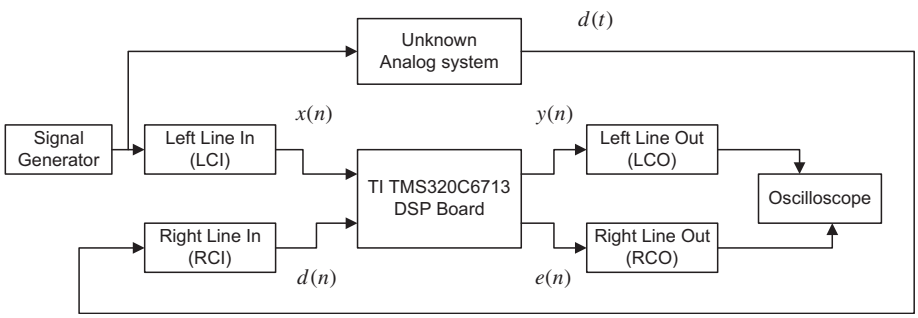

FIGURE 10.23

System modeling using an LMS adaptive filter.

```
  { x[i]=x[i-1]; }
  x[0]=lc;
  d[0]=rc; /*Unknown system output*/
// Adaptive filter
  y[0]=0;
  for(i=0;i<40; i++)
  { y[0]=y[0]+w[i]*x[i];}
  e[0]=d[0]-y[0]; /* Error output */
  for(i=0;i<40; i++)
  { w[i]=w[i]+2*mu*e[0]*x[i];} /* LMS algorithm */
// End of the DSP algorithm
  lcnew=y[0]; /* Send the tracked output */
  rcnew=e[0]; /* Send the error signal*/
  AIC23_data.channel[LEFT]=(short) lcnew;
  AIC23_data.channel[RIGHT]=(short) rcnew;
  output_sample(AIC23_data.combo);
}
```

Figure 10.24A shows an example of a tonal noise reduction system, and Figure 10.24B shows the details of the adaptive noise cancellation. The first DSP board is used to create the real-time corrupted signal, which is obtained by mixing the mono audio source (Left Line In [LCI1]) from any audio device and the tonal noise (Right Line In [RCI1]) generated from a function generator. The output (Left line Out [LCO1]) is the corrupted signal, which is fed to the second DSP board for noise cancellation application. The adaptive FIR filter in the second DSP board uses the reference input (Right Line In [RCI2]) to generate the output, which is used to cancel the tonal noise embedded in the corrupted signal (Left Line In [LCI2]). The output (Left Line Out [LCO2]) produces the clean mono audio signal (Jiang and Tan, 2012). Program 10.6 details the implementation.

Program 10.6. Program segments for noise cancellation.

(a) Program segment for DSK 1 (generation of the corrupted signal).

```
float x[1]={0.0}; /* Tonal reference noise */
Float s[1]={0,0}; /* Audio signal */
float d[1]={0.0}; /* Corrupted signal*/
```

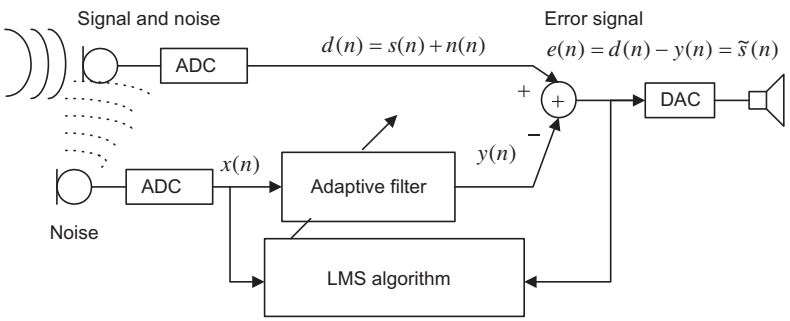

FIGURE 10.24A

Block diagram for tonal noise cancellation.

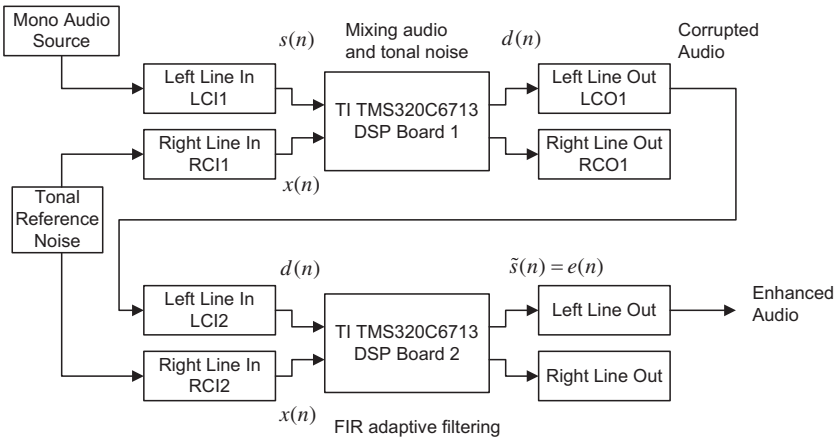

FIGURE 10.24B

Tonal noise cancellation with the adaptive filter.

```
interrupt void c_int11()
{
  float lc; /*left channel input */
  float rc; /*right channel input */
  float lcnew; /*left channel output */
  float rcnew; /*right channel output */
  int i;
//Left channel and right channel inputs
 AIC23_data.combo=input_sample();
 lc=(float) (AIC23_data.channel[LEFT]);
 rc= (float) (AIC23_data.channel[RIGHT]);
// Insert DSP algorithm below
  s[0]=lc;
  x[0]=rc;
  d[0]=s[0]+x[0];
// End of the DSP algorithm
  lcnew=d[0]; /* Send to DAC */
  rcnew=rc; /* keep the original data */
  AIC23_data.channel[LEFT]=(short) lcnew;
  AIC23_data.channel[RIGHT]=(short) rcnew;
  output_sample(AIC23_data.combo);
}
```

(b) Program segment for DSK 2 (LMS adaptive filter).

```
float x[20]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}; /*Reference input buffer*/
float w[20]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}; /*Adaptive filter
coefficients*/
float d[1]={0.0}; /* Corrupted signal*/
float y[1]={0,0}; /* Adaptive filter output */
```

```
float e[1]={0.0}; /* Enhanced signal */
float mu=0.000000000004; /*Adaptive filter convergence factor*/
interrupt void c_int11()
{
  float lc; /*left channel input */
  float rc; /*right channel input */
  float lcnew; /*left channel output */
  float rcnew; /*right channel output */
  int i;
//Left channel and right channel inputs
 AIC23_data.combo=input_sample();
 lc=(float) (AIC23_data.channel[LEFT]);
 rc= (float) (AIC23_data.channel[RIGHT]);
// Insert DSP algorithm below
 d[0]=lc; /*Corrupted signal*/
 for(i=19;i>0;i--) /*Update the reference noise buffer input buffer*/
 { x[i]=x[i-1]; }
 x[0]=rc;
// Adaptive filter
 y[0]=0;
 for(i=0;i<20; i++)
 { y[0]=y[0]+w[i]*x[i];}
 e[0]=d[0]-y[0]; /* Enhanced output */
 for(i=0;i<20; i++)
 { w[i]=w[i]+2*mu*e[0]*x[i]; }/* LMS algorithm */
// End of the DSP algorithm
  lcnew=e[0]; /* Send to DAC */
  rcnew=rc; /* keep the original data */
  AIC23_data.channel[LEFT]=(short) lcnew;
  AIC23_data.channel[RIGHT]=(short) rcnew;
  output_sample(AIC23_data.combo);
}
```

Many other practical configurations can be implemented similarly.

10.6 SUMMARY

1. Adaptive filters can be applied to signal-changing environments, spectral overlap between noise and signal, and unknown, or time-varying, noise.
2. Wiener filter theory provides optimal weight solutions based on statistics. It involves collection of a large block of data, calculation of an autocorrelation matrix and a cross-correlation matrix, and inversion of a large autocorrelation matrix.
3. The steepest decent algorithm can find the optimal weight solution using an iterative method, so a large matrix inversion is not needed. But it still requires calculating an autocorrelation matrix and cross-correlation matrix.
4. The LMS is a sample-based algorithm, which does not need collection of data or computation of statistics and does not involve matrix inversion.

5. The convergence factor for the LMS algorithm is bounded by the reciprocal of the product of the number of filter coefficients and input signal power.
6. The LMS adaptive FIR filter can be effectively applied for noise cancellation, system modeling, and line enhancement.
7. Further exploration includes other applications such as cancellation of periodic interference, biomedical ECG signal enhancement, and adaptive telephone echo cancellation.

10.7 PROBLEMS

10.1. Given a quadratic MSE function for the Wiener filter

$$J = 50 - 40w + 10w^2$$

find the optimal solution for w^* to achieve the minimum MSE $J_{\min}$ and determine $J_{\min}$.

10.2. Given a quadratic MSE function for the Wiener filter

$$J = 15 + 20w + 10w^2$$

find the optimal solution for w^* to achieve the minimum MSE $J_{\min}$ and determine $J_{\min}$.

10.3. Given a quadratic MSE function for the Wiener filter

$$J = 100 + 20w + 2w^2$$

find the optimal solution for w^* to achieve the minimum MSE $J_{\min}$ and determine $J_{\min}$.

10.4. Given a quadratic MSE function for the Wiener filter

$$J = 10 - 30w + 15w^2$$

find the optimal solution for w^* to achieve the minimum MSE $J_{\min}$ and determine $J_{\min}$.

10.5. Given a quadratic MSE function for the Wiener filter

$$J = 50 - 40w + 10w^2$$

use the steepest descent method with an initial guess of $w_0 = 0$ and a convergence factor $\mu = 0.04$ to find the optimal solution for w^* and determine $J_{\min}$ by iterating three times.

10.6. Given a quadratic MSE function for the Wiener filter

$$J = 15 + 20w + 10w^2$$

use the steepest descent method with an initial guess of $w_0 = 0$ and a convergence factor $\mu = 0.04$ to find the optimal solution for w^* and determine $J_{\min}$ by iterating three times.

10.7. Given a quadratic MSE function for the Wiener filter

$$J = 100 + 20w + 2w^2$$

use the steepest descent method with an initial guess of $w_0 = -4$ and a convergence factor $\mu = 0.2$ to find the optimal solution for w^* and determine $J_{\min}$ by iterating three times.

10.8. Given a quadratic MSE function for the Wiener filter

$$J = 10 - 30w + 15w^2$$

use the steepest descent method with an initial guess of $w_0 = 2$ and a convergence factor $\mu = 0.02$ to find the optimal solution for w^* and determine $J_{\min}$ by iterating three times.

10.9. Consider the following DSP system used for noise cancellation applications (Figure 10.25), in which $d(0) = 3, d(1) = -2, d(2) = 1, x(0) = 3, x(1) = -1, x(2) = 2$, and there is an adaptive filter with two taps $y(n) = w(0)x(n) + w(1)x(n-1)$ with initial values $w(0) = 0, w(1) = 1$, and $\mu = 0.1$,

a. Determine the LMS algorithm equations

$$\begin{aligned} y(n) &= \\ e(n) &= \\ w(0) &= \\ w(1) &= \end{aligned}$$

b. Perform adaptive filtering for each $n = 0, 1, 2$.

10.10. Given a DSP system with a sampling rate of 8,000 samples per second, implement an adaptive filter with five taps for system modeling.

As shown in Figure 10.26, assume the unknown system transfer function is

$$H(z) = \frac{0.25 + 0.25z^{-1}}{1 - 0.5z^{-1}}$$

Determine the DSP equations

$$\begin{aligned} y(n) &= \\ e(n) &= \\ w(i) &= \end{aligned}$$

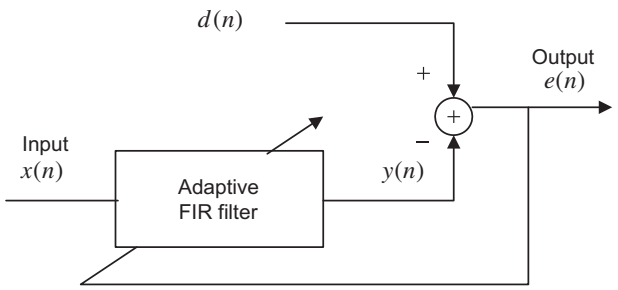

FIGURE 10.25

Noise cancellation in Problem 10.9.

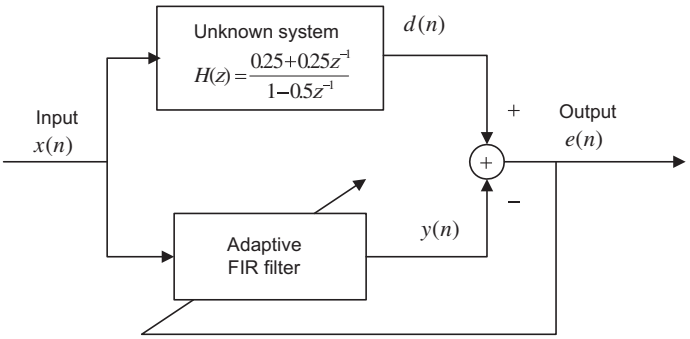

FIGURE 10.26

System modeling in Problem 10.10.

using the LMS algorithm for $i = 0, 1, 2, 3, 4$; that is, write the equations for all adaptive coefficients:

$$w(0) =$$
$$w(1) =$$
$$w(2) =$$
$$w(3) =$$
$$w(4) =$$

10.11. Consider the adaptive filter used for the noise cancellation application in Problem 10.9, in which $d(0) = 3$, $d(1) = -2$, $d(2) = 1$, $x(0) = 3$, $x(1) = -1$, $x(2) = 2$, and an adaptive filter with three taps $y(n) = w(0)x(n) + w(1)x(n-1) + w(2)x(n-2)$ with initial values $w(0) = 0$, $w(1) = 0$, $w(2) = 0$ and $\mu = 0.2$.

a. Determine the LMS algorithm equations

$$y(n) =$$
$$e(n) =$$
$$w(0) =$$
$$w(1) =$$
$$w(2) =$$

b. Perform adaptive filtering for each of $n = 0, 1, 2$.

10.12. Consider the DSP system with a sampling rate of 8,000 samples per second in Problem 10.10. Implement an adaptive filter with five taps for system modeling, assuming the unknown system transfer function is

$$H(z) = 0.2 + 0.3z^{-1} + 0.2z^{-2}$$

Determine the DSP equations

$$y(n) =$$
$$e(n) =$$
$$w(i) =$$

using the LMS algorithm for $i = 0, 1, 2, 3, 4$; that is, write the equations for all adaptive coefficients:

$$w(0) =$$
$$w(1) =$$
$$w(2) =$$
$$w(3) =$$
$$w(4) =$$

10.13. Consider the DSP system set up for noise cancellation applications with a sampling rate of 8,000 Hz shown in Figure 10.27. The desired 1,000 Hz tone is generated internally via a tone generator, and the generated tone is corrupted by the noise captured from a microphone. An FIR adaptive filter with 25 taps is applied to reduce the noise in the corrupted tone.

a. Determine the DSP equation for the channel noise $n(n)$.

b. Determine the DSP equation for signal tone $yy(n)$.

c. Determine the DSP equation for the corrupted tone $d(n)$.

d. Set up the LMS algorithm for the adaptive FIR filter.

10.14. Consider the DSP system for noise cancellation applications with two taps in Figure 10.28.

a. Set up the LMS algorithm for the adaptive filter.

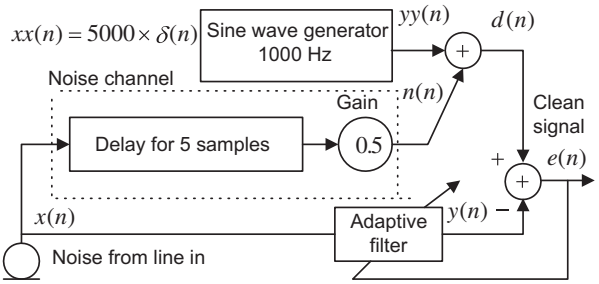

FIGURE 10.27

Noise cancellation in Problem 10.13.

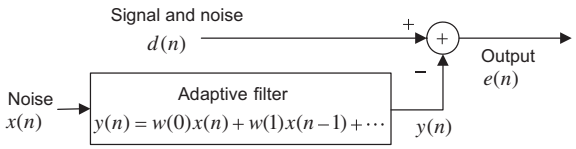

FIGURE 10.28

Noise cancellation in Problem 10.14.

b. Assume the inputs and outputs $x(0) = 1, x(1) = 1, x(2) = -1, x(3) = 2, d(0) = 0, d(1) = 2, d(2) = -1$, and $d(3) = 1$; initial weights $w(0) = w(1) = 0$; and convergence factor $\mu = 0.1$. Perform adaptive filtering to obtain outputs $e(n)$ for $n = 0, 1, 2$.

10.15. Again consider the DSP system for noise cancellation applications with three taps in Figure 10.28.

a. Set up the LMS algorithm for the adaptive filter.

b. Assume the inputs and outputs $x(0) = 1, x(1) = 1, x(2) = -1, x(3) = 2, d(0) = 0, d(1) = 2, d(2) = -1$, and $d(3) = 1$; initial weights $w(0) = w(1) = w(2) = 0$, and convergence factor $\mu = 0.1$. Perform adaptive filtering to obtain outputs $e(n)$ for $n = 0, 1, 2$.

10.16. For a line enhancement application using the FIR adaptive filter depicted in Figure 10.29.

a. Set up the LMS algorithm for the adaptive filter using two filter coefficients and delay $\Delta = 2$.

b. Assume the inputs and outputs $d(0) = -1, d(1) = 1, d(2) = -1, d(3) = 1, d(4) = -1, d(5) = 1$ and $d(6) = -1$; initial weights $w(0) = w(1) = 0$; and convergence factor $\mu = 0.1$. Perform adaptive filtering to obtain outputs $y(n)$ for $n = 0, 1, 2, 3, 4$.

10.17. Repeat Problem 10.16 using a three-tap FIR filter and $\Delta = 3$.

10.18. An audio playback application is described in Figure 10.30.

Due to the interference environment, the audio is corrupted by 15 different periodic interferences. The DSP engineer uses an FIR adaptive filter to remove such interferences as shown in Figure 10.30.

a. What is the minimum number of filter coefficients?

b. Set up the LMS algorithm for the adaptive filter using the number of taps obtained in (a).

10.19. Repeat Problem 10.18 for corrupted audio that contains five different periodic interferences.

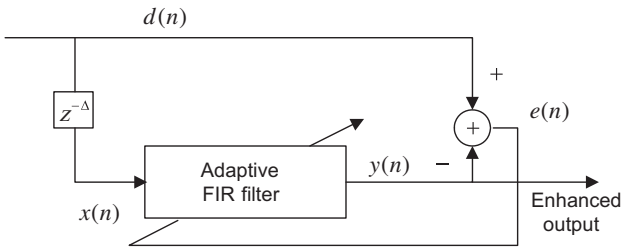

FIGURE 10.29

Line enhancement in Problem 10.16.

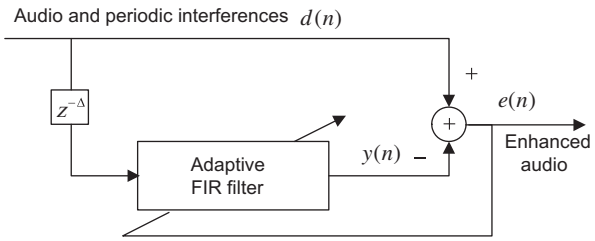

FIGURE 10.30

Interference cancellation in Problem 10.18.

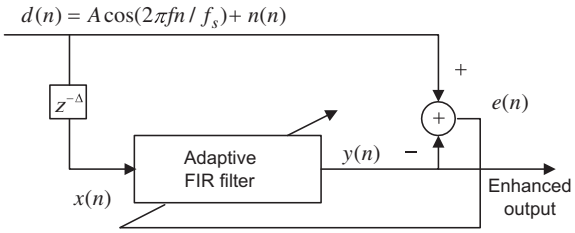

FIGURE 10.31

A line enhancement system in Problem 10.26

10.20. In a noisy ECG acquisition environment, the DSP engineer uses an adaptive FIR filter with 20 coefficients to remove 60-Hz interference. The system is set up as shown in Figure 10.19, where the corrupted ECG and enhanced ECG are represented as $d(n)$ and $e(n)$, respectively; $x(n)$ is the captured reference signal from the 60-Hz interference; and $y(n)$ is the adaptive filter output. Determine all difference equations to implement the adaptive filter.

10.21. Given an application of the echo cancellation shown in Figure 10.21B, determine all difference equations to implement the adaptive filter with four adaptive coefficients at the speaker A site.

10.22. Given an application of the echo cancellation shown in Figure 10.21B,

a. explain the concepts and benefits using the echo canceller;

b. explain the operations of the adaptive filter at the speaker B site;

c. determine all difference equations to implement the adaptive filter at the speaker A site.

10.7.1 Computer Problems with MATLAB

Use MATLAB to solve Problems 10.23 to 10.26.

10.23. Write a MATLAB program for minimizing the two-weight MSE (mean squared error) function

$$J = 100 + 100w_1^2 + 4w_2^2 - 100w_1 - 8w_2 + 10w_1w_2$$

by applying the steepest descent algorithm for 500 iterations. The derivatives are

$$\frac{dJ}{dw_1} = 200w_1 - 100 + 10w_2 \quad \text{and} \quad \frac{dJ}{dw_2} = 8w_2 - 8 + 10w_1$$

and the initial weights are assumed as $w_1(0) = 0, w_2(0) = 0, \mu = 0.001$. Plot $w_1(k)$, $w_2(k)$, and $J(k)$ versus the number of iterations, respectively. Summarize your results.

10.24. In Problem 10.10, the unknown system is assumed to be a fourth-order Butterworth bandpass filter with a lower cutoff frequency of 700 Hz and an upper cutoff frequency of 900 Hz. Design a bandpass filter by the bilinear transformation method for simulating the unknown system with a sampling rate of 8,000 Hz.

a. Generate the input signal for 0.1 second using a sum of three sinusoids with 100 Hz, 800 Hz, and 1,500 Hz and a sampling rate of 8,000 Hz.

b. Use the generated input as the unknown system input to produce the system output. The adaptive FIR filter is then applied to model the designed bandpass filter. The following parameters are assumed:

Adaptive FIR filter
Number of taps: 15 coefficients
Algorithm: LMS algorithm
Convergence factor: 0.01

c. Implement the adaptive FIR filter, and plot the system input, system output, adaptive filter output, and error signal, respectively.

d. Plot the input spectrum, system output spectrum, and adaptive filter output spectrum, respectively.

10.25. Use the following MATLAB code to generate reference noise and a signal of 300 Hz corrupted by the noise with a sampling rate of 8,000 Hz.

```
fs=8000; T=1/fs;                              % Sampling rate and sampling period

t=0:T:1;                                      % Create time instants

x=randn(1,length(t));                         % Generate reference noise

n=filter([ 0 0 0 0 0 0 0 0 0 0.8 ],1,x);% Generate the corruption noise

d=sin(2*pi*300*t)+n;                          % Generate the corrupted signal
```

a. Implement an adaptive FIR filter to remove the noise. The adaptive filter specifications are as follows:

Sample rate = 8,000 Hz
Signal corrupted by Gaussian noise delayed by nine samples from the reference noise
Reference noise: Gaussian noise with a power of 1
Number of FIR filter taps: 16
Convergence factor for the LMS algorithm: 0.01

b. Plot the corrupted signal, reference noise, and enhanced signal, respectively.

c. Compare the spectral plots between the corrupted signal and the enhanced signal.

10.26. A line enhancement system (Figure 10.31) has following specifications:

Sampling rate = 1,000 Hz
Corrupted signal: 100 Hz tone with the unit amplitude added with the unit power Gaussian noise
Adaptive filter: FIR type, 16 taps
Convergence factor: 0.001
Delay value Δ: to be decided according to the experiment
LMS algorithm

a. Write a MATLAB program to perform the line enhancement for a one-second corrupted signal. Run the developed program with a trail of delay value Δ and plot the noisy signal, the enhanced signals, and their spectra.

b. Run your simulation to find the delay value Δ that achieves the largest noise reduction.

10.7.2 MATLAB Projects

10.27. Active noise control:

The ANC (active noise control) system is based on the principle of superposition of the primary noise source and secondary source with its acoustic output being the same amplitude but the opposite phase of the primary noise source, as shown in Figure 10.32. The primary noise is captured using the reference microphone, which is located close to the noise source. The ANC system uses the sensed reference signal $x(n)$ to generate a canceling signal $y(n)$, which drives the secondary speaker to destructively attenuate the primary noise. An error microphone is used to detect the residue noise $e(n)$, which is fed back to the ANC system to monitor the system performance. The residue noise $e(n)$ together with the reference signal $x(n)$ are used by the linear adaptive controller whose coefficients are adjusted via an adaptive algorithm to minimize the measured error signal $e(n)$, or the residue acoustic noise. $P(z)$ designates the physical primary path between the reference sensor and the error sensor, and $S(z)$ designates the physical secondary path between the ANC adaptive filter output and the error sensor. To control the noise at the cancelling point, the instantaneous power $e^2(n)$ must be minimized. Note that

$$E(z) = D(z) - Y(z)S(z) = D(z) - [W(z)X(z)]S(z)$$

where $W(z)$ denotes the adaptive control filter. Exchange of the filter order (since the filters are linear filters) gives

$$E(z) = D(z) - W(z)[S(z)X(z)] = D(z) - W(z)U(z)$$

Assuming that $\overline{S}(z)$ is the secondary path estimate and noticing that $U(z) = \overline{S}(z)X(z)$ and $Y(z) = W(z)X(z)$ are the filtered reference signal and adaptive filter output, applying the LMS algorithm gives the filtered-x LMS algorithm

$$w(i) = w(i) + 2\mu e(n)u(n-i)$$

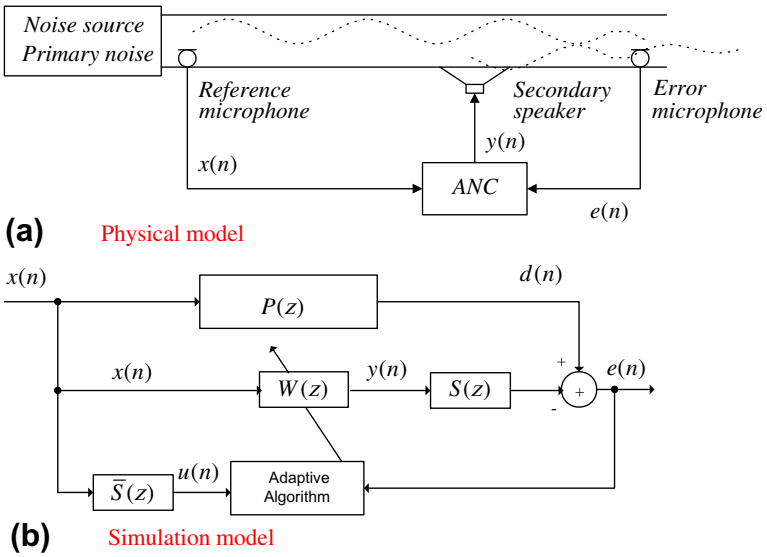

FIGURE 10.32

An active noise control system.

Note that $e(n)$ is measured from the error microphone.

The completed filtered-x LMS algorithm is summarized below:

1. Initialize $w(0), w(1), \ldots w(N-1)$ to arbitrary values.

2. Read $x(n)$ and perform digital filtering:

$$y(n) = w(0)x(n) + w(1)x(n-1) + \cdots + w(N-1)x(n-N+1)$$

3. Compute the filtered inputs:

$$u(n) = \overline{s}(0)x(n) + \overline{s}(1)x(n-1) + \cdots + \overline{s}(M-1)x(n-M+1)$$

4. Read $e(n)$ and update each filter coefficient:

$$\text{for } i = 0, \cdots, N-1,\ w(i) = w(i) + 2\mu e(n)u(n-i),$$

Assuming the following:

Sampling rate = 8,000 Hz and simulation duration = 10 seconds
Primary noise: $x(n)$ = 500-Hz sine wave
Primary path: $P(z) = 0.2 + 0.25z^{-1} + 0.2z^{-2}$
Secondary path: $S(z) = 0.25 + 0.2z^{-1}$
Secondary path estimate: $\overline{S}(z) = S(z) = 0.2 + 0.2z^{-1}$ (can have slight error as compared to $S(z)$)

Residue error signal: $e(n) = d(n) -$ filtering $y(n)$ using coefficients $s(n) = d(n) - y(n)*$ $s(n)$, where the symbol "*" denotes the filter convolution.
Implement the ANC system and plot the residue sensor signal to verify the effectiveness. The primary noise at cancelling point $d(n)$, and filtered reference signal $u(n)$ can be generated in MATLAB as follows:

```
d = filter([0.2 0.25 0.2],1,x); % Simulate physical media
u=filter([0.2 0.2],1,x);
```

The residue error signal $e(n)$ should be generated sample by sample and embedded into the adaptive algorithm, that is,

```
e(n)=d(n)-(s(1)*y(n)+s(2)*y(n-1)); % Simulate the residue error
```

where s(1) = 0.25 and s(2) = 0.2. Details of active control systems can be found in the textbook by Kuo and Morgan (1996).

10.28. Frequency tracking:

An adaptive filter can be applied for real-time frequency tracking (estimation). In this application, a special second notch IIR filter structure, as shown in Figure 10.33, is preferred for simplicity.

The notch filter transfer function

$$H(z) = \frac{1 - 2\cos(\theta)z^{-1} + z^{-2}}{1 - 2r\cos(\theta)z^{-1} + r^2z^{-2}}$$

has only one adaptive parameter θ. It has two zeros on the unit circle resulting in an infinite-depth notch. The parameter r controls the notch bandwidth. It requires $0 << r < 1$ for achieving a narrowband notch. When r is close to 1, the 3-dB notch filter bandwidth can be approximated as $BW \approx 2(1 - r)$ (see Chapter 8). The input sinusoid whose frequency f needs to be estimated and tracked is given below:

$$x(n) = A\cos(2\pi fn/f_s + \alpha)$$

where A and α are the amplitude and phase angle. The filter output is expressed as

$$y(n) = x(n) - 2\cos[\theta(n)]x(n-1) + x(n-2) + 2r\cos[\theta(n)]y(n-1) - r^2y(n-2)$$

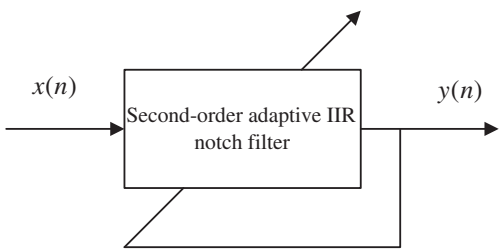

FIGURE 10.33

A frequency tracking system.

The objective is to minimize the filter instantaneous output power $y^2(n)$. Once the output power is minimized, the filter parameter $\theta = 2\pi f/f_s$ will converge to its corresponding frequency f Hz. The LMS algorithm to minimize the instantaneous output power $y^2(n)$ is given as

$$\theta(n+1) = \theta(n) - 2\mu y(n)\beta(n),$$

where the gradient function $\beta(n) = \partial y(n)/\partial\theta(n)$ can be derived as follows:

$$\beta(n) = 2\sin[\theta(n)]x(n-1) - 2r\sin[\theta(n)]y(n-1) + 2r\cos[\theta(n)]\beta(n-1) - r^2\beta(n-2)$$

μ is the convergence factor which controls speed of algorithm convergence.

In this project, plot and verify the notch frequency response by setting $f_s = 8{,}000$ Hz, $f = 1{,}000$ Hz, and $r = 0.95$. Then generate the sinusoid with a duration of 10 seconds, frequency of 1,000 Hz, and amplitude of 1. Implement the adaptive algorithm using an initial guess $\theta(0) = 2\pi \times 2000/f_s = 0.5\pi$ and plot the tracked frequency $f(n) = \theta(n)f_s/2\pi$ for tracking verification.

Notice that this particular notch filter only works for a single frequency tracking, since the mean squares error function $E[y^2(n)]$ has one global minimum (one best solution when the LMS algorithm converges). Details of the adaptive notch filter can be found in Tan and Jiang (2012). Notice that the general IIR adaptive filter suffers from local minima, that is, the LMS algorithm converges to local minimum and the nonoptimal solution results.