CHAPTER

ADAPTIVE FILTERS AND APPLICATIONS

9

CHAPTER OUTLINE

9.1 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 (ANC), and adaptive control systems. Adaptive filters work generally for the 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 the fixed filter coefficients will fail to preserve the desired signal spectrum, as shown in Fig. 9.1.

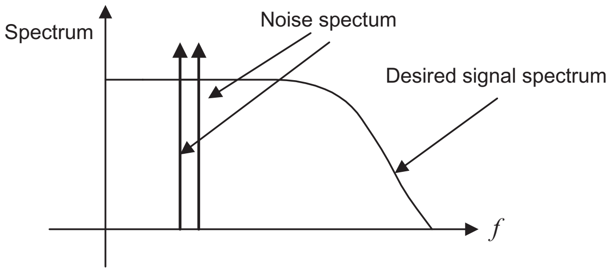


FIG. 9.1

Spectrum illustration for using adaptive filters.

However, an adaptive filter will do the job. Note that adaptive filtering, with its applications, has existed more than two decades in the research community and is still active. This chapter introduces some fundamentals of the subject, adaptive finite impulse response (FIR) filters with a simple and popular least mean square (LMS) algorithm and recursive least squares (RLS) 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 (2014), Stearns and Hush (2011), 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 Fig. 9.2.

As shown in Fig. 9.2, 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 captures noise $x(n)$, which is fed to the adaptive filter.

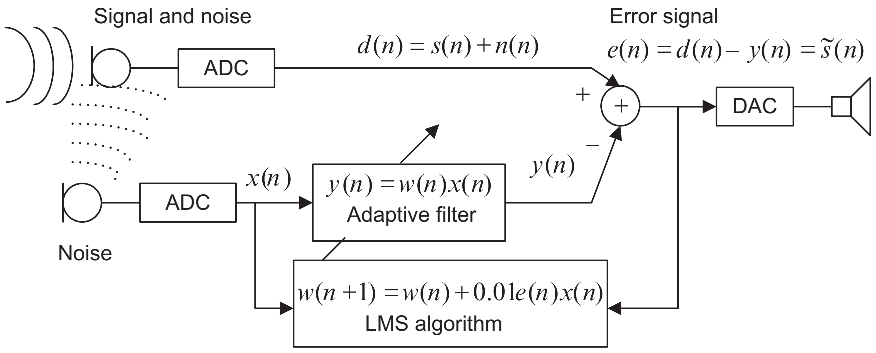


FIG. 9.2

Simplest noise canceller using a one-tap 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 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:

$$w(n+1)=w(n)+0.01\times e(n)\times x(n),$$

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 Fig. 9.2, the LMS algorithm has the initial coefficient set to be $w(0)=0.3$ and leads to

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

The corrupted signal is generated by adding noise to a sine wave. The corrupted signal and noise reference are shown in Fig. 9.3, and their first 16 values are listed in Table 9.1.

Let us perform adaptive filtering for several samples using the values for the corrupted signal and reference noise in Table 9.1. We see that

$$\begin{aligned}
n=0,\ & y(0)=w(0)x(0)=0.3\times(-0.5893)=-0.1768\\
& e(0)=d(0)-y(0)=-0.2947-(-0.1768)=-0.1179=\tilde{s}(0)\\
& w(1)=w(0)+0.01e(0)x(0)=0.3+0.01\times(-0.1179)\times(-0.5893)=0.3007\\
n=1,\ & y(1)=w(1)x(1)=0.3007\times 0.5893=0.1772\\
& e(1)=d(1)-y(1)=1.0017-0.1772=0.8245=\tilde{s}(1)\\
& w(2)=w(1)+0.01e(1)x(1)=0.3007+0.01\times 0.8245\times 0.5893=0.3056\\
n=2,\ & y(2)=w(2)x(2)=0.3056\times 3.1654=0.9673\\
& e(2)=d(2)-y(2)=2.5827-0.9673=1.6155=\tilde{s}(2)\\
& w(3)=w(2)+0.01e(2)x(2)=0.3056+0.01\times 1.6155\times 3.1654=0.3567\\
n=3,\ & \cdots
\end{aligned}$$

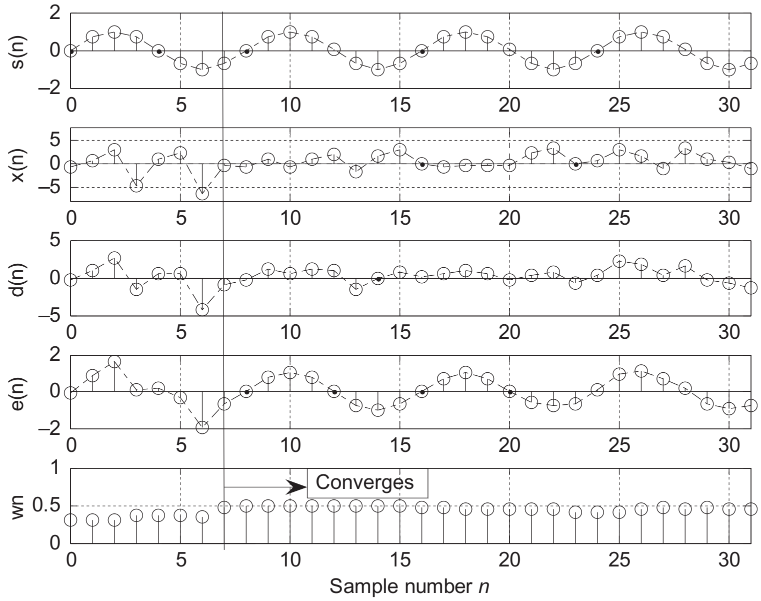


FIG. 9.3

Original signal $s(n)$, reference noise $x(n)$, corrupted signal $d(n)$, enhanced signal $e(n)$, and adaptive coefficient $w(n)$ in the noise cancellation.

Table 9.1 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	0.3
1	1.0017	0.5893	0.8245	0.7071	0.3007
2	2.5827	3.1654	1.6155	1	0.3056
3	−1.6019	−4.6179	0.0453	0.7071	0.3567
4	0.5622	1.1244	0.1635	0	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	0.48
9	1.2274	1.0406	0.7278	0.7071	0.4802
10	0.6021	−0.7958	0.9902	1	0.4877
11	1.1647	0.9152	0.7255	0.7071	0.4799
12	0.963	1.926	0.026	0	0.4865
13	−1.5065	−1.5988	−0.7279	−0.7071	0.487
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 9.1. Fig. 9.3 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 Fig. 9.3, 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 9.1.

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:

$$y(n) = \sum_{k=0}^{N-1} w_k(n)x(n-k) = w_0(n)x(n) + w_1(n)x(n-1) + \cdots + w_{N-1}(n)x(n-N+1). \tag{9.1}$$

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

9.2 BASIC WIENER FILTER THEORY AND ADAPTIVE ALGORITHMS

In this section, we will first study Wiener filter theory and linear prediction. Then we will develop standard adaptive algorithms such as steepest decent algorithm, LMS algorithm, and RLS algorithm.

9.2.1 WIENER FILTER THEORY AND LINEAR PREDICTION

9.2.1.1 Basic Wiener Filter Theory

Many adaptive algorithms can be viewed as approximations of the discrete Wiener filter shown in Fig. 9.4, where the Wiener filter output $y(n)$ is a sum of its N weighted inputs, that is,

$$y(n) = w_0x(n) + w_1x(n-1) + \cdots + w_{N-1}x(n-N+1).$$

The Wiener filter adjusts its weight(s) to produce filter output $y(n)$, which would be as close as possible to the noise $n(n)$ contained in the corrupted signal $d(n)$. Hence, at the subtracted output, the noise is cancelled and the output $e(n)$ contains clean signal.

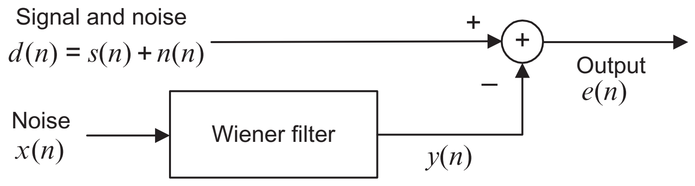


FIG. 9.4

Wiener filter for noise cancellation.

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

$$e(n) = d(n) - wx(n). \tag{9.2}$$

Now let us solve the best weight w^*. Taking the square of the output error leads to

$$e^2(n) = (d(n) - wx(n))^2 = d^2(n) - 2d(n)wx(n) + w^2x^2(n). \tag{9.3}$$

Taking the statistical expectation of Eq. (9.3), we have

$$E\left(e^2(n)\right) = E\left(d^2(n)\right) - 2wE(d(n)x(n)) + w^2E\left(x^2(n)\right). \tag{9.4}$$

Using the notations in statistics, we define

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

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 Eq. (9.4) as

$$J = \sigma^2 - 2wP + w^2R. \tag{9.5}$$

Since σ^2, P, and R are constants, J is a quadratic function of w which may be plotted in Fig. 9.5A.

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

$$\frac{dJ}{dw} = -2P + 2wR = 0. \tag{9.6}$$

Solving Eq. (9.6), we get the best weight solution as

$$w^* = R^{-1}P. \tag{9.7}$$

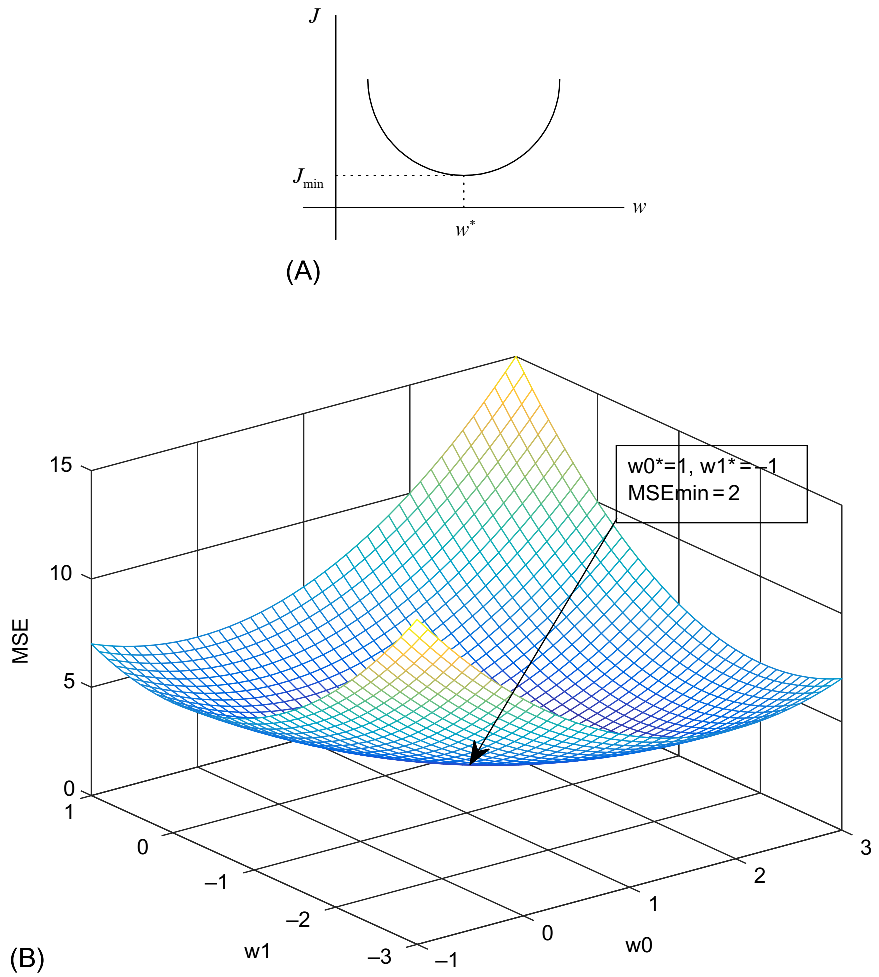


FIG. 9.5

(A) Mean square error quadratic function. (B) Plot of the MSE function vs. two weights.

EXAMPLE 9.1

Given a 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 the derivative of the MSE function and setting it to zero, we have

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

Continued

EXAMPLE 9.1—CONT'D

Solving the equation leads to

$$w^* = 1.$$

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

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

In general, the Wiener filter has N coefficients. The filter output from Eq. (9.1) in a matrix form can be written as

$$y(n) = \sum_{k=0}^{N-1} w_k x(n-k) = W^T X(n), \tag{9.8}$$

where $X(n) = [x(n)x(n-1)\cdots x(n-N+1)]^T$ and $W = [w_0 w_1 \cdots w_{N-1}]^T$ are the input and coefficient vectors, respectively. Taking this modification into the MSE function, it follows that

$$\begin{aligned} J &= E\{e^2(n)\} = E\left\{\left(d(n) - W^T X(n)\right)^2\right\} \\ &= E\{d^2(n)\} - 2W^T E\{X(n)d(n)\} + W^T E\{X^T(n)X(n)\}W. \end{aligned} \tag{9.9}$$

The MSE function of Eq. (9.9) can be rewritten as

$$J = \sigma^2 - 2W^T P + W^T RW, \tag{9.10}$$

where $R = E\{X^T(n)X(n)\}$ and $P = E\{X(n)d(n)\}$ are the autocorrelation matrix and cross-correlation vector, respectively. Setting the derivative of the MSE function over W to zero, we obtain

$$\frac{\partial J}{\partial W} = -2P + RW = 0. \tag{9.11}$$

Finally, the optimal solution is obtained by

$$W^* = R^{-1}P, \tag{9.12}$$

and the minimum MSE can be derived by substituting Eq. (9.12) into Eq. (9.10):

$$\begin{aligned} J_{\min} &= \sigma^2 - 2W^{*T}P + W^{*T}RW^* \\ &= \sigma^2 - 2W^{*T}P + W^{*T}RR^{-1}P \\ &= \sigma^2 - W^{*T}P. \end{aligned} \tag{9.13}$$

Next, we examine the MSE function assuming the following statistical data:

$$\begin{aligned} &\sigma^2 = E[d^2(n)] = 6,\ \ E[x^2(n)] = E[x^2(n-1)] = 1,\ \ E[x(n)x(n-1)] = 1/2, \\ &E[d(n)x(n)] = 1/2,\ \text{and}\ \ E[d(n)x(n)] = 1/2, \end{aligned}$$

for the two-tap adaptive filter $y(n) = w_0 x(n) + w_1 x(n-1)$. Notice that

$$R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \text{ and } P = \begin{bmatrix} 1/2 \\ -1/2 \end{bmatrix}.$$

Applying Eq. (9.12), we yield

$$\begin{bmatrix} w_0^* \\ w_1^* \end{bmatrix} = R^{-1}P = \begin{bmatrix} 1 & 1/2 \\ 1/2 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 1/2 \\ -1/2 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}.$$

Using Eq. (9.10), it follows that

$$\begin{aligned} J &= \sigma^2 - 2W^T P + W^T RW \\ &= 4 - 2[w_0 \quad w_1] \begin{bmatrix} 1/2 \\ -1/2 \end{bmatrix} + [w_0 \quad w_1] \begin{bmatrix} 1 & 1/2 \\ 1/2 & 1 \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} = 4 + w_0^2 + w_0 w_1 + w_1^2 - w_0 + w_1. \end{aligned}$$

Eq. (9.13) gives

$$J_{\min} = \sigma^2 - W^{*T} P = 4 - [2 \quad -2] \begin{bmatrix} 1/2 \\ -1/2 \end{bmatrix} = 2.$$

Fig. 9.5B shows the MSE function vs. the weights, where the optimal weights and the minimum MSE are $w_0^* = 1$, $w_1^* = -1$, and $J_{\min} = 2$. The plot also indicates that the MSE function is a quadratic function and that there exists only one minimum of the MSE surface.

Notice that a few points need to be clarified for Eq. (9.12):

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 sufficiently long. This will cause a long processing delay that will hinder real-time implementation.

9.2.1.2 Forward Linear Prediction

Linear prediction deals with the problem of predicting a future value of a stationary random process using the observed past values. Let us consider a one-step forward linear predictor in which the prediction of the value $x(n)$ by using a weighted linear combination of the past values, that is, $x(n-1)$, $x(n-2)$, …, $x(n-P)$. We will only focus on the treatment of the forward linear predictor, and others such as backward linear prediction, and forward and backward prediction can be found in Haykin (2014) and Proakis and Manolakis (2007). The output of the one-step forward linear predictor can be written as

$$\hat{x}(n) = -\sum_{k=1}^{P} a_k x(n-k), \tag{9.14}$$

where $-a_k$ are the prediction coefficients and P is the order of the forward linear predictor. Fig. 9.6 shows the block diagram of the one-step forward linear prediction.

As shown in Fig. 9.6, the difference between the value $x(n)$ and the predicted value $\hat{x}(n)$ is referred to as the forward prediction error, which is defined as

$$\begin{aligned} e_f(n) &= x(n) - \hat{x}(n) \\ &= x(n) + \sum_{k=1}^{P} a_k x(n-k). \end{aligned} \tag{9.15}$$

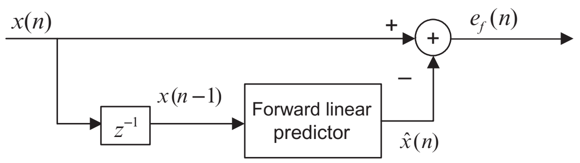


FIG. 9.6

One-step forward linear prediction.

We can minimize $E\{e_f^2(n)\}$ to find the prediction coefficients. Taking the derivative of $E\{e_f^2(n)\}$ to a_p, it follows that

$$\begin{aligned} E\left\{\frac{\partial e_f^2(n)}{\partial a_p}\right\} &= E\left\{e_f(k)\frac{\partial e_f(n)}{\partial a_p}\right\} \\ &= E\{e_f(k)x(n-p)\} \text{ for } p=1,2,\cdots,P \\ &= E\left\{\left[x(n)+\sum_{k=1}^{P} a_k x(n-k)\right]x(n-p)\right\}. \end{aligned} \tag{9.16}$$

Distribute the expectation in Eq. (9.16), we obtain

$$E\left\{\frac{\partial e_f^2(n)}{\partial a_p}\right\} = E\{x(n)x(n-p)\} + \sum_{k=1}^{P} a_k E\{x(n-k)x(n-p)\} \text{ for } p=1,2,\cdots,P. \tag{9.17}$$

Setting Eq. (9.17) to zero, we obtain

$$\begin{bmatrix} r_{xx}(0) & r_{xx}(-1) & \cdots & r_{xx}(-P+1) \\ r_{xx}(1) & r_{xx}(0) & \cdots & r_{xx}(-P+2) \\ \vdots & \vdots & \ddots & \vdots \\ r_{xx}(P-1) & r_{xx}(P-2) & \cdots & r_{xx}(0) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_P \end{bmatrix} = -\begin{bmatrix} r_{xx}(1) \\ r_{xx}(2) \\ \vdots \\ r_{xx}(P) \end{bmatrix}. \tag{9.18}$$

where $r_{xx}(p)=E\{x(n)x(n-p)\}$.

For the stationary process of $x(n)$, we use $r_{xx}(p)=r_{xx}(-p)$. Eq. (9.18) becomes

$$\begin{bmatrix} r_{xx}(0) & r_{xx}(1) & \cdots & r_{xx}(P-1) \\ r_{xx}(1) & r_{xx}(0) & \cdots & r_{xx}(P-2) \\ \vdots & \vdots & \ddots & \vdots \\ r_{xx}(P-1) & r_{xx}(P-2) & \cdots & r_{xx}(0) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_P \end{bmatrix} = -\begin{bmatrix} r_{xx}(1) \\ r_{xx}(2) \\ \vdots \\ r_{xx}(P) \end{bmatrix}. \tag{9.19}$$

We refer the following matrix:

$$R = \begin{bmatrix} r_{xx}(0) & r_{xx}(1) & \cdots & r_{xx}(P-1) \\ r_{xx}(1) & r_{xx}(0) & \cdots & r_{xx}(P-2) \\ \vdots & \vdots & \ddots & \vdots \\ r_{xx}(P-1) & r_{xx}(P-2) & \cdots & r_{xx}(0) \end{bmatrix}, \tag{9.20}$$

as the autocorrelation matrix. The solution for the prediction coefficients is given by

$$\begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_P \end{bmatrix} = -R^{-1} \begin{bmatrix} r_{xx}(1) \\ r_{xx}(2) \\ \vdots \\ r_{xx}(P) \end{bmatrix}. \tag{9.21}$$

We can also extend the one-step forward linear prediction to a Δ-step forward linear prediction by considering the following predictor output:

$$\hat{x}(n) = -\sum_{k=1}^{P} a_k x(n - \Delta - k + 1), \tag{9.22}$$

where $\Delta \geq 1$. Following the similar derivation, we achieve

$$\begin{bmatrix} r_{xx}(0) & r_{xx}(-1) & \cdots & r_{xx}(-P+1) \\ r_{xx}(1) & r_{xx}(0) & \cdots & r_{xx}(-P+2) \\ \vdots & \vdots & \ddots & \vdots \\ r_{xx}(P-1) & r_{xx}(P-2) & \cdots & r_{xx}(0) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_P \end{bmatrix} = -\begin{bmatrix} r_{xx}(\Delta) \\ r_{xx}(\Delta+1) \\ \vdots \\ r_{xx}(\Delta+P-1) \end{bmatrix} \tag{9.23}$$

$$\begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_P \end{bmatrix} = -R^{-1} \begin{bmatrix} r_{xx}(\Delta) \\ r_{xx}(\Delta+1) \\ \vdots \\ r_{xx}(\Delta+P-1) \end{bmatrix}. \tag{9.24}$$

EXAMPLE 9.2

A second-order forward linear predictor is used to predict the sinusoid $x(n) = \sqrt{2}\sin(\pi n/6)$.

(a) For the one-step forward linear predictor, find the predictor coefficients and show that the sinusoid can be fully predicted by showing $E\{e_f^2(n)\} = 0$

(b) Compute $x(n)$ and $\hat{x}(n)$ for 10 samples for verification.

Solution:

(a) The second-order predictor is given by

$$\hat{x}(n) = -a_1 x(n-1) - a_2 x(n-2).$$

Since

$$r_{xx}(0) = E\left\{\left[\sqrt{2}\sin(\pi n/6)\right]^2\right\} = 1$$
$$r_{xx}(1) = E\left\{\left[\sqrt{2}\sin(n\pi/6)\right]\left[\sqrt{2}\sin((n-1)\pi/6)\right]\right\} = \cos(\pi/6) = \sqrt{3}/2$$
$$r_{xx}(2) = E\left\{\sqrt{2}\sin(n\pi/6)\left[\sqrt{2}\sin((n-2)\pi/6)\right]\right\} = \cos(2\pi/6) = 1/2.$$

Therefore,

Continued

EXAMPLE 9.2—CONT'D

$$R = \begin{bmatrix} 1 & \sqrt{3}/2 \\ \sqrt{3}/2 & 1 \end{bmatrix}$$

$$\begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = -\begin{bmatrix} 1 & \sqrt{3}/2 \\ \sqrt{3}/2 & 1 \end{bmatrix}^{-1} \begin{bmatrix} \sqrt{3}/2 \\ 1/2 \end{bmatrix}$$

$$= -\begin{bmatrix} 4 & -2\sqrt{3} \\ -2\sqrt{3} & 4 \end{bmatrix} \begin{bmatrix} \sqrt{3}/2 \\ 1/2 \end{bmatrix} = \begin{bmatrix} -\sqrt{3} \\ 1 \end{bmatrix}.$$

Thus, the predictor is found to be

$$\hat{x}(n) = \sqrt{3}x(n-1) - x(n-2).$$

To show the performance of the predictor, we can apply the Wiener filter results by considering

$$\sigma^2 = r_{xx}(0) = 1,\ R = \begin{bmatrix} 1 & \sqrt{3}/2 \\ \sqrt{3}/2 & 1 \end{bmatrix},\ \text{and } P = \begin{bmatrix} \sqrt{3}/2 \\ 1/2 \end{bmatrix}$$

$$W^* = -\begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} \sqrt{3} \\ -1 \end{bmatrix}.$$

Finally, the MSE of prediction results in

$$J_{\min} = \sigma^2 - W^{*T}P = 1 - \begin{bmatrix} \sqrt{3} & -1 \end{bmatrix} \begin{bmatrix} \sqrt{3}/2 \\ 1/2 \end{bmatrix} = 0.$$

(b) Using MATLAB, we have

```
>>x=sin(pi*(0:9)/6)
x =
0 0.5000 0.8660 1.0000 0.8660 0.5000 0.0000 -0.5000 -0.8.660 -1.0000
xhat=filter([0 sqrt(3) −1],1,x)
xhat = 0 0 0.8660 1.0000 0.8660 0.5000 -0.0000 -0.5000 -0.8660 -1.0000
```

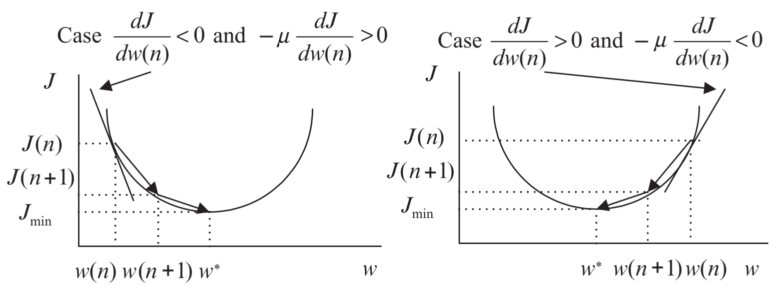
It is observed that a sinusoid can fully be predicted by a second-order one-step forward predictor after P samples ($P=2$).

In general, for predicting a sinusoid $x(n) = A\sin(n\Omega)$, a forward predictor can be used as

$$x(n) = -a_1 x(n-n_1) - a_2 x(n-n_2) \text{ for } 0 < n_1 < n_2. \tag{9.25}$$

We can show that the predictor coefficients are

$$\begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = -\begin{bmatrix} \dfrac{\sin(n_2\Omega)}{\sin[(n_2-n_1)\Omega]} \\ -\dfrac{\sin(n_1\Omega)}{\sin[(n_2-n_1)\Omega]} \end{bmatrix} \text{ for } (n_2-n_1)\Omega \neq k\pi \text{ for } k = 0, \pm 1, \pm 2, \ldots \tag{9.26}$$



Case $\frac{dJ}{dw(n)} < 0$ and $-\mu\frac{dJ}{dw(n)} > 0$
J
$J(n)$
$J(n+1)$
$J_{\min}$
$w(n)$ $w(n+1)$ w^*
w
Case $\frac{dJ}{dw(n)} > 0$ and $-\mu\frac{dJ}{dw(n)} < 0$
J
$J(n)$
$J(n+1)$
$J_{\min}$
w^* $w(n+1)$ $w(n)$
w

FIG. 9.7

Illustration of the steepest descent algorithm.

9.2.2 STEEPEST DESCENT ALGORITHM

As we pointed out, solving the Wiener solution Eq. (9.12) requires a lot of computations, including matrix inversion for a general multi-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 using a single weight as illustrated in Eq. (9.27):

$$w(n+1) = w(n) - \mu\frac{dJ}{dw(n)}, \tag{9.27}$$

where μ = constant controlling the speed of convergence.

The illustration of the steepest decent algorithm for solving the optimal coefficient(s) is described in Fig. 9.7.

As shown in the first plot of Fig. 9.7, if $\frac{dJ}{dw(n)} < 0$, notice that $-\mu\frac{dJ}{dw(n)} > 0$. The new coefficient $w(n+1)$ will be increased to approach the optimal value w^* by Eq. (9.27). On the other hand, if $\frac{dJ}{dw(n)} > 0$, as shown in the second plot of Fig. 9.7, we see that $-\mu\frac{dJ}{dw(n)} < 0$. The new coefficient $w(n+1)$ will be decreased to approach the optimal value w^*. When $\frac{dJ}{dw(n)} = 0$, the best coefficient $w(n+1)$ is reached.

EXAMPLE 9.3

Given a quadratic MSE function for the Wiener filter:

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

Use the steepest decent method with an initial guess as $w(0) = 0$ 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

$$\frac{dJ}{dw(n)} = -20 + 10 \times 2w(n).$$

Continued

EXAMPLE 9.3—CONT'D

When $n=0$, we calculate

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

Applying the steepest decent algorithm, it follows that

$$w(1)=w(0)-\mu\frac{dJ}{dw(0)}=0-(-0.8)=0.8.$$

Similarly, for $n=1$, we get

$$\mu\frac{dJ}{dw(1)}=0.04\times(-20+10\times 2w(1))\Big|_{w(1)=0.8}=-0.16$$
$$w(2)=w(1)-\mu\frac{dJ}{dw(1)}=0.8-(-0.16)=0.96,$$

and for $n=2$, it follows that

$$\mu\frac{dJ}{dw(2)}=0.04\times(-20+10\times 2w(2))\Big|_{w(2)=0.96}=-0.032$$
$$w(3)=w(2)-\mu\frac{dJ}{dw(2)}=0.96-(-0.032)=0.992.$$

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 9.1.

For general case, that is, $y(n)=W(n)^T X(n)$, we have

$$\begin{bmatrix} w_0(n+1)\\ w_1(n+1)\\ \vdots\\ w_{N-1}(n+1)\end{bmatrix}=\begin{bmatrix} w_0(n)\\ w_1(n)\\ \vdots\\ w_{N-1}(n)\end{bmatrix}-\mu\begin{bmatrix} \partial J/\partial w_0(n)\\ \partial J/\partial w_1(n)\\ \vdots\\ \partial J/\partial w_{N-1}(n)\end{bmatrix}. \tag{9.28}$$

In a vector form, we yield

$$W(n+1)=W(n)-\mu\nabla J_{W(n)}, \tag{9.29}$$

where $\nabla \mathrm{J}_{W(n)}$ is the gradient vector.

EXAMPLE 9.4

Given a quadratic MSE function for the Wiener filter:

$$J = 4 + w_0^2 + w_0 w_1 + w_1^2 - w_0 + w_1,$$

(a) Set up the steepest decent algorithm.

(b) For initial weights as $w_0(0) = 0$ and $w_1(0) = -3$, and $\mu = 0.2$, use MATLAB to find the optimal solution by iteration 100 times and plot the trajectory of $w_0(n)$ and $w_1(n)$ on the MSE contour surface.

Solution:

(a) Taking partial derivatives of the MSE function, we obtain the gradients as

$$\frac{\partial J}{\partial w_0(n)} = 2w_0(n) + w_1(n) - 1$$

and

$$\frac{\partial J}{\partial w_1(n)} = 2w_1(n) + w_0(n) + 1.$$

Then we obtain the coefficients updating equations from Eq. (9.28) as follows:

$$\begin{aligned} w_0(n+1) &= w_0(n) - \mu[2w_0(n) + w_1(n) - 1] \\ w_1(n+1) &= w_1(n) - \mu[2w_1(n) + w_0(n) + 1] \end{aligned}$$

(b) The trajectory of $w_0(n)$ and $w_1(n)$ are plotted in Fig. 9.8.

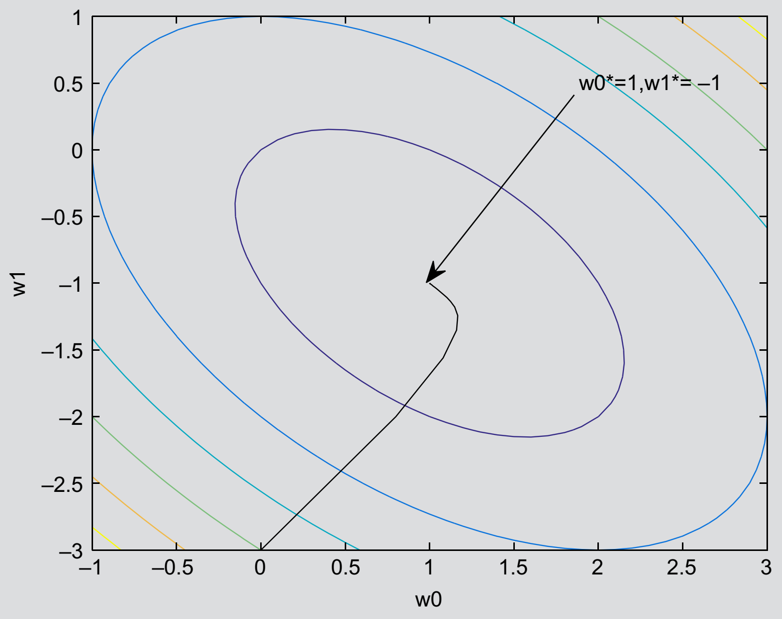


FIG. 9.8

Trajectory of coefficients in Example 9.4.

9.2.3 LEAST MEAN SQUARE ALGORITHM

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 approximate of $\frac{dJ}{dw(n)}$, that is,

$$J = e^2(n) = (d(n) - w(n)x(n))^2 \tag{9.30}$$

$$\frac{dJ}{dw(n)} = 2(d(n) - w(n)x(n))\frac{d(d(n) - w(n)x(n))}{dw(n)} = -2e(n)x(n). \tag{9.31}$$

Substituting $\frac{dJ}{dw(n)}$ into the steepest descent algorithm in Eq. (9.27), we achieve the LMS algorithm for updating a single-weight case as

$$w(n+1) = w(n) + 2\mu e(n)x(n), \tag{9.32}$$

where μ is the convergence parameter controlling the speed of the convergence and must satisfy $0 < \mu < 1/\lambda_{\max}$. $\lambda_{\max}$ is the maximum eigenvalue of the autocorrelation matrix $R = E\{X^T(n)X(n)\}$. In our case, $R = E\{x^2(n)\}$ and $\lambda_{\max} = 1/R$. 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:

$$y(n) = w_0(n)x(n) + w_1(n)x(n-1) + \cdots + w_{N-1}(n)x(n-N+1) \tag{9.33}$$

$$\text{for } k = 0, \cdots, N-1$$

$$w_k(n+1) = w_k(n) + 2\mu e(n)x(n-k). \tag{9.34a}$$

$$\text{vector form}: W(n+1) = W(n) + 2\mu e(n)X(n) \tag{9.34b}$$

The convergence factor is empirically chosen to be

$$0 < \mu < \frac{1}{NP_x}, \tag{9.35}$$

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 is less than

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

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

$$\mu = \frac{1}{N \times 2^{30}} \approx \frac{9.3 \times 10^{-10}}{N}. \tag{9.36}$$

We further neglect time index for $w_k(n)$ and use the notation $w_k = w_k(n)$, since only the current updated coefficients are needed for next sample adaptation. We conclude the implementation of the LMS algorithm by 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:
   $y(n) = w_0x(n) + w_1x(n-1) + \cdots + w_{N-1}x(n-N+1)$.

3. Compute the output error:
   $e(n)=d(n)-y(n)$.
4. Update each filter coefficient using the LMS algorithm:
   for $k=0, \cdots, N-1$
   $w_k=w_k+2\mu e(n)x(n-k)$.
   vector from: $W=W+2\mu e(n)X(n)$.

9.2.4 RECURSIVE LEAST SQUARES ALGORITHM

The RLS algorithms aim to minimize the sum of the squares of the difference between the desired signal and the filter output signal using the new samples of the incoming signal. The RLS algorithms compute filter coefficients in a recursive form at each iteration. The RLS algorithms are well known for their fast convergence even when the eigenvalue spread of the input signal correlation matrix is large. These algorithms offer excellent performance with the cost of larger computational complexity and problem of stability in comparison with the LMS algorithm. In next section, we will study the standard RLS algorithm.

Given a linear adaptive filter $y(n)=W^T(n)X(n)$ with its coefficient and input vectors defined below:

$$W(n)=[w_0(n)w_1(n)\cdots w_{N-1}(n)]^T \tag{9.37}$$

$$X(i)=[x(i)x(i-1)\cdots x(i-N+1)]^T. \tag{9.38}$$

We begin with the exponential weighted squared-error function with a forgetting factor of λ given by

$$\zeta(n)=\sum_{i=1}^{n}\lambda^{n-i}e^2(i), \tag{9.39}$$

where $0<\lambda<1$, and the error at past time index i is expressed as

$$e(i)=d(i)-W^T(n)X(i) \text{ for } 1\le i\le n. \tag{9.40}$$

Taking the derivative of the exponential weighted squared-error function to $W(n)$ leads to

$$\frac{\partial\zeta(n)}{\partial W(n)}=2\sum_{i=1}^{n}\lambda^{n-i}\left\{\left(d(i)-W^T(n)X(i)\right)X(i)\right\}. \tag{9.41}$$

Setting Eq. (9.41) to zero yield the following:

$$\sum_{i=1}^{n}\lambda^{n-i}X(i)X^T(i)W(n)=\sum_{i=1}^{n}\lambda^{n-i}d(i)X(i). \tag{9.42}$$

Now, the term on the right-hand side of Eq. (9.42) is the cross-correlation vector, which can be expanded to be

$$\begin{aligned}P(n)&=\sum_{i=1}^{n}\lambda^{n-i}d(i)X(i)\\&=d(n)X(n)+\lambda\sum_{i=1}^{n-1}\lambda^{n-1-i}d(i)X(i).\end{aligned} \tag{9.43}$$

We modify Eq. (9.43) to a recursion form, that is,

$$P(n) = \lambda P(n-1) + d(n)X(n). \tag{9.44}$$

According to Eq. (9.42), we define the autocorrelation matrix below:

$$R(n) = \sum_{i=1}^{n} \lambda^{n-i} X(i) X^T(i). \tag{9.45}$$

Expanding Eq. (9.45), we obtain the recursion relation as follows:

$$\begin{aligned} R(n) &= \sum_{i=1}^{n} \lambda^{n-i} X(i) X^T(i) \\ &= \lambda \sum_{i=1}^{n-1} \lambda^{n-1-i} X(i) X^T(i) + X(n) X^T(n) \\ &= \lambda R(n-1) + X(n) X^T(n). \end{aligned} \tag{9.46}$$

In order to determine the filter coefficients $W(n)$, we need to compute the inverse of the autocorrelation matrix, $R^{-1}(n)$. This can be completed by using the matrix inversion lemma (Haykin, 2014) listed below:

$$A^{-1} = B - BC\left(D + C^T BC\right)^{-1} C^T B. \tag{9.47}$$

To use the matrix inversion lemma, we set up the following:
$A = R(n)$, $B^{-1} = \lambda R(n-1)$, $C = X(n)$, $D = 1$, and $Q(n) = R^{-1}(n)$, we can conclude

$$Q(n) = \lambda^{-1} Q(n-1) - \lambda^{-1} Q(n-1) X(n) \left[1 + X^T(n) \lambda^{-1} Q(n-1) X(n)\right]^{-1} X^T(n) \lambda^{-1} Q(n-1). \tag{9.48}$$

To simplify Eq. (9.48), define the gain vector as

$$k(n) = \frac{\lambda^{-1} Q(n-1) X(n)}{1 + \lambda^{-1} X^T(n) Q(n-1) X(n)}. \tag{9.49}$$

Substituting Eq. (9.49) into Eq. (9.48), we have

$$Q(n) = \lambda^{-1} Q(n-1) - \lambda^{-1} k(n) X^T(n) Q(n-1). \tag{9.50}$$

From Eq. (9.49), we explore a new modified relation for $k(n)$, that is,

$$k(n) = \lambda^{-1} Q(n-1) X(n) - \lambda^{-1} k(n) X^T(n) Q(n-1) X(n). \tag{9.51}$$

Substituting Eq. (9.50) into Eq. (9.51) yields

$$k(n) = Q(n) X(n). \tag{9.52}$$

Based on the Wiener solution for the filter coefficient vector and using Eq. (9.44), it follows that

$$\begin{aligned} W(n) &= Q(n) P(n) \\ &= \lambda Q(n) P(n-1) + Q(n) d(n) X(n). \end{aligned} \tag{9.53}$$

Applying Eqs. (9.50) and (9.52) to Eq. (9.53), we obtain

$$\begin{aligned} W(n) &= \lambda Q(n)P(n-1) + k(n)d(n) \\ &= \left(Q(n-1) - k(n)X^T(n)Q(n-1)\right)P(n-1) + k(n)d(n). \end{aligned} \tag{9.54}$$

Distributing $P(n-1)$ in Eq. (9.54) and noting that $W(n-1) = Q(n-1)P(n-1)$, it follows that

$$W(n) = W(n-1) - k(n)X^T(n)W(n-1) + d(n)k(n). \tag{9.55}$$

Factoring out the gain vector *k(n)* in Eq. (9.55), it leads to

$$W(n) = W(n-1) + k(n)\left(d(n) - X^T(n)W(n-1)\right). \tag{9.56}$$

Define the innovation $\alpha(n)$ as

$$\alpha(n) = d(n) - X^T(n)W(n-1). \tag{9.57}$$

Substituting Eq. (9.57) into Eq. (9.56), we yield the following weight update equation:

$$W(n) = W(n-1) + k(n)\alpha(n). \tag{9.58}$$

Finally, we can summarize the RLS algorithm:

Initialize $Q(-1) = \delta I$, $I =$ the identity matrix and $\delta =$ the inverse of the input signal power

$$\begin{aligned} \alpha(n) &= d(n) - X^T(n)W(n-1) \\ k(n) &= \frac{\lambda^{-1}Q(n-1)X(n)}{1 + \lambda^{-1}X^T(n)Q(n-1)X(n)} \\ W(n) &= W(n-1) + k(n)\alpha(n) \\ Q(n) &= \lambda^{-1}Q(n-1) - \lambda^{-1}k(n)X^T(n)Q(n-1) \\ y(n) &= W^T(n)X(n) \\ e(n) &= d(n) - y(n). \end{aligned}$$

Similar to the LMS algorithm, by omitting time index for $w_k(n)$, that is, $w_k = w_k(n)$, we conclude the implementation of the RLS algorithm by the following steps:

1. Initialize $W = [w_0 \ \ w_1 \ \ \cdots \ \ w_{N-1}]^T$ to arbitrary values
   Initialize $Q = \delta I$, $I =$ the identity matrix and $\delta =$the inverse of the input signal power

2. Compute the innovation and gain vector:

$$\begin{aligned} \alpha &= d(n) - X^T(n)W \\ k &= \frac{QX(n)}{\lambda + X^T(n)QX(n)} \end{aligned}$$

3. Update the filter coefficients:

$$W = W + k\alpha$$

4. Update Q matrix, produce the filter output, and measure the error:

$$\begin{aligned} Q &= \left[Q - kX^T(n)Q\right]/\lambda \\ y(n) &= W^TX(n) \\ e(n) &= d(n) - y(n) \end{aligned}$$

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

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

We now examine several applications of the adaptive filters, 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 adaptive FIR filter.

9.3.1 NOISE CANCELLATION

The concept of noise cancellation was introduced in the previous section. Fig. 9.9A shows the main idea.

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 adaptive 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.

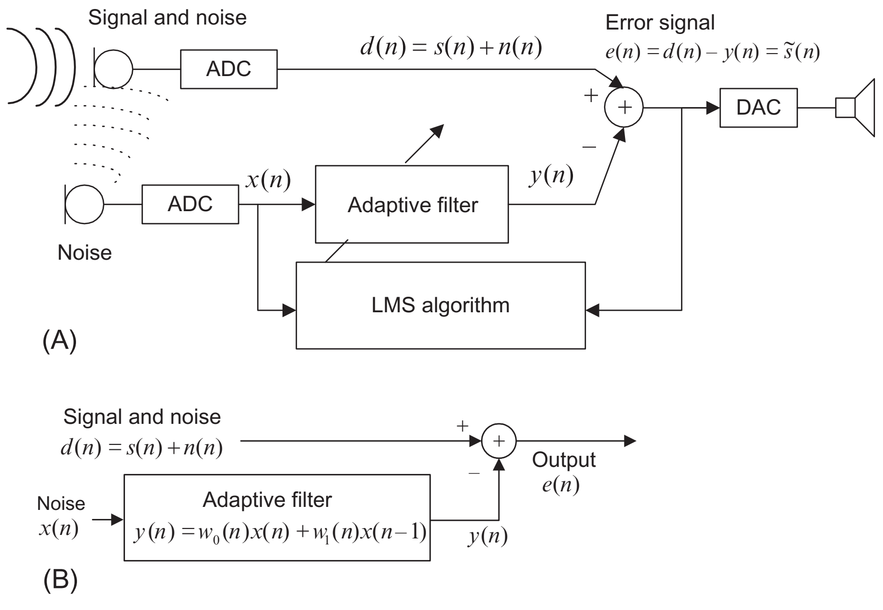


FIG. 9.9

(A) Simplest noise canceller using a one-tap adaptive filter. (B) Noise cancellation in Example 9.5.

We first study the noise cancellation problem using a simple two-tap adaptive filter via Example 9.5 and assumed data. The purpose of doing so is to become familiar with the setup and operations of the adaptive filter algorithms. The simulation for real adaptive noise cancellation follows.

EXAMPLE 9.5

Given the DSP system for the noise cancellation application using an adaptive filter with two coefficients shown in Fig. 9.9B,

(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:

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

and initial weights: $w_0=w_1=0$,
convergence factor is set to be $\mu=0.1$.

(c) Set up the RLS algorithm.

(d) Repeat (a) using $\lambda=0.96$.

Solution:

(a) The adaptive LMS algorithm is set up as
Initialization: $w_0=0$, $w_1=0$
Digital filtering:

$$y(n)=w_0x(n)+w_1x(n-1)$$

Computing the output error = output:

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

Updating each weight for the next coming sample:

$$w_k=w_k+2\mu e(n)x(n-k),\ \text{for}\, k=0,1$$

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:

$$y(0)=w_0x(0)+w_1x(-1)=0\times 1+0\times 0=0$$

Computing the output:

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

Updating coefficients:

$$w_0=w_0+2\times 0.1\times e(0)x(0)=0+2\times 0.1\times 2\times 1=0.4$$
$$w_1=w_1+2\times 0.1\times e(0)x(-1)=0+2\times 0.1\times 2\times 0=0.0$$

Continued

EXAMPLE 9.5—CONT'D

For $n=1$

Digital filtering:

$$y(1)=w_0x(1)+w_1x(0)=0.4\times1+0\times1=0.4$$

Computing the output:

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

Updating coefficients:

$$\begin{aligned}w_0&=w_0+2\times0.1\times e(1)x(1)=0.4+2\times0.1\times0.6\times1=0.52\\ w_1&=w_1+2\times0.1\times e(1)x(0)=0+2\times0.1\times0.6\times1=0.12\end{aligned}$$

For $n=2$

Digital filtering:

$$y(2)=w_0x(2)+w_1x(1)=0.52\times(-1)+0.12\times1=-0.4$$

Computing the output:

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

Updating coefficients:

$$\begin{aligned}w_0&=w_0+2\times0.1\times e(2)x(2)=0.52+2\times0.1\times(-1.6)\times(-1)=0.84\\ w_1&=w_1+2\times0.1\times e(2)x(1)=0.12+2\times0.1\times(-1.6)\times1=-0.2\end{aligned}$$

Hence, the adaptive filter outputs for the first three samples are listed as $e(0)=2$, $e(1)=0.6$, $e(2)=-1.6$.

(c) The RLS algorithm is set up as

$$\delta\approx1/[(x^2(0)+x^2(1)+x^2(2))/3]=1;\ \lambda=0.96$$

$$Q=\begin{bmatrix}1&0\\0&1\end{bmatrix},\ \begin{bmatrix}w_0\\w_1\end{bmatrix}=\begin{bmatrix}0\\0\end{bmatrix}$$

$$\alpha=d(n)-[x(n)x(n-1)]\begin{bmatrix}w_0\\w_1\end{bmatrix}$$

$$k=\frac{Q\begin{bmatrix}x(n)\\x(n-1)\end{bmatrix}}{\lambda+[x(n)\ \ x(n-1)]Q\begin{bmatrix}x(n)\\x(n-1)\end{bmatrix}}$$

$$\begin{bmatrix}w_0\\w_1\end{bmatrix}=\begin{bmatrix}w_0\\w_1\end{bmatrix}+k\alpha$$

$$Q=(Q-k[x(n)\ \ x(n-1)]Q)/\lambda$$

$$y(n)=[w_0\ \ w_1]\begin{bmatrix}x(n)\\x(n-1)\end{bmatrix}$$

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

(d) For $n=0$

$$\alpha = d(0) - [x(0)\ \ x(-1)]\begin{bmatrix} w_0 \\ w_1 \end{bmatrix} = 2 - [1\ \ 0]\begin{bmatrix} 0 \\ 0 \end{bmatrix} = 2$$

$$k = \frac{Q\begin{bmatrix} x(0) \\ x(-1) \end{bmatrix}}{\lambda + [x(0)\ \ x(-1)]Q\begin{bmatrix} x(0) \\ x(-1) \end{bmatrix}} = \frac{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix}}{0.96 + [1\ \ 0]\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix}} = \begin{bmatrix} 0.5102 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} w_0 \\ w_1 \end{bmatrix} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} + k\alpha = \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0.5102 \\ 0 \end{bmatrix} \times 2 = \begin{bmatrix} 1.0204 \\ 0 \end{bmatrix}$$

$$Q = \{Q - k[x(0)\ \ x(-1)]Q\}/\lambda$$

$$= \left\{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} 0.5102 \\ 0 \end{bmatrix}[1\ \ 0]\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right\}\Big/0.96 = \begin{bmatrix} 0.5102 & 0 \\ 0 & 1.0417 \end{bmatrix}$$

$$y(0) = [w_0\ \ w_1]\begin{bmatrix} x(0) \\ x(-1) \end{bmatrix} = [1.0204\ \ 0]\begin{bmatrix} 1 \\ 0 \end{bmatrix} = 1.0204$$

$$e(0) = d(0) - y(0) = 2 - 1.0204 = 0.9796.$$

For $n=1$

$$\alpha = d(1) - [x(1)x(0)]\begin{bmatrix} w_0 \\ w_1 \end{bmatrix} = 1 - [1\ \ 1]\begin{bmatrix} 1.0204 \\ 0 \end{bmatrix} = -0.0204$$

$$k = \frac{Q\begin{bmatrix} x(1) \\ x(0) \end{bmatrix}}{\lambda + [x(1)\ \ x(0)]Q\begin{bmatrix} x(1) \\ x(0) \end{bmatrix}} = \frac{\begin{bmatrix} 0.5102 & 0 \\ 0 & 1.0417 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \end{bmatrix}}{0.96 + [1\ \ 1]\begin{bmatrix} 0.5102 & 0 \\ 0 & 1.0417 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \end{bmatrix}} = \begin{bmatrix} 0.2031 \\ 0.4147 \end{bmatrix}$$

$$\begin{bmatrix} w_0 \\ w_1 \end{bmatrix} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} + k\alpha = \begin{bmatrix} 1.0204 \\ 0 \end{bmatrix} + \begin{bmatrix} 0.2031 \\ 0.4147 \end{bmatrix} \times (-0.0204) = \begin{bmatrix} 1.0163 \\ -0.0085 \end{bmatrix}$$

$$Q = \{Q - k[x(1)\ \ x(0)]Q\}/\lambda$$

$$= \left\{\begin{bmatrix} 0.5102 & 0 \\ 0 & 1.0417 \end{bmatrix} - \begin{bmatrix} 0.2031 \\ 0.4147 \end{bmatrix}[1\ \ 1]\begin{bmatrix} 0.5102 & 0 \\ 0 & 1.0417 \end{bmatrix}\right\}\Big/0.96 = \begin{bmatrix} 0.4235 & -0.2204 \\ -0.2204 & 0.6351 \end{bmatrix}$$

$$y(1) = [w_0\ \ w_1]\begin{bmatrix} x(1) \\ x(0) \end{bmatrix} = [1.0163\ \ -0.0085]\begin{bmatrix} 1 \\ 1 \end{bmatrix} = 1.0078$$

$$e(1) = d(1) - y(1) = 1 - 1.0078 = -0.0078.$$

Continued

EXAMPLE 9.5—CONT'D

For $n=2$

$$\alpha = d(2) - [x(2)x(1)]\begin{bmatrix} w_0 \\ w_1 \end{bmatrix} = 1 - [-1 \quad 1]\begin{bmatrix} 1.0163 \\ -0.0085 \end{bmatrix} = -0.9753$$

$$k = \frac{Q\begin{bmatrix} x(2) \\ x(1) \end{bmatrix}}{\lambda + [x(2) \quad x(1)]Q\begin{bmatrix} x(2) \\ x(1) \end{bmatrix}} = \frac{\begin{bmatrix} 0.4235 & -0.2204 \\ -0.2204 & 0.6351 \end{bmatrix}\begin{bmatrix} -1 \\ 1 \end{bmatrix}}{0.96 + [-1 \quad 1]\begin{bmatrix} 0.4235 & -0.2204 \\ -0.2204 & 0.6351 \end{bmatrix}\begin{bmatrix} -1 \\ 1 \end{bmatrix}} = \begin{bmatrix} -0.2618 \\ 0.3478 \end{bmatrix}$$

$$\begin{bmatrix} w_0 \\ w_1 \end{bmatrix} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} + k\alpha = \begin{bmatrix} 1.0163 \\ -0.0085 \end{bmatrix} + \begin{bmatrix} -0.2618 \\ 0.3478 \end{bmatrix} \times (-0.9753) = \begin{bmatrix} 1.2716 \\ -0.3477 \end{bmatrix}$$

$$Q = \{Q - k[x(2) \quad x(1)]Q\}/\lambda$$

$$= \left\{ \begin{bmatrix} 0.4235 & -0.2204 \\ -0.2204 & 0.6351 \end{bmatrix} - \begin{bmatrix} -0.2618 \\ 0.3478 \end{bmatrix} [-1 \quad 1] \begin{bmatrix} 0.4235 & -0.2204 \\ -0.2204 & 0.6351 \end{bmatrix} \right\} \Big/ 0.96$$

$$= \begin{bmatrix} 1.1556 & -0.6013 \\ -0.6013 & 1.7382 \end{bmatrix}$$

$$y(2) = [w_0 \quad w_1]\begin{bmatrix} x(2) \\ x(1) \end{bmatrix} = [1.2716 \quad -0.3477]\begin{bmatrix} -1 \\ 1 \end{bmatrix} = -1.6193$$

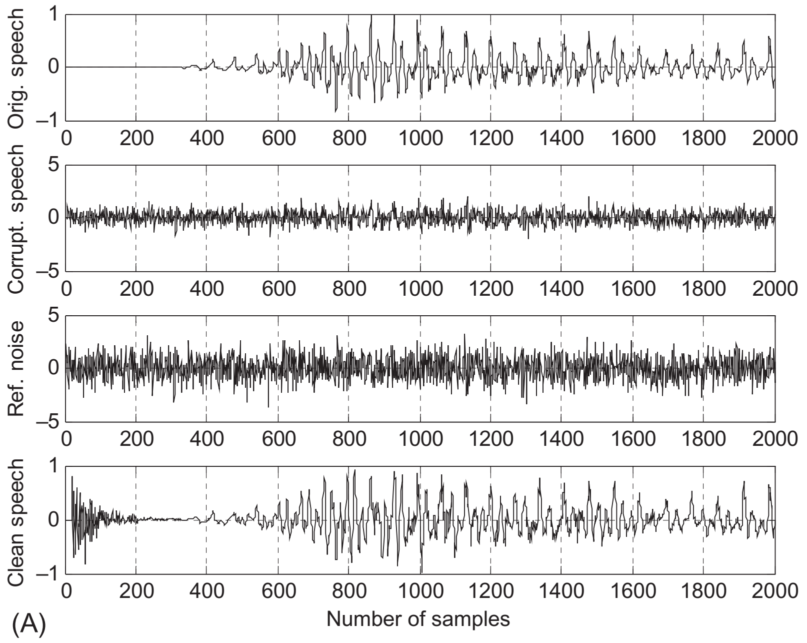
$$e(2) = d(2) - y(2) = -2 - (-1.6139) = -0.3807.$$

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 = 8000 Hz

- Original speech data: wen.dat
- Speech corrupted by Gaussian noise with a power of 1 delayed by five 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 tap = 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 Fig. 9.10A and B. 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. MATLAB implementation is detailed in Program 9.1A.



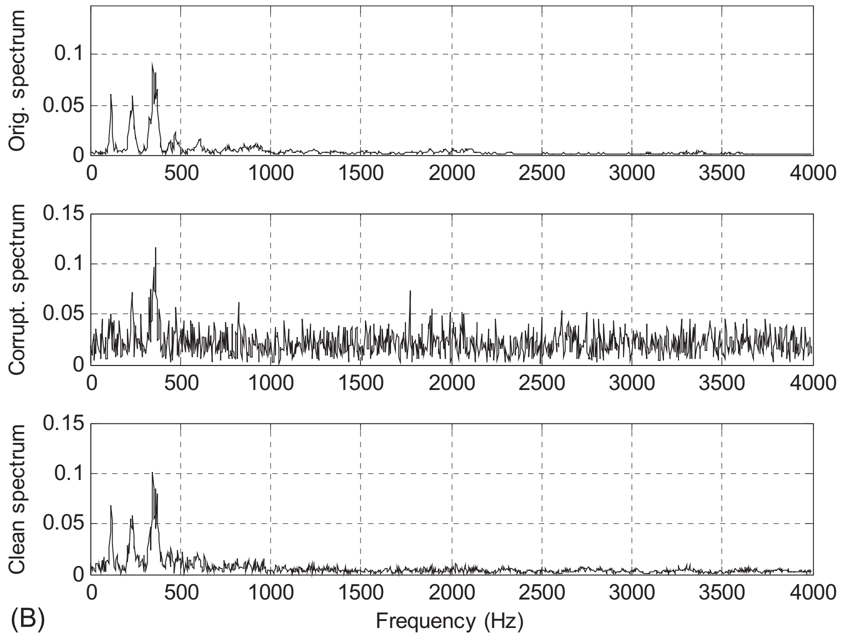


FIG. 9.10

(A) Waveforms for original speech, corrupted speech, reference noise, and clean speech. (B) Spectra for original speech, corrupted speech, and clean speech.

Continued

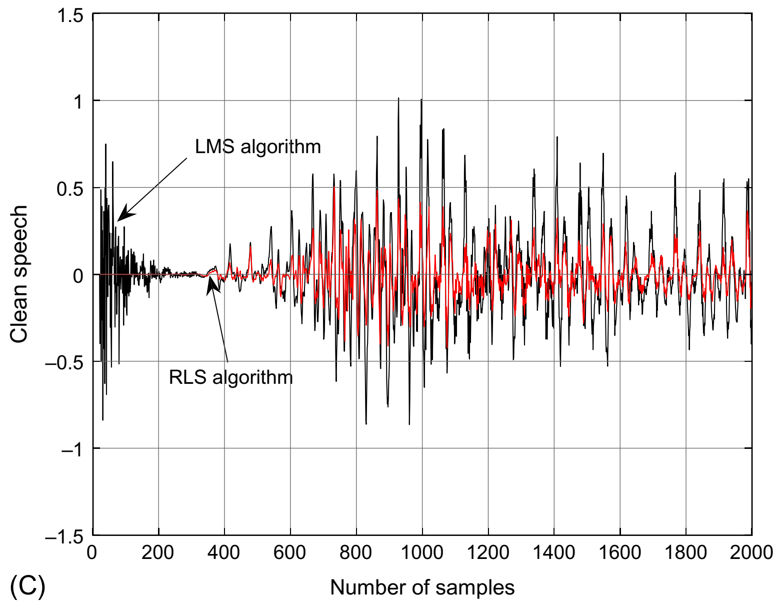


FIG. 9.10, CONT'D

(C) Comparison of the enhanced speech signals by the LMS (converges after 400 iterations) and RLS (converges immediately) algorithms.

Program 9.1A. MATLAB program for adaptive noise cancellation using the LMS algorithm.

```
close all; clear all
load wen.dat                          % Given by the instructor
fs=8000;                              % Sampling rate
t=0:1:length(wen)-1;                  % Create index array
t=t/fs;                               % Convert indices to time instant
x=randn(1,length(wen));               % Generate random noise
n=filter([ 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 k=1:1:21
    sum=sum+w(k)*x(m-k);
    end
    y(m)=sum;
    e(m)=d(m)-y(m);
    for k=1:1:21
    w(k)=w(k)+2*mu*e(m)*x(m-k);
    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)');
```

Program 9.1B shows a program segment for the RLS algorithm with $\lambda = 0.96$. The comparison for the enhanced speech signals via both the LMS and RLS algorithms is displayed in Fig. 9.10C. It can be seen that the RLS algorithm converges much faster (converges immediately) and has better performance but the algorithm has the large computational complexity.

Program 9.1B. MATLAB program for adaptive noise cancellation using the RLS algorithm.

```
w=zeros(1,21);        % Initialize adaptive filter coefficients
Q=eye(21)*mean(sum(x.*x)); % Initialize the inverse of auto-correlation matrix
% Adaptive filtering using the RLS algorithm
lamda=0.96;
% Perform adaptive filtering using the RLS algorithm
for n=21:length(t)
  xx=x(n:-1:n-20)'; % Obtain input vector
  alpha=d(n)-w*xx;
  k=Q*xx/(lamda+xx'*Q*xx);
  w=w+k'*alpha;
  Q=(Q-k*xx'*Q)/lamda;
  y(n)=w*xx;
  e(n)=d(n)-y(n);
end
```

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 a later section.

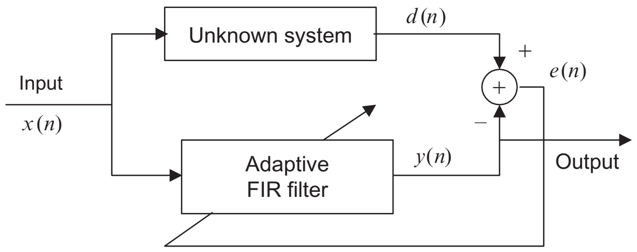


FIG. 9.11

Adaptive filter for system modeling.

9.3.2 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 Fig. 9.11.

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

EXAMPLE 9.6

Given system modeling described in Fig. 9.11, use 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 the initial $w=0$ and $\mu=0.3$.

(b) Perform adaptive filtering to obtain $y(0)$, $y(1)$, $y(2)$, and $y(3)$, given

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

(c) Set up the RLS algorithm $\lambda=0.96$ and $\delta=1$.

(d) Repeat (b) using the RLS algorithm.

Solution:

(a) Adaptive filtering equations are set up as

$$w=0 \text{ and } 2\mu=2\times 0.3=0.6$$
$$y(n)=wx(n)$$
$$e(n)=d(n)-y(n)$$
$$w=w+0.6\times e(n)x(n)$$

(b) Adaptive filtering:

$$\begin{aligned}
n=0,\ & y(0)=wx(0)=0\times 0.5=0\\
& e(0)=d(0)-y(0)=1-0=1\\
& w=w+0.6\times e(0)x(0)=0+0.6\times 1\times 0.5=0.3\\
n=1,\ & y(1)=wx(1)=0.3\times 1=0.3\\
& e(1)=d(1)-y(1)=2-0.3=1.7\\
& w=w+0.6\times e(1)x(1)=0.3+0.6\times 1.7\times 1=1.32\\
n=2,\ & y(2)=wx(2)=1.32\times(-1)=-1.32\\
& e(2)=d(2)-y(2)=-2-(-1.32)=-0.68\\
& w=w+0.6\times e(2)x(2)=1.32+0.6\times(-0.68)\times(-1)=1.728\\
n=3,\ & y(3)=wx(3)=1.728\times 1=1.728\\
& e(3)=d(3)-y(3)=2-1.728=0.272\\
& w=1.728+0.6\times e(3)x(3)=1.728+0.6\times 0.272\times 1=1.8912.
\end{aligned}$$

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

(c) The RLS algorithm is set up as

$$\begin{aligned}
&\delta=1;\lambda=0.96\\
&Q=\delta\times 1=1, w=0\\
&\alpha=d(n)-x(n)w\\
&k=\frac{Qx(n)}{\lambda+x(n)Qx(n)}\\
&w=w+k\alpha\\
&Q=[Q-kx(n)Q]/\lambda\\
&y(n)=x(n)w\\
&e(n)=d(n)-y(n)
\end{aligned}$$

(d) For $n=0$

$$\begin{aligned}
&\alpha=d(0)-x(0)w=1-0.5\times 0=1\\
&k=\frac{Qx(0)}{\lambda+x(0)Qx(0)}=1\times 0.5/(0.96+0.5\times 1\times 0.5)=0.4132\\
&w=w+k\alpha=0+0.4132\times 1=0.4132\\
&Q=[Q-kx(0)Q]/\lambda=(1-0.4132\times 0.5\times 1)/0.96=0.8265\\
&y(0)=wx(0)=0.4132\times 0.5=0.2066\\
&e(0)=d(0)-y(0)=0.7934.
\end{aligned}$$

Continued

EXAMPLE 9.6—CONT'D

For $n=1$

$$\alpha = d(1) - x(1)w = 1.5868$$
$$k = \frac{Qx(1)}{\lambda + x(1)Qx(1)} = \frac{0.8265 \times 1}{0.96 + 1 \times 0.8265 \times 1} = 0.4626$$
$$w = w + k\alpha = 0.4123 + 0.4626 \times 1.5868 = 1.1473$$
$$Q = [Q - kx(1)Q]/\lambda = [0.8265 - 0.4626 \times 1 \times 0.8265]/0.96 = 0.4626$$
$$y(1) = wx(1) = 1.1473 \times 1 = 1.1473$$
$$e(1) = d(1) - y(1) = 2 - 1.1473 = 0.8527.$$

For $n=2$

$$\alpha = d(2) - wx(2) = -2 - 1.1473 \times (-1) = -0.8527$$
$$k = \frac{Qx(2)}{\lambda + x(2)Qx(2)} = \frac{0.4626 \times (-1)}{0.96 + (-1) \times 0.4626 \times (-1)} = -0.3252$$
$$w = w + k\alpha = 1.1473 + (-0.3252) \times (-0.8527) = 2.1708$$
$$Q = [Q - kx(2)Q]/\lambda = [0.2626 - (-0.3252) \times (-1) \times 0.4626]/0.96 = 0.3252$$
$$y(2) = wx(2) = 2.1708 \times (-1) = -2.1708$$
$$e(2) = d(2) - y(2) = -2 - (-2.1708) = 0.1708.$$

For $n=3$

$$\alpha = d(3) - x(3)w = 2 - 1 \times 2.1708 = -0.1708$$
$$k = \frac{Qx(3)}{\lambda + x(3)Qx(3)} = \frac{0.3252 \times 1}{0.96 + 1 \times 0.3252 \times 1} = 0.2530$$
$$w = w + k\alpha = 2.1708 + 0.2530 \times (-0.1708) = 2.1276$$
$$Q = [Q - kx(3)Q]/\lambda = [0.3252 - 0.2530 \times 1 \times 0.3252]/0.96 = 0.2530$$
$$y(3) = wx(3) = 2.1276 \times 1 = 2.1276$$
$$e(3) = d(3) - y(3) = 2 - 2.1276 = -0.1276.$$

Next, we assume the unknown system is a fourth-order bandpass IIR filter whose 3-dB lower and upper cutoff frequencies are 1400 and 1600 Hz operating at 8000 Hz. We use an input consisting of tones of 500, 1500, and 2500 Hz. The unknown system's frequency responses are shown in Fig. 9.12.

The input waveform $x(n)$ with three tones is shown in the first plot of Fig. 9.13A. We can predict that the output of the unknown system will contain a 1500-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 be 0.01. In 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.

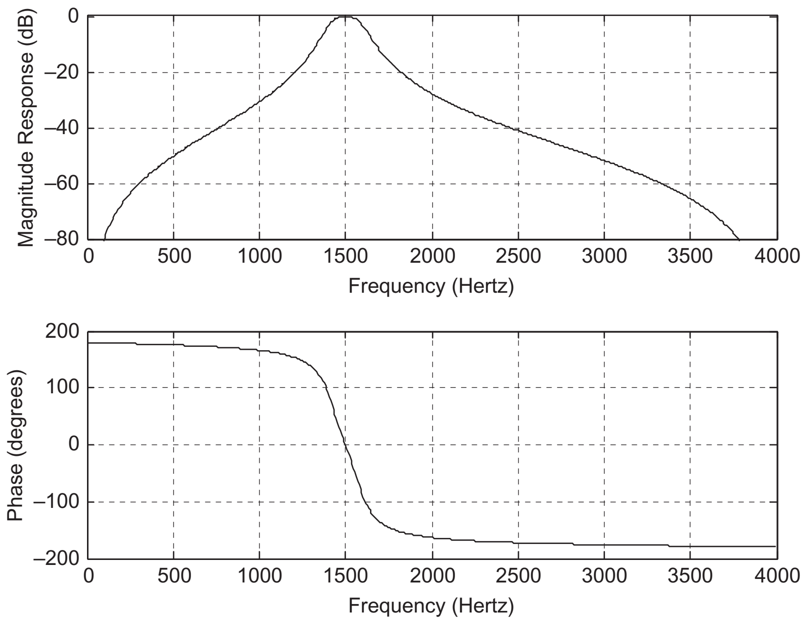


FIG. 9.12

Unknown system's frequency responses.

Fig. 9.13B depicts the frequency domain comparisons. The first plot displays the frequency components of the input signal, which clearly shows 500, 1500, and 2500 Hz. The second plot shows the unknown system's output spectrum, which contains only 1500 Hz tone, while the third plot displays the spectrum of the adaptive filter output. As we can see, in frequency domain, the adaptive filter tracks the characteristics of the unknown system. The MATLAB implementation is given in Program 9.2.

Program 9.2. MATLAB program for adaptive system identification.

```
close all; clear all
%Create the unknown system
fs=8000; T=1/fs;                    % Sampling rate and sampling period
% 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]
[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 k=1:1:21
    sum=sum+w(k)*x(m-k);
    end
    y(m)=sum;
    e(m)=d(m)-y(m);
    for k=1:1:21
    w(k)=w(k)+2*mu*e(m)*x(m-k);
 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)');
```

Fig. 9.14 displays the error comparison of the system modeling by the LMS and RLS algorithms for first 160 samples. As shown in Fig. 9.14, the LMS algorithm takes 40 samples for the absolute value of its system error to reach less than 0.2. The RLS algorithms offer much better performance.

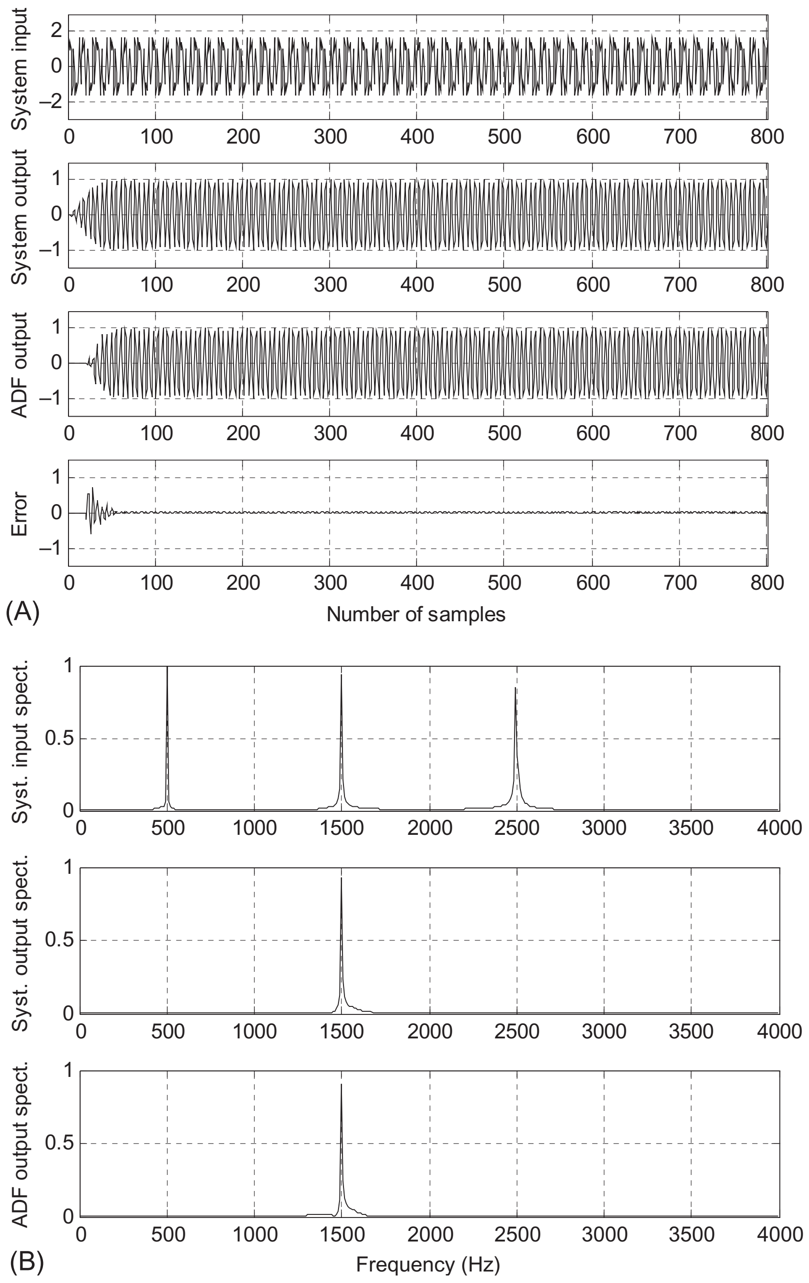


FIG. 9.13

(A) Waveforms for the unknown system's output, adaptive filter output, and error output. (B) Spectra for the input signal, unknown system output, and the adaptive filter output.

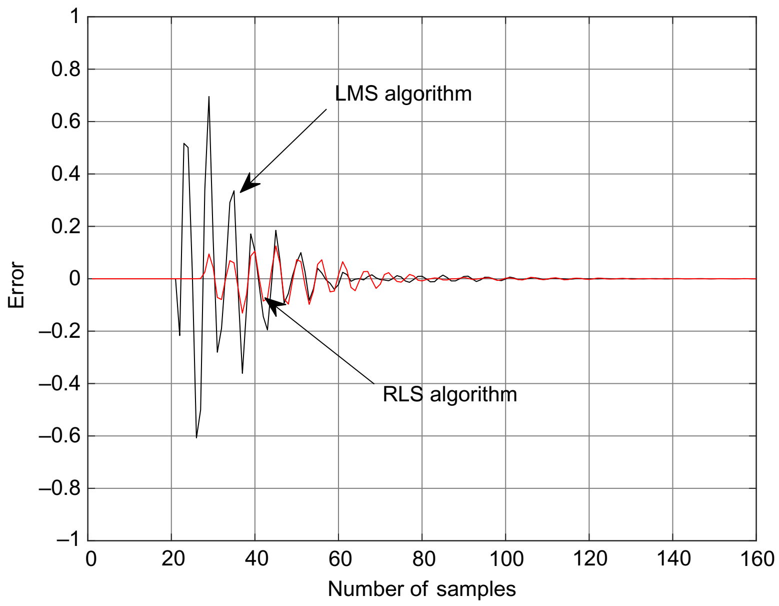


FIG. 9.14

Modeling error comparison of the system modeling by the LMS and RLS algorithms.

9.3.3 LINE ENHANCEMENT USING LINEAR PREDICTION

We study adaptive filtering via another application example: line enhancement. If a 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. Fig. 9.15 shows line enhancement using the adaptive filter where the LMS algorithm is used. As illustrated in the figure, the signal $d(n)$ is the corrupted sine wave signal by the white Gaussian noise $n(n)$. The enhanced line consists of

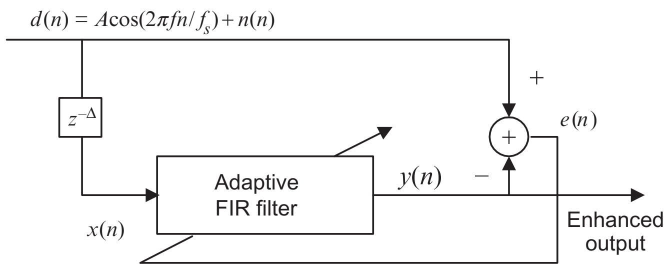


FIG. 9.15

Line enhancement using an adaptive filter.

a 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. 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 = 8000 Hz
- Corrupted signal = 500 Hz tone with the unit amplitude added with white Gaussian noise
- Adaptive filter = FIR type, 21 taps
- Convergence factor = 0.001
- Delay value Δ = 7
- LMS algorithm is applied

Fig. 9.16 shows time domain results. The first plot is the noisy signal, while the second plot clearly demonstrates the enhanced signal. Fig. 9.17 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 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 changes with time. Program 9.3 lists the MATLAB program for this simulation.

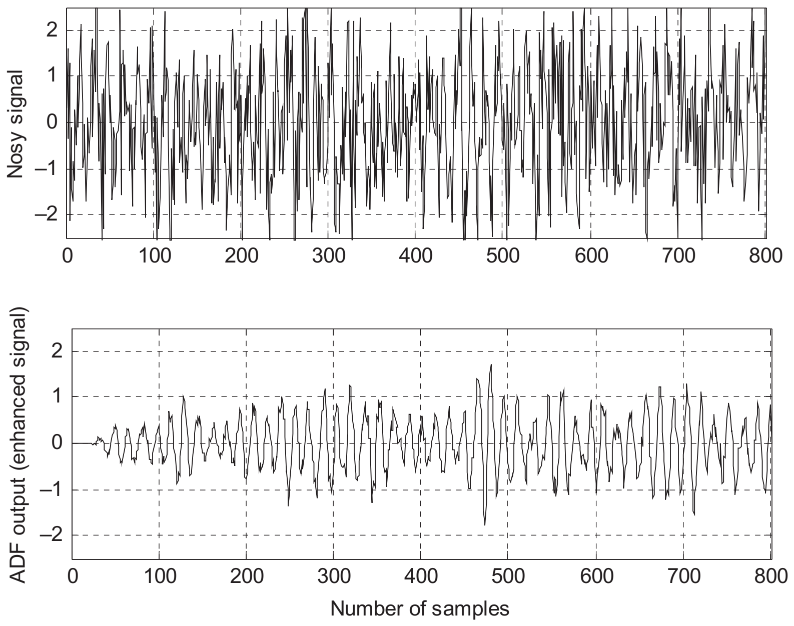


FIG. 9.16

Noisy signal and enhanced signal.

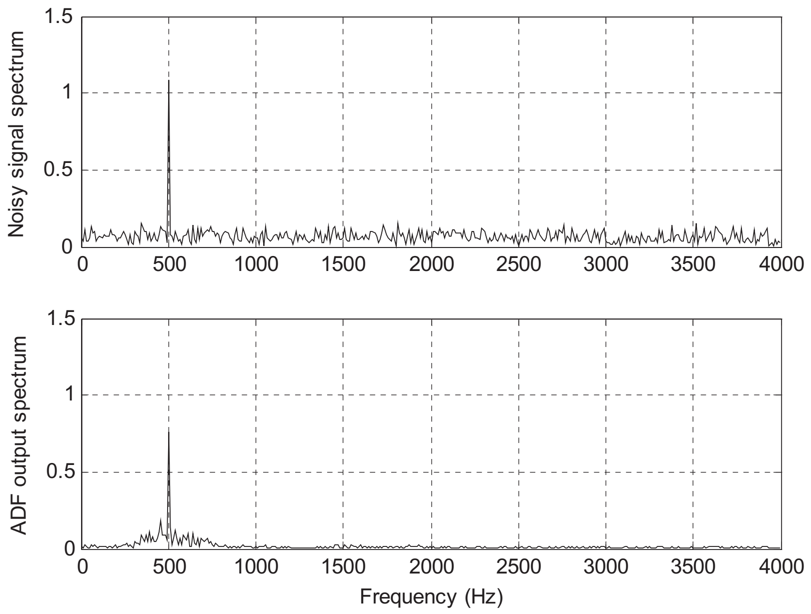


FIG. 9.17

Spectrum plots for the noisy signal and enhanced signal.

Program 9.3. 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 instants
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);
    end
    y(m)=sum;
    e(m)=d(m)-y(m);
    for k=1:1:21
    w(k)=w(k)+2*mu*e(m)*x(m-k);
    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)');
```

9.4 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 (2014), Ifeachor and Jervis (2002), Stearns and Hush (2011), and Widrow and Stearns (1985).

9.4.1 CANCELING PERIODIC INTERFERENCES USING LINEAR PREDICTION

An audio signal may be corrupted by periodic interference and no noise reference available. Such examples include the playback of speech or music with the interference of tape hum, or tunable rumble, or vehicle engine, or power line interference. We can use the modified line enhancement structure as shown in Fig. 9.18.

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

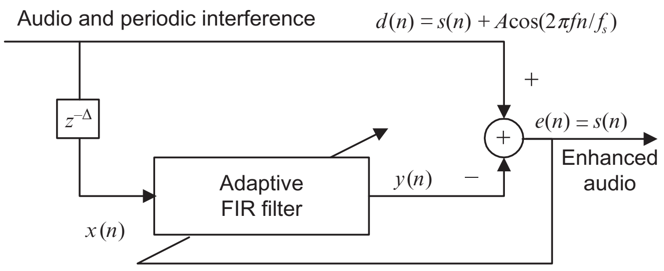


FIG. 9.18

Canceling periodic interference using the adaptive filter.

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 will predict the interference as

$$y(n) = \sum_{k=0}^{N-1} w_k(n)x(n-k) \approx A\cos(2\pi fn/f_s). \tag{9.59}$$

Therefore, the error signal contains only the desired audio signal

$$e(n) \approx s(n). \tag{9.60}$$

9.4.2 ELECTROCARDIOGRAPHY INTERFERENCE CANCELLATION

As 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 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.

Fig. 9.19 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:

$$y(n) \approx n(n). \tag{9.61}$$

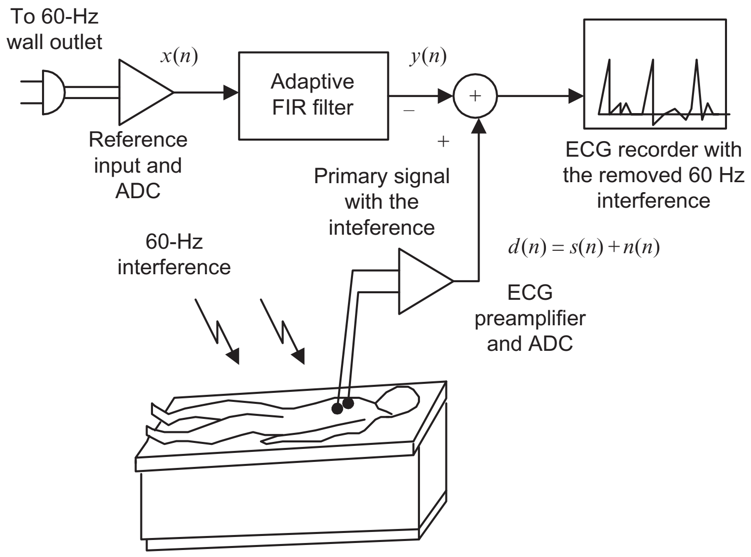


FIG. 9.19

Illustration of canceling 60-Hz interference in ECG.

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

$$y(n) = w_0(n)x(n) + w_1(n)x(n-1) + \cdots + w_{N-1}(n)x(n-N+1). \tag{9.62}$$

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:

$$e(n) = d(n) - y(n) = s(n) + n(n) - x(n) \approx s(n). \tag{9.63}$$

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

Cancelling maternal ECG in fetal monitoring is another important application. The block diagram is shown in Fig. 9.20A. Fetal ECG plays important roles in monitoring the condition of the baby before or during birth. However, the ECG acquired from the mother's abdomen is contaminated by the noise such as muscle activity and fetal motion, as well as mother's own ECG. In order to reduce the effect of the mother's ECG, four (or more) chest leads (electrodes) are used to acquire the reference inputs: $x_0(n)$, $x_1(n)$, $x_2(n)$, and $x_3(n)$, assuming these channels only contain mother's ECG [see Fig. 9.20B]. One lead (electrode) placed on the mother's abdomen is used to capture the fetal information $d(n)$,

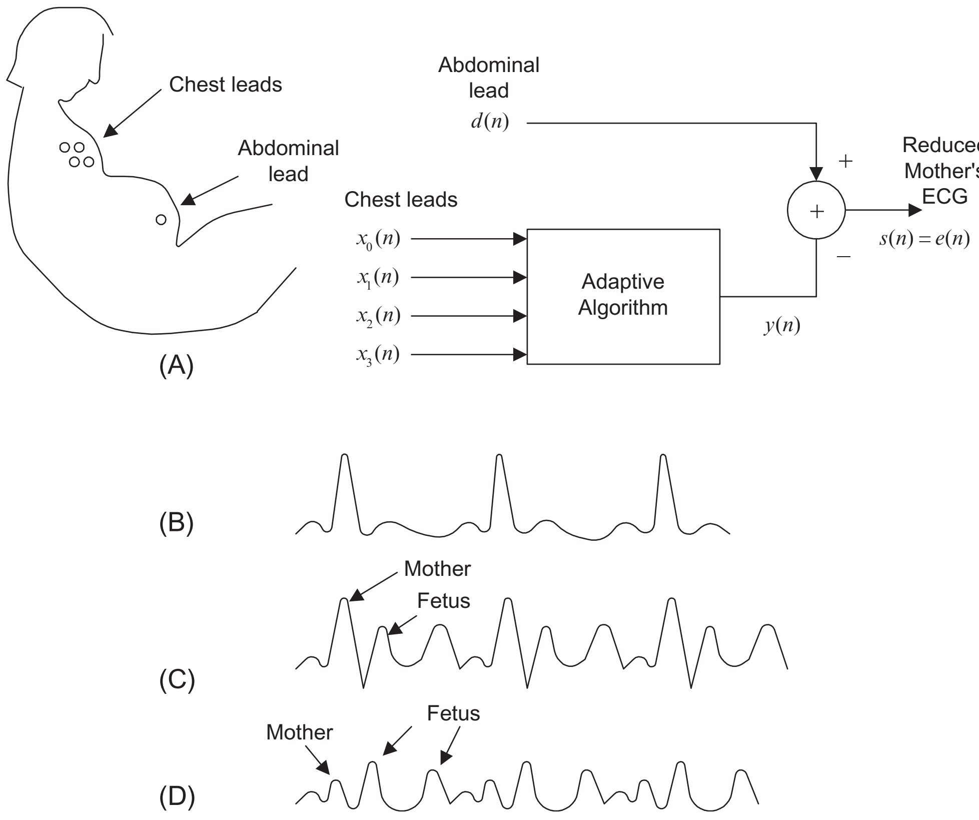


FIG. 9.20

Fetal monitoring with cancelling maternal ECG.

which may be corrupted by the mother's ECG as shown in Fig. 9.20C. 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 expected, as depicted in Fig. 9.20D. One possible LMS algorithm is listed below:

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

$$y_k(n)=w_{k,0}(n)x_k(n)+w_{k,1}(n)x_k(n-1)+\cdots+w_{k,N-1}(n)x_k(n-N+1)$$
$$y(n)=y_0(n)+y_1(n)+y_2(n)+y_3(n)$$
$$s(n)=e(n)=d(n)-y(n)$$

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

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

9.4.3 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 Fig. 9.21A, 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 1000 miles, such as geostationary satellites), the echo can be delayed by as much as 540 ms. The echo impairment can be annoying to the customer and increases with the distance.

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 Fig. 9.21B. Let us examine the adaptive filter installed at the speaker A site. The coming signal is $x_B(n)$ from the speaker B, while the outgoing signal contains the speech from the speaker A and a portion of leakage from the hybrid circuit $d_A(n)=x_A(n)+\bar{x}_B(n)$. 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,

$$y_A(n)=\sum_{k=0}^{N-1} w_k(n)x_B(n-k)\approx\bar{x}_B(n). \tag{9.64}$$

As shown in Fig. 9.21B, 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 the speaker B is removed. We can illustrate similar operations for the adaptive filter used at the speaker B site. In practice, an adaptive FIR 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.

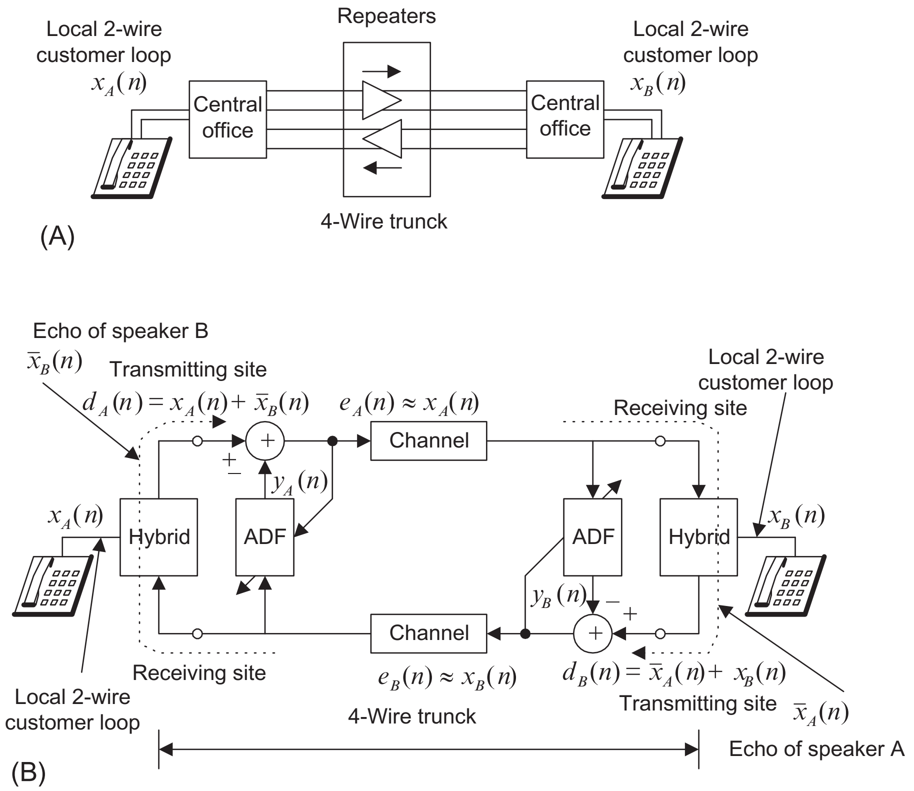


FIG. 9.21

(A) Simplified long-distance circuit. (B) Adaptive echo cancellers.

9.5 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 solution 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 size of the 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 the 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 RLS adaptive FLR filter is introduced. The RLS algorithm offers fast convergence rate but requires a large computational load.
7. The adaptive FIR filter can effectively be applied for noise cancellation, system modeling, and line enhancement.
8. Further exploration includes other applications such as cancellation of periodic interference, biomedical ECG signal enhancement, and adaptive telephone echo cancellation.

9.6 PROBLEMS

9.1 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 $J_{\min}$.

9.2 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 $J_{\min}$.

9.3 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 $J_{\min}$.

9.4 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 $J_{\min}$.

9.5 Given a quadratic MSE function for the Wiener filter:

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

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

9.6 Given a quadratic MSE function for the Wiener filter:

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

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

9.7 Given a quadratic MSE function for the Wiener filter:

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

use the steepest decent method with an initial guess as $w(0) = -4$ and the convergence factor $\mu = 0.2$ to find the optimal solution for w^* and determine $J_{\min}$ by iterating three times.

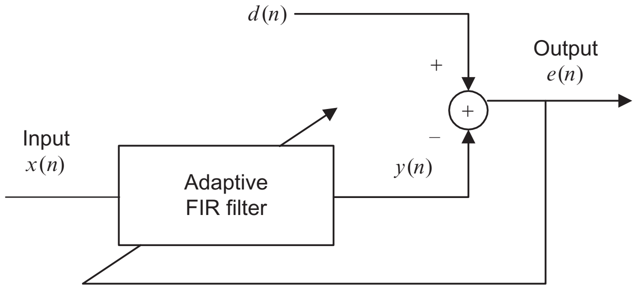


FIG. 9.22

Noise cancellation in Problem 9.9.

9.8 Given a quadratic MSE function for the Wiener filter:

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

use the steepest decent method with an initial guess as $w(0)=2$ and the convergence factor $\mu=0.02$ to find the optimal solution for w^* and determine $J_{\min}$ by iterating three times.

9.9 Given the following adaptive filter used for noise cancellation application (Fig. 9.22), 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 two taps: $y(n)=w_0x(n)+w_1x(n-1)$ with initial values $w_0=0$, $w_1=1$, and $\mu=0.1$

(a) Determine the LMS algorithm equations

$y(n)=$
$e(n)=$
$w_0=$
$w_1=$

(b) Perform adaptive filtering for each $n=0, 1, 2$.
(c) Determine equations using the RLS algorithm.
(d) Repeat (b) using the RLS algorithm with $\delta=1/2$ and $\lambda=0.96$.

9.10 Given a DSP system with a sampling rate set up to 8000 samples per second, implement adaptive filter with five taps for system modeling.
As shown in Fig. 9.23, assume that the unknown system transfer function is

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

(a) Determine the DSP equations using the LMS algorithm

$y(n)=$
$e(n)=$
$w_k=$

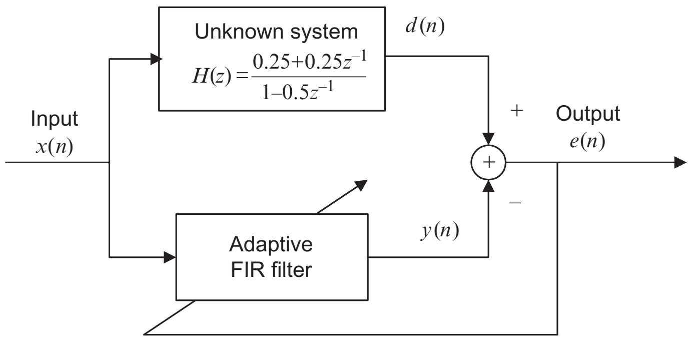


FIG. 9.23

System modeling in Problem 9.9.

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

$w_0 =$

$w_1 =$

$w_2 =$

$w_3 =$

$w_4 =$

(b) Determine equations using the RLS algorithm.

9.11 Given an adaptive filter used for noise cancellation application in Problem 9.9, 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:
$y(n)=w_0x(n)+w_1x(n-1)+w_2x(n-2)$ 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 $n=0, 1, 2$.

(c) Determine equations using the RLS algorithm.

(d) Repeat (b) using the RLS algorithm with $\delta=1/2$ and $\lambda=0.96$.

9.12 Given a DSP system with a sampling rate set up to 8000 samples per second in Problem 9.10, implement adaptive filter with five taps for system modeling, assuming that the unknown system transfer function is

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

(a) Determine the DSP equations using the LMS algorithm

$y(n) =$
$e(n) =$
$w_k =$

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

$w_0 =$
$w_1 =$
$w_2 =$
$w_3 =$
$w_4 =$

(b) Determine equations using the RLS algorithm.

9.13 Given a DSP system set up for noise cancellation application with a sampling rate set up to 8000 Hz, as shown in Fig. 9.24, the desired signal of a 1000-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.
(e) Set up equations using the RLS algorithm.

9.14 Given a DSP system for noise cancellation application with two (2) taps in Fig. 9.25:
(a) Set up the LMS algorithm for the adaptive filter.
(b) Given the following inputs and outputs:

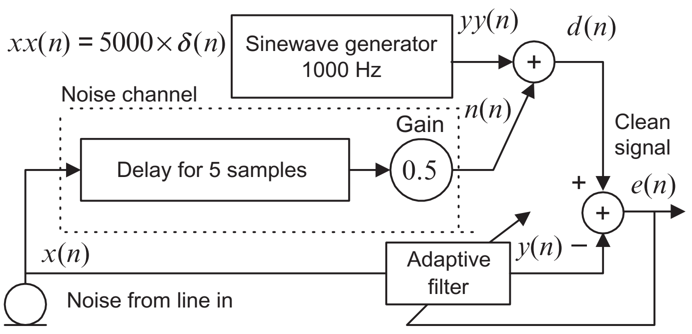


FIG. 9.24

Noise cancellation in Problem 9.13.

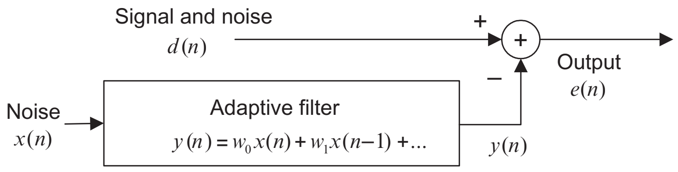


FIG. 9.25

Noise cancellation in Problem 9.14.

$x(0)=1$, $x(1)=1$, $x(2)=-1$, $x(3)=2$, $d(0)=0$, $d(1)=2$, $d(2)=-1$, and $d(3)=1$ and initial weights: $w_0=w_1=0$, convergence factor is set to be $\mu=0.1$, perform adaptive filtering to obtain outputs $e(n)$ for $n=0, 1, 2$.

9.15 Given a DSP system for noise cancellation application with three (3) taps in Fig. 9.25:

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

(b) Given the following 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 $d(3)=1$ and initial weights: $w_0=w_1=w_2=0$, convergence factor is set to be $\mu=0.1$, perform adaptive filtering to obtain outputs $e(n)$ for $n=0, 1, 2$.

(c) Determine equations using the RLS algorithm.

(d) Repeat (b) using the RLS algorithm with $\delta=1$ and $\lambda=0.96$.

9.16 For a line enhancement application using the FIR adaptive filter depicted in Fig. 9.26,

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

(b) Given the following inputs and outputs:

$d(0)=-1$, $d(1)=1$, $d(2)=-1$, $d(3)=1$, $d(4)=-1$, $d(5)=1$and $d(6)=-1$ and initial weights: $w_0=w_1=0$, convergence factor is set to be $\mu=0.1$, perform adaptive filtering to obtain outputs $y(n)$ for $n=0, 1, 2, 3, 4$.

(c) Determine equations using the RLS algorithm.

(d) Repeat (b) using the RLS algorithm with $\delta=1$ and $\lambda=0.96$.

9.17 Repeat Problem 9.16 using three-tap FIR filter and $\Delta=3$.

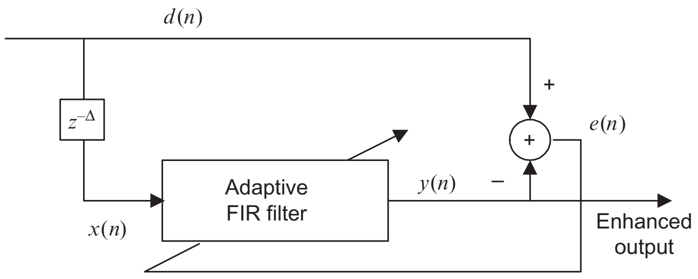


FIG. 9.26

Line enhancement in Problem 9.16.

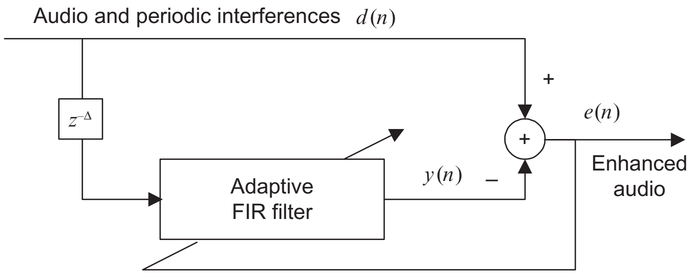


FIG. 9.27

Interference cancellation in Problem 9.18.

9.18 An audio playback application is described in Fig. 9.27.

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 Fig. 9.27.

(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).

(c) Set up equations using the RLS algorithm.

9.19 Repeat Problem 9.18 for the corrupted audio which contains five different periodic interferences.

9.20 In a noisy ECG acquisition environment, the DSP engineer uses an adaptive FIR filter with 20 coefficients to remove 60-Hz interferences. The system is set up as shown in Fig. 9.19, 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.

9.21 Given an application of the echo cancellation as shown in Fig. 9.21B, determine all the difference equations to implement the adaptive filter with four adaptive coefficients at the speaker A site.

9.22 Given an application of the echo cancellation as shown in Fig. 9.21B,

(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.

Computer Problems with MATLAB:

Use MATLAB to solve Problems 9.23–9.26.

9.23 Write a MATLAB program for minimizing the two-weight mean squared error (MSE) 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 derived as

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

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 $J(k)$ vs. the number of iterations, respectively, and summarize your results.

9.24 In Problem 9.10, the unknown system is assumed as 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 8000 Hz.

(a) Generate the input signal for 0.1 s using a sum of three sinusoids having 100, 800, and 1500 Hz with a sampling rate of 8000 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, plot the system input, system output, adaptive filter output, and the error signal, respectively.

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

(e) Repeat (a)–(d) using the RLS algorithm with $\delta=1$ and $\lambda=0.96$.

9.25 Use the following MATLAB code to generate the reference noise and the signal of 300 Hz corrupted by the noise with a sampling rate of 8000 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 = 8000 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 tap: 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.

(d) Repeat (a)–(c) using the RLS algorithm with $\delta=1$ and $\lambda=0.96$.

9.26 A line enhancement system (Fig. 9.28) has following specifications:

Sampling rate = 1000 Hz
Corrupted signal: 100 Hz tone with the unit amplitude added with the unit power white Gaussian noise
Adaptive filter: FIR type, 16 taps
Convergence factor = 0.001
Delay value Δ = to be decided according to the experiment

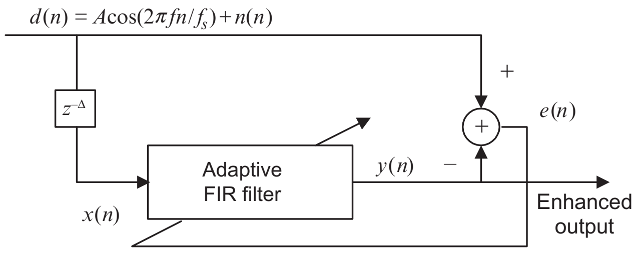


FIG. 9.28

A line enhancement system in Problem 9.26.

LMS algorithm is applied

(a) Write a MATLAB program to perform the line enhancement for 1-s 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 best delay value Δ, which achieves the largest noise reduction.

(c) Repeat (a)–(b) using the RLS algorithm with $\delta=1$ and $\lambda=0.96$.

MATLAB Projects

9.27 Active noise control:

The ANC system is based on the principle of superposition of the primary noise source and secondary source with its acoustic output being of the same amplitude but the opposite phase of the primary noise source, as shown in Fig. 9.29. 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 $S(z)$ 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 $e(n)$ 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 $u(n)$ and $y(n)$ are the filtered reference signal and adaptive filter output, applying the LMS algorithm gives the filtered-x LMS algorithm:

$$w_k = w_k + 2\mu e(n)u(n-k).$$

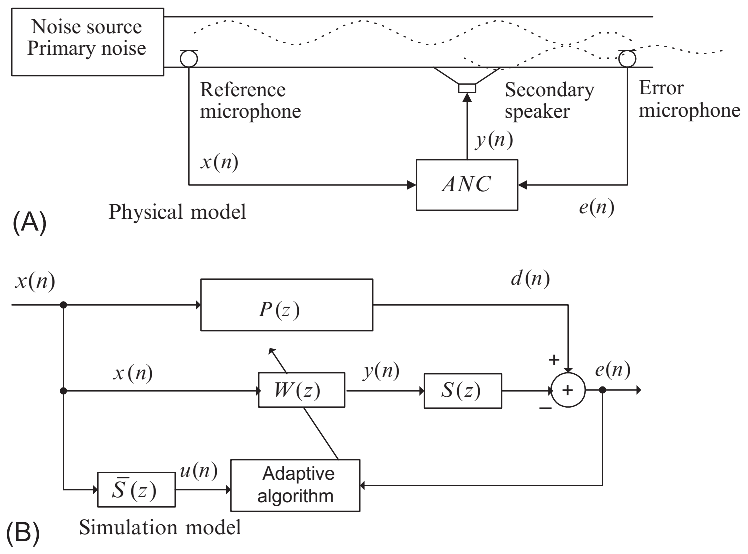


FIG. 9.29

An active noise control system. (A) Physical model, (B) Simulation model.

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 filtered inputs
    $u(n) = \bar{s}_0 x(n) + \bar{s}_1 x(n-1) + \cdots + \bar{s}_{M-1} x(n-M+1)$
(4) Read $e(n)$and update each filter coefficient
    for $k = 0, \ldots, N-1$, $w_k = w_k + 2\mu e(n) u(n-k)$
```

Assuming the following:
Sampling rate = 8000 Hz and simulation duration = 10 s
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: $\bar{S}(z) = S(z) = 0.2 + 0.2z^{-1}$ [can have slight error as compared to $S(z)$]
Residue error signal:

$$e(n) = d(n) - \text{filtering } y(n) \text{ using coefficients of the secondary path } s(n) = d(n) - y(n) * s(n),$$

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

9.28 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 Fig. 9.30, 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

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

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

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

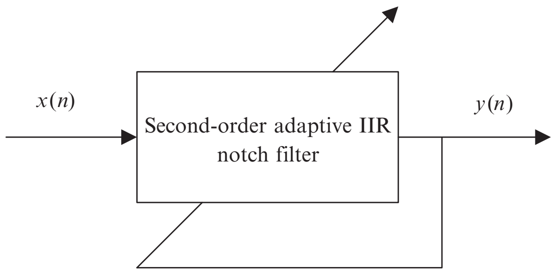


FIG. 9.30

A frequency tracking system.

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

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

and μ is the convergence factor which controls the speed of algorithm convergence.
In this project, plot and verify the notch frequency response by setting $f_s = 8000\,\text{Hz}$, $f = 1000\,\text{Hz}$, and $r = 0.95$. Then generate the sinusoid with duration of 10 s, frequency of 1000 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 squared error function $E[y^2(n)]$ has a one global minimum (one best solution when the LMS algorithm converges). Details of adaptive notch filter can be found in the reference (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 in.

9.29 Channel equalization:
Channel equalization or inverse modeling is depicted in Fig. 9.31. An adaptive filter (equalizer) is utilized to compensate for the linear distortion caused by the channel. For noise-free case: $n(n) = 0$,
we wish

$$H(z)W(z) = z^{-L}$$

where $H(z)$ and $W(z)$ are the unknown channel transfer function and the adaptive filter. L is the time delay, which is best chosen to be half of time span of the equalizer. In this project, only the FIR adaptive filter is considered. The channel $H(z)$ usually causes the symbols $s(n)$ dispersed after they arrive at the receiver. $x(n)$, the received symbols, may interfere with each other. This effect will cause the wrong information after detection. Equalizer uses $x(n)$ to recover $s(n-L)$. This type equalization usually involves two steps: training step and decision step. In the training step, a previously chosen training signal (pseudo-noise sequence long enough to allow the equalizer to compensate for channel distortion) is transmitted through the channel $H(z)$. The received symbols $x(n)$ are used for reference signal for the adaptive filter while the properly delayed symbols $s(n-L)$ are employed for the designed signal. After the adaptive filter converges, the optimal

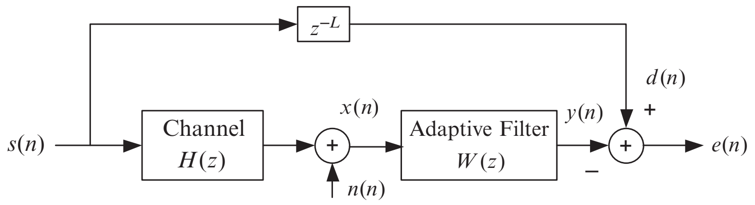


FIG. 9.31

Channel equalization system.

coefficients $W(z)$ are found. Step 2 is the decision stage. The equalizer operates using real communication symbols.

In this project, assuming

$$H(z) = \frac{1}{z-0.8},\ n(n) = 0,\ \text{and adaptive filter length} = 32,$$

Stage 1: Generate 1000 symbols (+1 and -1): s = 2*round(rand(1,1000))-1

(a) Choose the best delay L to design the equalizer using the LMS algorithm.

(b) Choose the best delay L to design the equalizer using the RLS algorithm.

Stage 2: Generate another 1000 symbols (+1 and -1): s = 2*round(rand(1,1000))-1

Use the designed equalizer (a) or (b) to obtain the equalized symbols

$$\hat{s}(n) = \begin{cases} 1 & y(n) \geq 0 \\ -1 & y(n) < 0 \end{cases}.$$

Delay the test symbols: $s_d(n) = s(n-L)$,

Verify the error rate: $0.5\sum_{n=L+1}^{1000} |s_d(n) - \hat{s}(n)|/(1000-L)$.

Advanced Problems

9.30 Determine the following

(a) $E\{\sin n\Omega\}$

(b) $E\{e^{jn\Omega}\}$

(c) $E\{(A\sin n\Omega)^2\}$

(d) $E\{(A\cos n\Omega)^2\}$

(e) $E\{(A\cos n\Omega)(A\sin n\Omega)\}$

(f) $E\{[A\cos (n-n_0)\Omega]^2\}$

(g) $E\{[A\sin (n-n_0)\Omega][A\cos(n)\Omega]\}$

where n is the time index and n_0 is the fixed number of samples, and $\Omega \neq 0$.

9.31 The following Wiener filter is used to predict the sinusoid:

$$d(n) = A\sin(n\Omega),$$

where the Wiener filter predictor with delays of n_1, n_2, and n_3 is given as

$$y(n) = w_1 d(n-n_1) + w_2 d(n-n_2) + w_3 d(n-n_3).$$

Find the Wiener filter coefficients: w_1, w_2, and w_3.

9.32 Given the MSE function as

$$J = 16 + 2w_0^2 + w_1^2 - 2w_0 + 2w_1.$$

Determine the optimal weights and $J_{\min}$.

9.33 Given the MSE function as

$$J = 16 + 2w_0^2 + w_1^2 - 2w_0 + 2w_1 - 2w_0 w_1.$$

Determine the optimal weights and $J_{\min}$.

9.34 Given the MSE function as

$$J = 16 + 2w_0^2 + w_1^2 - 2w_0 + 2w_1.$$

(a) Set up the steepest decent algorithm.

(b) For initial weights as $w_0(0) = 0$ and $w_1(0) = 0$, and $\mu = 0.1$, use MATLAB to find the optimal solution by iterating 100 times and plot the trajectory of $w_0(n)$ and $w_1(n)$ on the MSE contour surface.

9.35 Given the MSE function as

$$J = 16 + 2w_0^2 + w_1^2 - 2w_0 + 2w_1 - 2w_0w_1.$$

(a) Set up the steepest decent algorithm.

(b) For initial weights as $w_0(0) = 1$ and $w_1(0) = 0$, and $\mu = 0.1$, use MATLAB to find the optimal solution by iterating 100 times and plot the trajectory of $w_0(n)$ and $w_1(n)$ on the MSE contour surface.

9.36 For predicting a sinusoid $x(n) = A\sin(n\Omega)$ using a forward predictor

$$x(n) = -a_1x(n-n_1) - a_2x(n-n_2) \text{ for } 0 < n_1 < n_2.$$

Show that the predictor coefficients are as follows:

$$\begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = -\begin{bmatrix} \dfrac{\sin(n_2\Omega)}{\sin[(n_2-n_1)\Omega)]} \\ -\dfrac{\sin(n_1\Omega)}{\sin[(n_2-n_1)\Omega)]} \end{bmatrix} \text{ for } (n_2-n_1)\Omega \neq k\pi \text{ for } k = 0, \pm 1, \pm 2, \cdots$$

9.37 A fourth-order forward linear predictor is used to predict the sinusoids: $x(n) = \sqrt{2}\sin(\pi n/8) + \cos(\pi n/16)$.

For the one-step forward linear predictor, find the predictor coefficients and show the sinusoid can be fully predicted by showing $E\{e_f^2(n)\} = 0$.

9.38 Given the matrix inversion lemma: $A^{-1} = B - BC(D + C^TBC)^{-1}C^TB$

Verify the matrix inversion lemma using the following:

$$A = \begin{bmatrix} 2 & -1/2 \\ 0 & 1/2 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 1 \\ 0 & 2 \end{bmatrix}, \quad C = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad D = 1.$$

9.39 For predicting $x(n) = A\sin^2(n\Omega)$ using a forward predictor with a constant

$$\hat{x}(n) = -a_0 - a_1x(n-n_1) - a_2x(n-n_2) \text{ for } 0 < n_1 < n_2.$$

Determine the predictor coefficients.