10 Adaptive digital filters

10.1 When to use adaptive filters and where they have been used 646

10.2 Concepts of adaptive filtering 647

10.3 Basic Wiener filter theory 651

10.4 The basic LMS adaptive algorithm 654

10.5 Recursive least squares algorithm 662

10.6 Application example 1 – adaptive filtering of ocular artefacts from the human EEG 666

10.7 Application example 2 – adaptive telephone echo cancellation 668

10.8 Other applications 670

Problems 674

References 674

Bibliography 675

Appendices 676

An adaptive filter is essentially a digital filter with self-adjusting characteristics. It adapts, automatically, to changes in its input signals. Adaptive filters are the central topic in the sub-area of DSP known as adaptive signal processing. This chapter describes key aspects of this important topic based on the LMS (least mean square) and RLS (recursive least squares) algorithms which are two of the most widely used algorithms in adaptive signal processing. The treatment is practical with only the essential theory included in the main text. C language implementations of a variety of LMS and RLS based adaptive filters can be found on the CD included with the companion manual – *A Practical Guide for MATLAB and C Language Implementations of DSP Algorithms* (see the Preface for details). A number of real-world applications are presented.

10.1 When to use adaptive filters and where they have been used

The contamination of a signal of interest by other unwanted, often larger, signals or noise is a problem often encountered in many applications. Where the signal and noise occupy fixed and separate frequency bands, conventional linear filters with fixed coefficients are normally used to extract the signal. However, there are many instances when it is necessary for the filter characteristics to be variable, adapted to changing signal characteristics, or to be altered intelligently. In such cases, the coefficients of the filter must vary and cannot be specified in advance. Such is the case where there is a spectral overlap between the signal and noise (see Figure 10.1) or if the band occupied by the noise is unknown or varies with time. Typical applications where fixed coefficient filters are inappropriate are the following:

(1) Electroencephalography (EEG), where artefacts or signal contamination produced by eye movements or blinks is much larger than the genuine electrical activity of the brain and shares the same frequency band with signals of clinical interest. It is not possible to use conventional linear filters to remove the artefacts while preserving the signals of clinical interest.

(2) Digital communication using a spread spectrum, where a large jamming signal, possibly intended to disrupt communication, could interfere with the desired signal. The interference often occupies a narrow but unknown band within the wideband spectrum, and can only be effectively dealt with adaptively.

(3) In digital data communication over the telephone channel at a high rate. Signal distortions caused by the poor amplitude and phase response characteristics of the channel lead to pulses representing different digital codes to interfere with each other (intersymbol interference), making it difficult to detect the codes reliably at the receiving end. To compensate for the channel distortions which may be varying with time or of unknown characteristics at the receiving end, adaptive equalization is used.

An adaptive filter has the property that its frequency response is adjustable or modifiable automatically to improve its performance in accordance with some criterion,

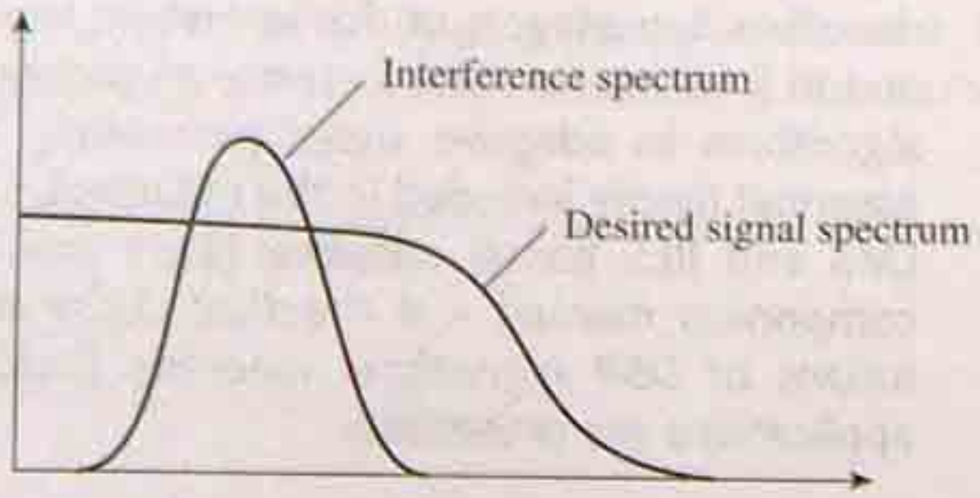


Figure 10.1 An illustration of spectral overlap between a signal and a strong interference.

allowing the filter to adapt to changes in the input signal characteristics. Because of their self-adjusting performance and in-built flexibility, adaptive filters have found use in many diverse applications such as telephone echo cancelling, radar signal processing, navigational systems, equalization of communication channels, and biomedical signal enhancement.

In summary we use adaptive filters

- when it is necessary for the filter characteristics to be variable, adapted to changing conditions;
- when there is spectral overlap between the signal and noise (see Figure 10.1); or
- if the band occupied by the noise is unknown or varies with time.

The use of conventional filters in the above cases would lead to unacceptable distortion of the desired signal. There are many other situations, apart from noise reduction, when the use of adaptive filters is appropriate (see later).

10.2 Concepts of adaptive filtering

10.2.1 Adaptive filters as a noise canceller

An adaptive filter consists of two distinct parts: a digital filter with adjustable coefficients, and an adaptive algorithm which is used to adjust or modify the coefficients of the filter (Figure 10.2). Two input signals, y_k and x_k, are applied simultaneously to the adaptive filter. The signal y_k is the contaminated signal containing both the desired signal, s_k, and the noise, n_k, assumed uncorrelated with each other. The signal, x_k, is a measure of the contaminating signal which is correlated in some way with n_k. x_k is processed by the digital filter to produce an estimate, $\hat{n}_k$, of n_k. An estimate of the desired signal is then obtained by subtracting the digital filter output, $\hat{n}_k$, from the contaminated signal, y_k:

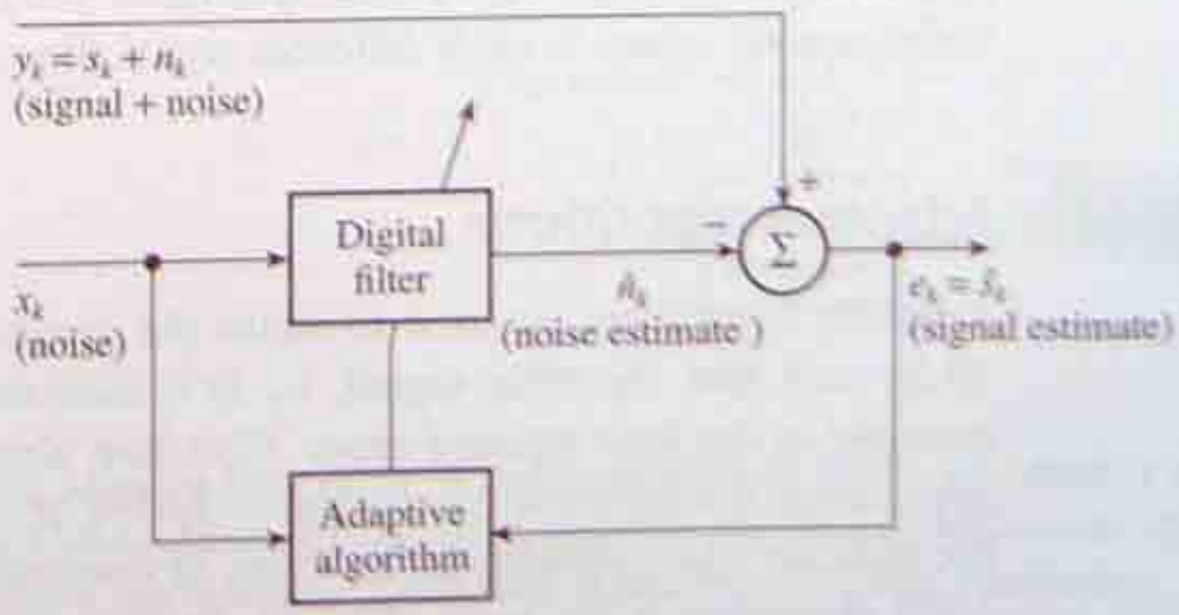


Figure 10.2 Block diagram of an adaptive filter as a noise canceller.

$$\hat{s}_k = y_k - \hat{n}_k = s_k + n_k - \hat{n}_k \tag{10.1}$$

The main objective in noise cancelling is to produce an optimum estimate of the noise in the contaminated signals and hence an optimum estimate of the desired signal. This is achieved by using $\hat{s}_k$ in a feedback arrangement to adjust the digital filter coefficients, via a suitable adaptive algorithm, to minimize the noise in $\hat{s}_k$. The output signal, $\hat{s}_k$, serves two purposes: (i) as an estimate of the desired signal and (ii) as an error signal which is used to adjust the filter coefficients.

10.2.2 Other configurations of the adaptive filter

The discussions above are based on the adaptive noise cancelling principles. It is important to keep in mind that adaptive filters can be and have been used for other purposes, such as for linear prediction, adaptive signal enhancement and adaptive control. In general, the meaning of the signals x_k, y_k, and e_k or the way they are derived are application dependent, a fact which should be borne in mind. Figure 10.3 shows different configurations of the adaptive filter.

10.2.3 Main components of the adaptive filter

In most adaptive systems, the digital filter in Figure 10.2 is realized using a transversal or finite impulse response (FIR) structure (Figure 10.4). Other forms are sometimes used, for example the infinite impulse response (IIR) or the lattice structures, but the FIR structure is the most widely used because of its simplicity and guaranteed stability. For the N-point filter depicted in Figure 10.4, the output is given by

$$\hat{n}_k = \sum_{i=0}^{N-1} w_k(i) x_{k-i} \tag{10.2}$$

where $w_k(i)$, $i = 0, 1, \ldots,$ are the adjustable filter coefficients (or weights), and $x_k(i)$ and $\hat{n}_k$ are the input and output of the filter. Figure 10.4 illustrates the single-input, single-output system. In a multiple-input single-output system, the x_k may be simultaneous inputs from N different signal sources.

10.2.4 Adaptive algorithms

Adaptive algorithms are used to adjust the coefficients of the digital filter (in Figure 10.2) such that the error signal, e_k, is minimized according to some criterion, for example in the least squares sense. Common algorithms that have found widespread application are the least mean square (LMS), the recursive least squares (RLS), and the Kalman filter algorithms. In terms of computation and storage requirements, the LMS algorithm is the most efficient. Further, it does not suffer from the numerical instability problem inherent in the other two algorithms. For these reasons, the LMS

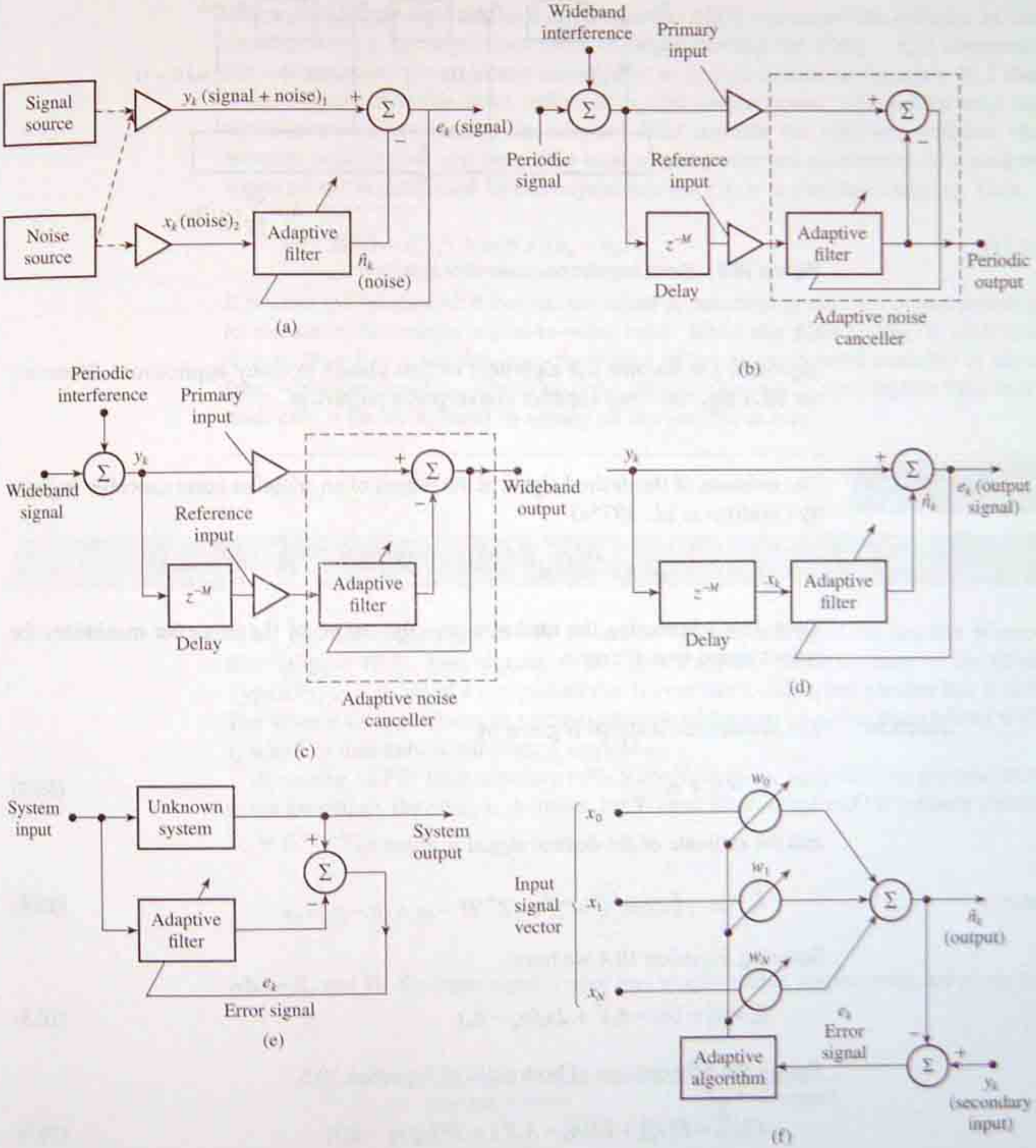


Figure 10.3 Some configurations of the adaptive filter (after Widrow and Winter, 1988): (a) adaptive noise canceller; (b) adaptive self-tuning filter; (c) cancelling periodic interference without an external reference source; (d) adaptive line enhancer; (e) system modelling; (f) linear combiner.

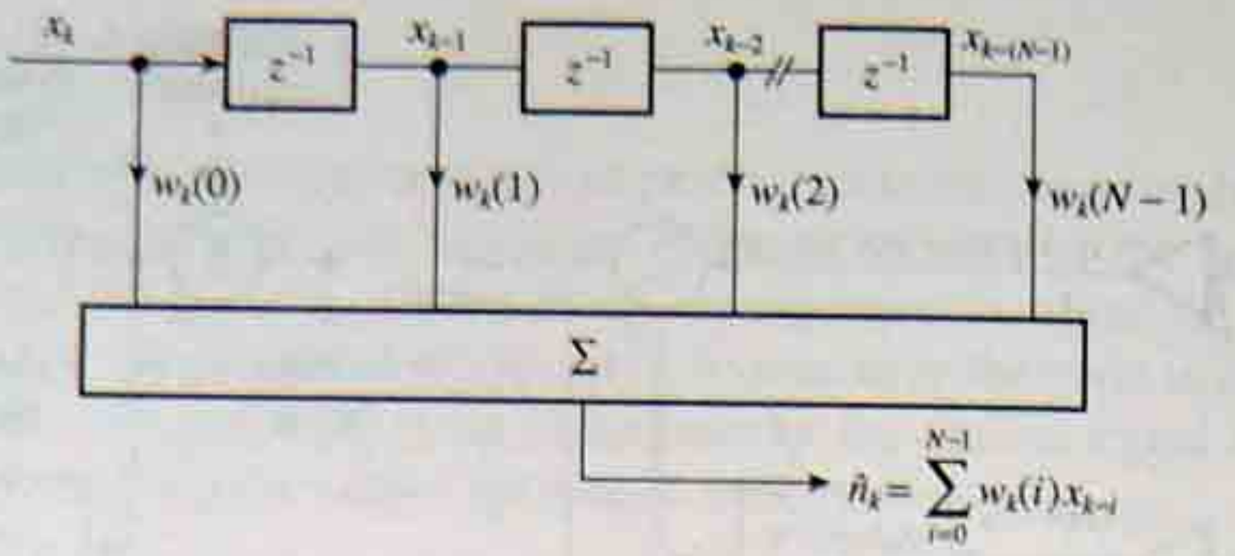


Figure 10.4 Finite impulse response filter structure.

algorithm has become the algorithm of first choice in many applications. However, the RLS algorithm has superior convergence properties.

Example 10.1 The estimate of the desired signal at the output of an adaptive noise canceller is given by (Widrow *et al.*, 1975a)

$$\hat{s}_k = y_k - \hat{n}_k = s_k + n_k - \hat{n}_k$$

Show that minimizing the total power at the output of the canceller maximizes the output signal-to-noise ratio.

Solution The contaminated signal is given by

$$y_k = s_k + n_k \tag{10.3}$$

and the estimate of the desired signal is given by

$$\hat{s}_k = y_k - \hat{n}_k = s_k + n_k - \hat{n}_k \tag{10.4}$$

Squaring Equation 10.4 we have

$$\hat{s}_k^2 = s_k^2 + (n_k - \hat{n}_k)^2 + 2s_k(n_k - \hat{n}_k) \tag{10.5}$$

Taking the expectations of both sides of Equation 10.5,

$$E[\hat{s}_k^2] = E[s_k^2] + E[(n_k - \hat{n}_k)^2] + 2E[s_k(n_k - \hat{n}_k)] \tag{10.6}$$

Since the desired signal, s_k, is uncorrelated with n_k or with $\hat{n}_k$ the last term in Equation 10.6 is zero and we have

$$E[\hat{s}_k^2] = E[s_k^2] + E[(n_k - \hat{n}_k)^2] \tag{10.7}$$

where $E[s_k^2]$ represents the total signal power, $E[\hat{s}_k^2]$ represents the estimate of the signal power (it also represents the total output power) and $E[(n_k - \hat{n}_k)^2]$ represents the remnant noise power which may still be in s_k. It is evident in Equation 10.7 that if the estimate $\hat{n}_k$ is the exact replica of n_k, the output power will contain only the signal power. By adjusting the adaptive filter towards the optimum position, the remnant noise power and hence the total output power are minimized. The desired signal power is unaffected by this adjustment since s_k is uncorrelated with n_k. Thus

$$\min E[\hat{s}_k^2] = E[s_k^2] + \min E[(n_k - \hat{n}_k)^2] \tag{10.8}$$

It is clear in Equation 10.8 that the net effect of minimizing the total output power is to maximize the output signal-to-noise ratio. When the filter setting is such that $\hat{n}_k = n_k$, then $\hat{s}_k = s_k$. In this case, the output of the adaptive noise canceller is noise free. When the signal y_k contains no noise, that is when $n_k = 0$, the adaptive filter turns itself off (in theory at least) by setting all the weights to zero.

10.3 Basic Wiener filter theory

Many adaptive algorithms can be viewed as approximations of the discrete Wiener filter (Figure 10.5). Two signals, x_k and y_k, are applied simultaneously to the filter. Typically, y_k consists of a component that is correlated with x_k and another that is not. The Wiener filter produces an optimal estimate of the part of y_k that is correlated with x_k which is then subtracted from y_k to yield e_k.

Assuming an FIR filter structure with N coefficients (or weights – the popular term in the literature), the error, e_k, between the Wiener filter output and the primary signal, y_k, is given by

$$e_k = y_k - \hat{n}_k = y_k - \mathbf{W}^T\mathbf{X}_k = y_k - \sum_{i=0}^{N-1} w(i)x_{k-i} \tag{10.9}$$

where $\mathbf{X}_k$ and $\mathbf{W}$, the input signal vector and weight vector, respectively, are given by

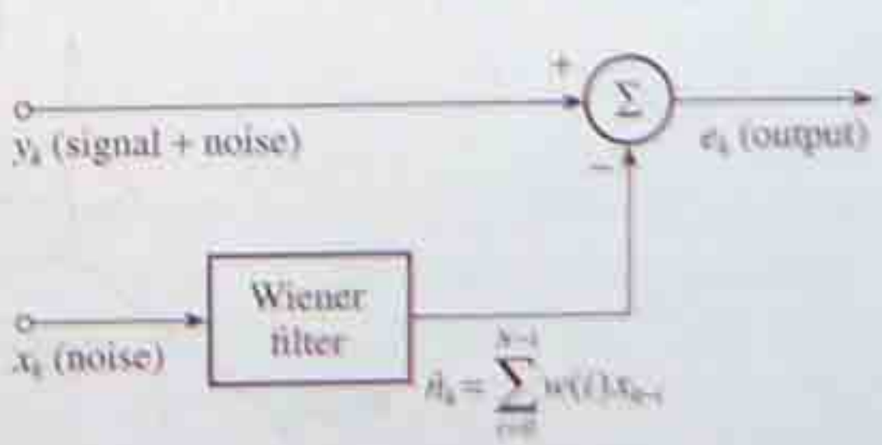


Figure 10.5 The basic Wiener filter.

$$\mathbf{X}_k = \begin{bmatrix} x_k \\ x_{k-1} \\ \vdots \\ x_{k-(N-1)} \end{bmatrix} \qquad \mathbf{W} = \begin{bmatrix} w(0) \\ w(1) \\ \vdots \\ w(N-1) \end{bmatrix} \tag{10.10}$$

The square of the error is given as

$$e_k^2 = y_k^2 - 2y_k\mathbf{X}_k^{\mathrm{T}}\mathbf{W} + \mathbf{W}^{\mathrm{T}}\mathbf{X}_k\mathbf{X}_k^{\mathrm{T}}\mathbf{W} \tag{10.11}$$

The mean square error (MSE), J, is obtained by taking the expectations of both sides of Equation 10.11, assuming that the input vector $\mathbf{X}_k$ and the signal y_k are jointly stationary:

$$\begin{aligned} J &= E[e_k^2] = E[y_k^2] - 2E[y_k\mathbf{X}_k^{\mathrm{T}}\mathbf{W}] + E[\mathbf{W}^{\mathrm{T}}\mathbf{X}_k\mathbf{X}_k^{\mathrm{T}}\mathbf{W}] \\ &= \sigma^2 + 2\mathbf{P}^{\mathrm{T}}\mathbf{W} + \mathbf{W}^{\mathrm{T}}\mathbf{R}\mathbf{W} \end{aligned} \tag{10.12}$$

where $E[\cdot]$ symbolizes expectation, $\sigma^2 = E[y_k^2]$ is the variance of y_k, $\mathbf{P} = E[y_k\mathbf{X}_k]$ is the N length cross-correlation vector and $\mathbf{R} = E[\mathbf{X}_k\mathbf{X}_k^{\mathrm{T}}]$ is the $N \times N$ autocorrelation matrix. A plot of the MSE against the filter coefficients, $\mathbf{W}$, is bowl shaped with a unique bottom (see Figure 10.6). This figure is known as the performance surface and is non-negative. The gradient of the performance surface is given by

$$\nabla = \frac{\mathrm{d}J}{\mathrm{d}\mathbf{W}} = -2\mathbf{P} + 2\mathbf{R}\mathbf{W} \tag{10.13}$$

Each set of coefficients, $w(i)$ $(i = 0, 1, \ldots, N-1)$, corresponds to a point on the surface. At the minimum point of the surface, the gradient is zero and the filter weight vector has its optimum value, $\mathbf{W}_{\mathrm{opt}}$ (see Example 10.2):

$$\mathbf{W}_{\mathrm{opt}} = \mathbf{R}^{-1}\mathbf{P} \tag{10.14}$$

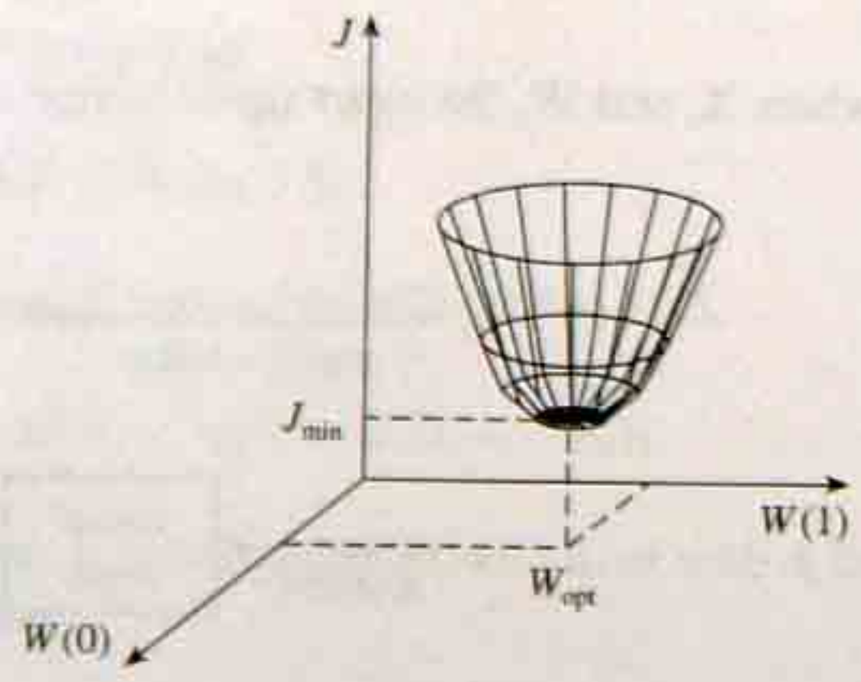


Figure 10.6 Error–performance surface.

Equation 10.14 is known as the Wiener–Hopf equation or solution. The task in adaptive filtering is to adjust the filter weights, $w(0)$, $w(1)$, . . . , using a suitable algorithm, to find the optimum point on the performance surface.

The Wiener filter has a limited practical usefulness because

- it requires the autocorrelation matrix, **R**, and the cross-correlation vector, **P**, both of which are not known *a priori*;
- it involves matrix inversion, which is time consuming; and
- if the signals are nonstationary, then both **R** and **P** will change with time and so $\mathbf{W}_{opt}$ will have to be computed repeatedly.

For real-time application, a way of obtaining $\mathbf{W}_{opt}$ on a sample-by-sample basis is required. Adaptive algorithms are used to achieve this without having to compute **R** and **P** explicitly or performing a matrix inversion.

Example 10.2 Starting with the equation for the mean square error (Equation 10.12), derive the Wiener–Hopf equation.

Solution The MSE is given by

$$\text{MSE} = J = \sigma^2 + 2\mathbf{P}^T\mathbf{W} + \mathbf{W}^T\mathbf{R}\mathbf{W} \tag{10.15}$$

The gradient, ∇, of the MSE is obtained by differentiating the MSE with respect to the weight vector **W**, and setting the result to zero (Haykin, 1986):

$$\nabla = \frac{dJ}{d\mathbf{W}} = \frac{d\sigma^2}{d\mathbf{W}} + \frac{d(\mathbf{P}^T\mathbf{W})}{d\mathbf{W}} + \frac{d(\mathbf{W}^T\mathbf{R}\mathbf{W})}{d\mathbf{W}} \tag{10.16}$$

Now,

$$\frac{d\sigma^2}{d\mathbf{W}} = 0$$

$$\frac{d(2\mathbf{P}^T\mathbf{W})}{d\mathbf{W}} = -2\mathbf{P}$$

$$\frac{d(\mathbf{W}^T\mathbf{R}\mathbf{W})}{d\mathbf{W}} = 2\mathbf{R}\mathbf{W}$$

Using these results, and setting $\nabla = \mathbf{0}$, Equation 10.16 becomes

$$\nabla = \frac{dJ}{d\mathbf{W}} = -2\mathbf{P} + 2\mathbf{R}\mathbf{W} = \mathbf{0} \tag{10.17}$$

The optimum coefficient vector is then given by

$$\mathbf{W}_{opt} = \mathbf{R}^{-1}\mathbf{P} \tag{10.18}$$

10.4 The basic LMS adaptive algorithm

One of the most successful adaptive algorithms is the LMS algorithm developed by Widrow and his coworkers (Widrow *et al.*, 1975a). Instead of computing $\mathbf{W}_{opt}$ in one go as suggested by Equation 10.18, in the LMS the coefficients are adjusted from sample to sample in such a way as to minimize the MSE. This amounts to descending along the surface of Figure 10.6 towards its bottom.

The LMS is based on the steepest descent algorithm where the weight vector is updated from sample to sample as follows:

$$\mathbf{W}_{k+1} = \mathbf{W}_k - \mu\nabla_k \tag{10.19}$$

where $\mathbf{W}_k$ and ∇_k are the weight and the true gradient vectors, respectively, at the kth sampling instant. μ controls the stability and rate of convergence.

The steepest descent algorithm in Equation 10.19 still requires knowledge of $\mathbf{R}$ and $\mathbf{P}$, since ∇_k is obtained by evaluating Equation 10.17. The LMS algorithm is a practical method of obtaining estimates of the filter weights $\mathbf{W}_k$ in real time without the matrix inversion in Equation 10.18 or the direct computation of the autocorrelation and cross-correlation. The Widrow–Hopf LMS algorithm for updating the weights from sample to sample is given by

$$\mathbf{W}_{k+1} = \mathbf{W}_k + 2\mu e_k \mathbf{X}_k \tag{10.20a}$$

where

$$e_k = y_k - \mathbf{W}_k^T\mathbf{X}_k \tag{10.20b}$$

Clearly, the LMS algorithm above does not require prior knowledge of the signal statistics (that is the correlations $\mathbf{R}$ and $\mathbf{P}$), but instead uses their instantaneous estimates (see Example 10.3). The weights obtained by the LMS algorithm are only estimates, but these estimates improve gradually with time as the weights are adjusted and the filter learns the characteristics of the signals. Eventually, the weights converge. The condition for convergence is

$$0 < \mu > 1/\lambda_{max} \tag{10.21}$$

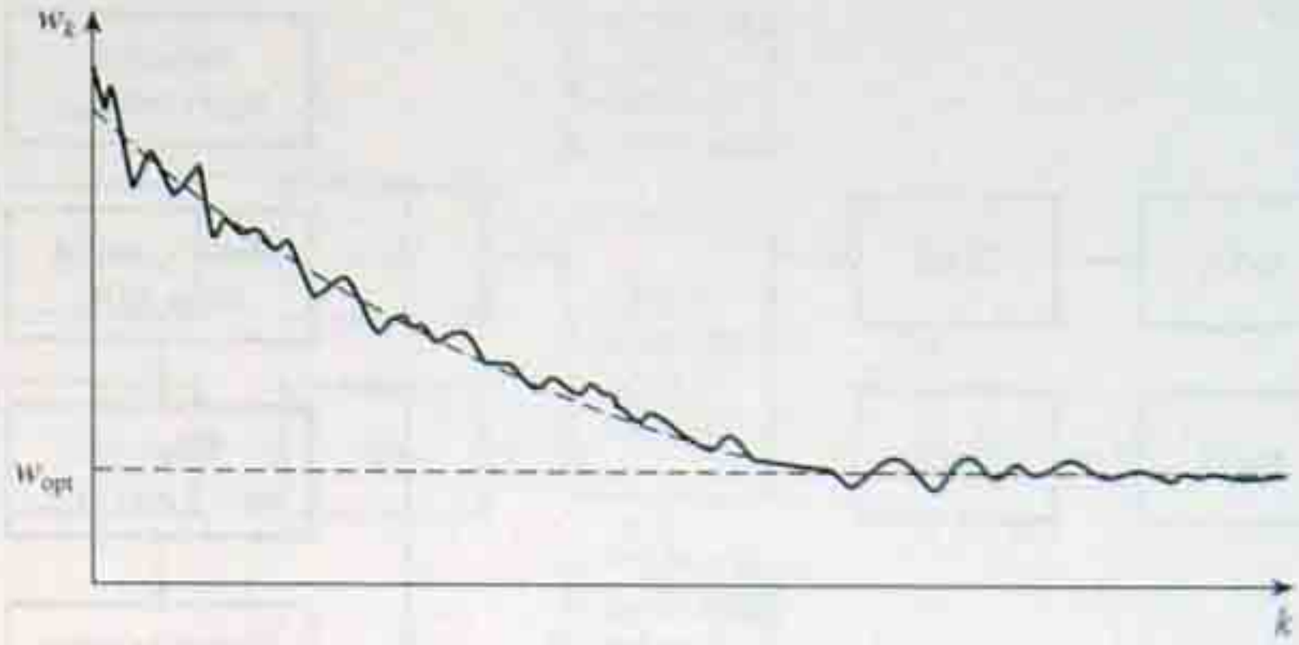


Figure 10.7 An illustration of the variations in the filter weights.

where λ_{max} is the maximum eigenvalue of the input data covariance matrix. In practice, $\mathbf{W}_k$ never reaches the theoretical optimum (the Wiener solution), but fluctuates about it (see Figure 10.7).

10.4.1 Implementation of the basic LMS algorithm

The computational procedure for the LMS algorithm is summarized below.

(1) Initially, set each weight $w_k(i)$, $i = 0, 1, \ldots, N-1$, to an arbitrary fixed value, such as 0.

For each subsequent sampling instant, $k = 1, 2, \ldots$, carry out steps (2) to (4) below:

(2) compute filter output

$$\hat{n}_k = \sum_{i=0}^{N-1} w_k(i) x_{k-i}$$

(3) compute the error estimate

$$e_k = y_k - \hat{n}_k$$

(4) update the next filter weights

$$w_{k+1}(i) = w_k(i) + 2\mu e_k x_{k-i}$$

The simplicity of the LMS algorithm and ease of implementation, evident from above, make it the algorithm of first choice in many real-time systems. The LMS algorithm requires approximately $2N + 1$ multiplications and $2N + 1$ additions for each new set of input and output samples. Most signal processors are suited to the mainly multiply–accumulate arithmetic operations involved, making a direct implementation of the LMS algorithm attractive.

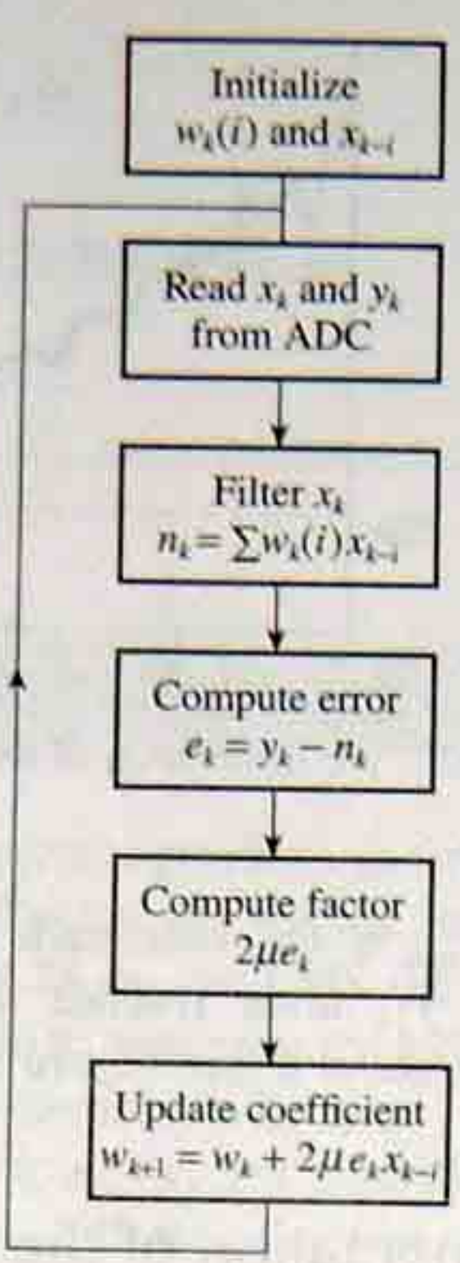


Figure 10.8 Flowchart for the LMS adaptive filter.

```
Inputs:    xk(i)   vector of the latest input samples
           yk      current contaminated signal sample
           wk(i)   vector of filter coefficients
Outputs:   ek      current desired output (or error) sample
           wk(i)   vector of updated filter coefficients

/. compute the current error estimate ./

ek = yk
for i=1 to N do
  ek = ek – xk(i) • wk(i)
end

/. update filter coefficients ./

gk = 2u • ek
for i = 1 to N do
  wk(i) = wk(i) + xk(i) • gk
end

return
```

Figure 10.9 Coding of the LMS adaptive filter.

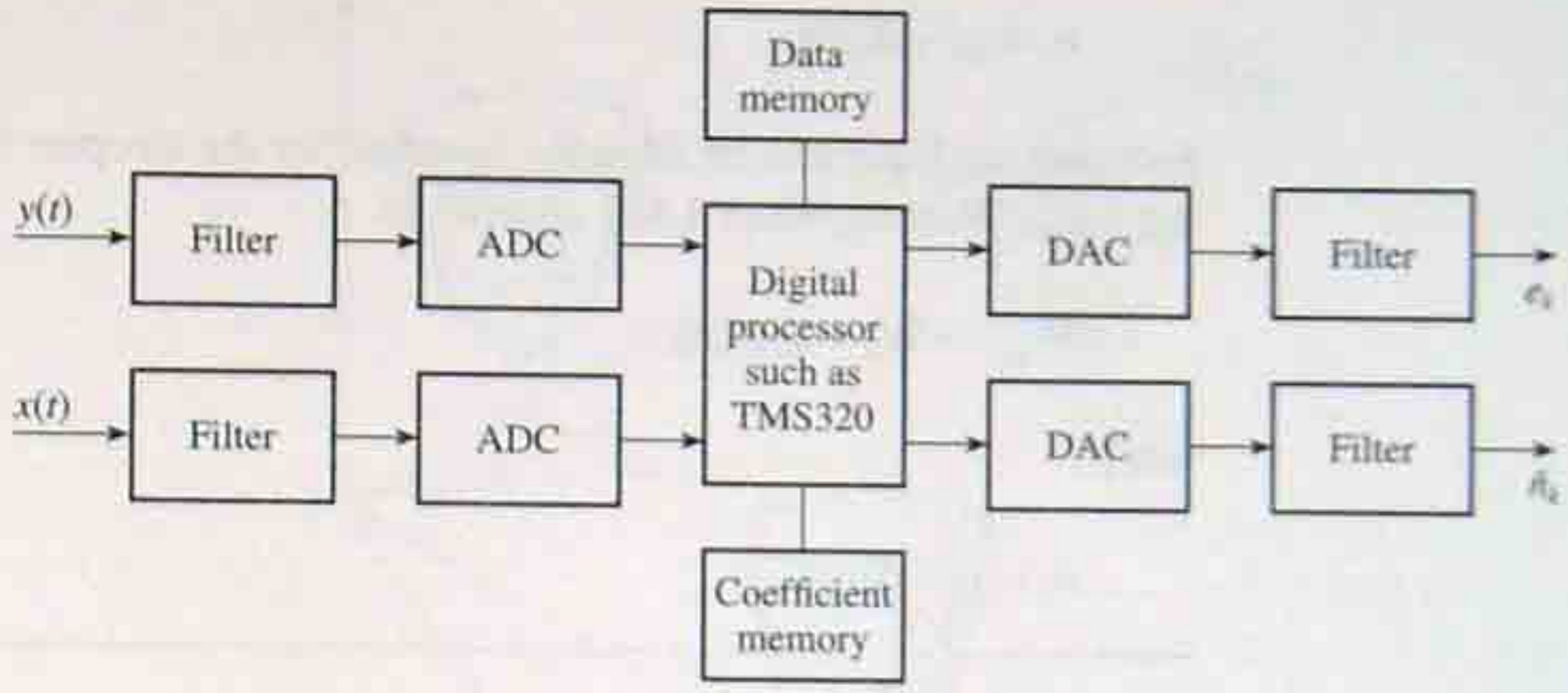


Figure 10.10 Hardware implementation for real-time LMS adaptive filtering.

The flowchart for the LMS algorithm is given in Figure 10.8. Figures 10.9 and 10.10, respectively, show a pseudo-code for the software and hardware implementations. A C language implementation of the LMS algorithm is given in the appendix.

Example 10.3 Starting with the steepest descent algorithm

$$\mathbf{W}_{k+1} = \mathbf{W}_k - \mu\nabla_k$$

where $\mathbf{W}_k$ is the filter weight vector at the kth sampling instant, μ controls stability and rate of convergence and ∇_k is the true gradient of the error–performance surface, derive the Widrow–Hopf LMS algorithm for adaptive noise cancelling, stating any reasonable assumptions made.

Solution The steepest descent algorithm is given by

$$\mathbf{W}_{k+1} = \mathbf{W}_k - \mu\nabla_k \tag{10.22}$$

The gradient vector, ∇, the cross-correlation between the primary and secondary inputs, $\mathbf{P}$, and the autocorrelation of the primary input, $\mathbf{R}$, are related as

$$\nabla = -2\mathbf{P} + 2\mathbf{RW} \tag{10.23}$$

In the LMS algorithm, instantaneous estimates are used for ∇. Thus

$$\begin{aligned}\nabla_k &= -2\mathbf{P}_k + 2\mathbf{R}_k\mathbf{W}_k = -2\mathbf{X}_k y_k + 2\mathbf{X}_k\mathbf{X}_k^{\mathrm{T}}\mathbf{W}_k \\ &= -2\mathbf{X}_k(y_k - \mathbf{X}_k^{\mathrm{T}}\mathbf{W}_k) = -2e_k\mathbf{X}_k\end{aligned} \tag{10.24}$$

where

$$e_k = y_k - \mathbf{X}_k^T \mathbf{W}_k$$

Substituting Equation 10.24 in the equation for the steepest descent algorithm we have the basic Widrow–Hopf LMS algorithm:

$$\mathbf{W}_{k+1} = \mathbf{W}_k + 2\mu e_k \mathbf{X}_k \tag{10.25a}$$

where

$$e_k = y_k - \mathbf{W}_k^T \mathbf{X}_k \tag{10.25b}$$

10.4.2 Practical limitations of the basic LMS algorithm

In practice, several practical problems are encountered when using the basic LMS algorithm, leading to a lowering of performance. Some of the more important problems are discussed here.

10.4.2.1 Effects of non-stationarity

In a stationary environment, the error performance surface of the filter has a constant shape and orientation, and the adaptive filter merely converges to and operates at or near the optimum point. If the signal statistics change after the weights have converged, the filter responds to the change by re-adjusting its weights to a new set of optimal values, provided that the change in signal statistics is sufficiently slow for the filter to converge between changes. In a non-stationary environment, however, the bottom or minimum point continually moves, and its orientation and curvature may also be changing (see Figure 10.11). Thus the algorithm in this case has the task not only of seeking the minimum point of the surface but also of tracking the changing position, leading to a significant lowering of performance. Note that a variable is said to be stationary if its statistics (such as mean, variance, autocorrelation) change with time. Such changes can result from, for example, sudden changes due to sporadic interference of short duration (Figure 10.12) or bad data, and often upset the filter weights.

A number of schemes have been developed to overcome this problem but these in general tend to increase the complexity of the basic LMS algorithm. One such scheme is the time-sequenced adaptive filter (Ferrara and Widrow, 1981).

10.4.2.2 Effects of signal component on the interference input channel

The performance of the algorithm relies on the measured interference signal, $x_1(i)$, being highly correlated with the actual interference, but weakly correlated (theoretically zero) with the desired signal. In most cases, this condition is not met. In some applications, the contaminating input may contain both the undesired interference as well as low level signal components. This leads to a cancellation of some of the

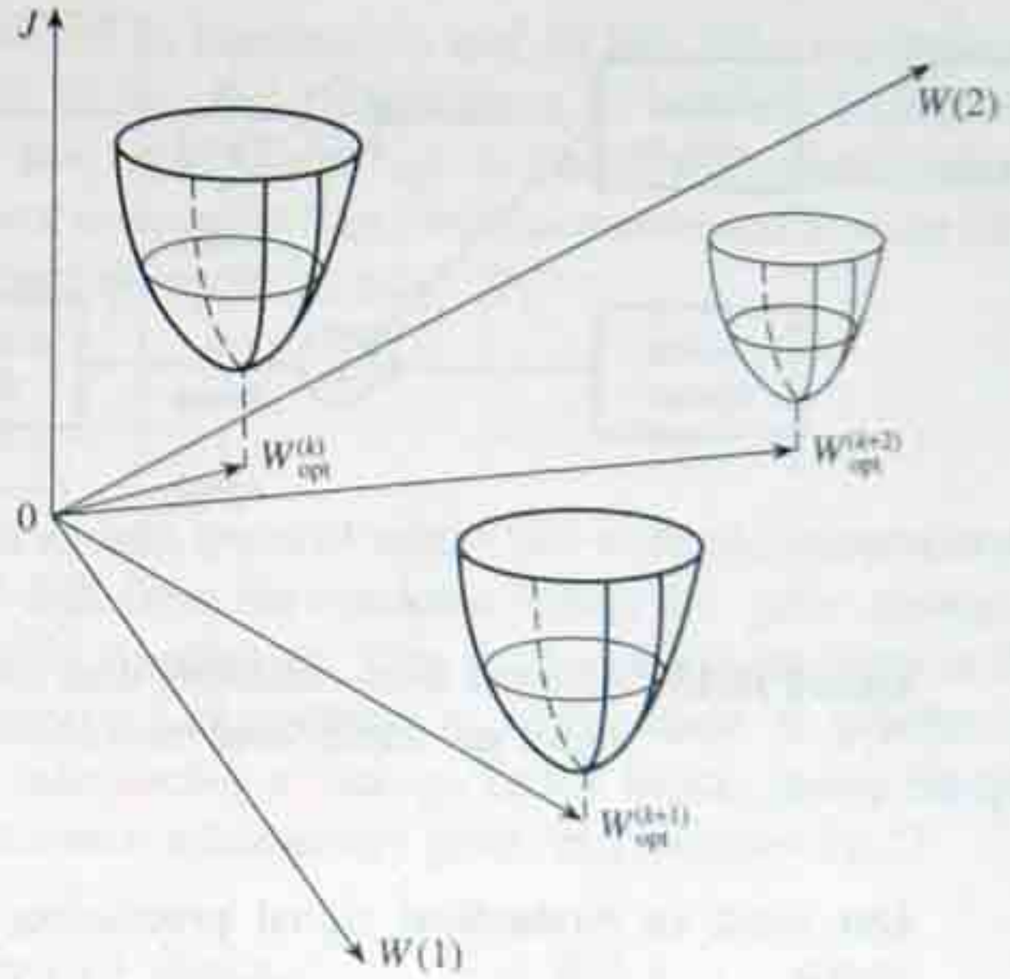
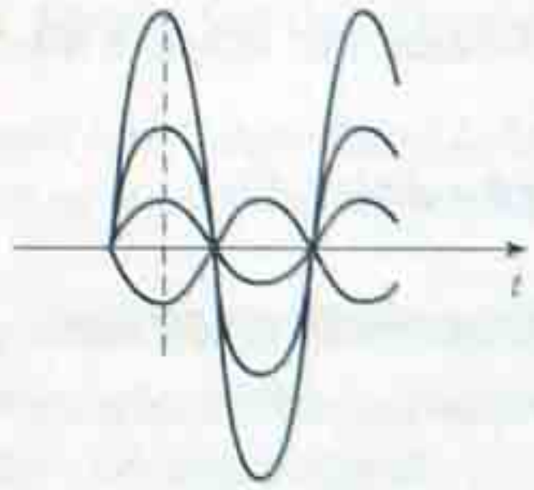
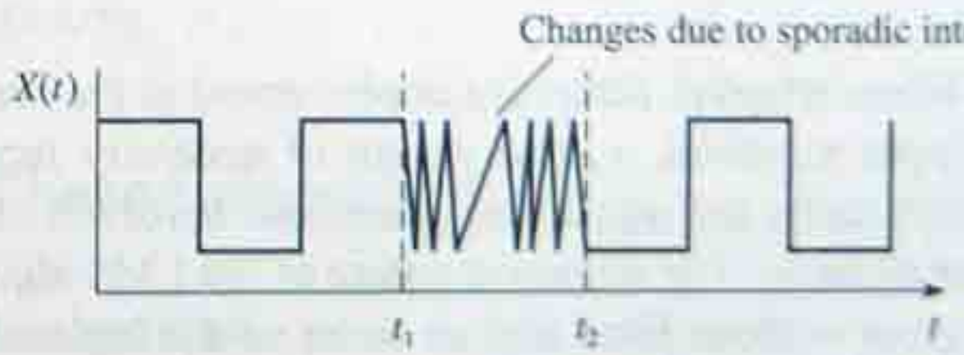


Figure 10.11 Time-varying error–performance surface.



(a)



(b)

Figure 10.12 An illustration of non-stationary processes: (a) modulated waveform; (b) sporadic interference.

desired signal components. Such a situation is illustrated in Figure 10.13. It is shown in Widrow *et al.* (1975a) that the adaptive noise cancelling process still leads to a significant improvement in the desired signal-to-noise ratio in these cases but only at the expense of a small signal distortion. However, if x_k contains only signals and no noise component whatsoever, the desired signal in y_k may be completely obliterated.

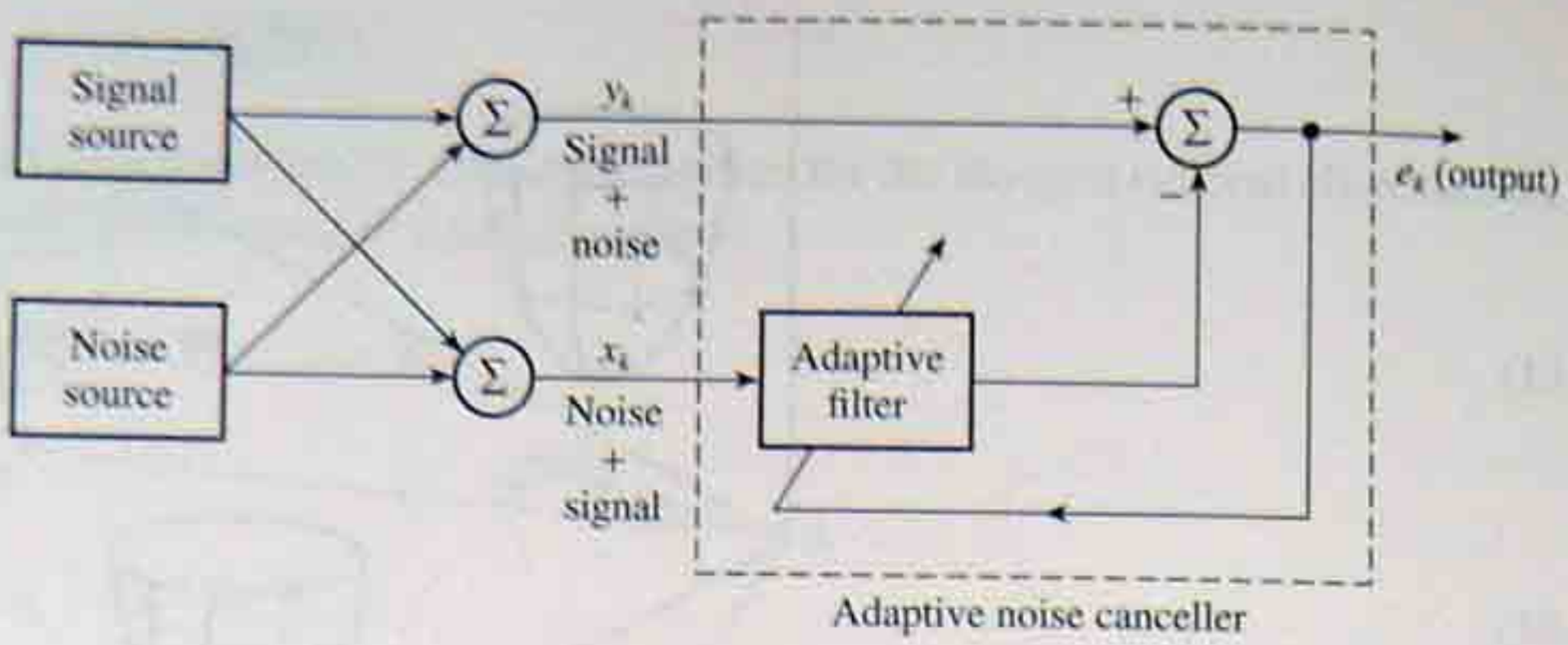


Figure 10.13 Adaptive noise cancelling with some signal components in both the desired signal and interference input channels.

Our work in biomedical signal processing confirms their results (Ifeachor *et al.*, 1986).

10.4.2.3 Computer wordlength requirements

The LMS-based FIR adaptive filter is characterized by the following equations:

for the digital filter
$$\hat{n}_k = \sum_{i=0}^{N-1} w_k(i) x_{k-i} \tag{10.26a}$$

for the adaptive algorithm
$$\mathbf{W}_{k+1} = \mathbf{W}_k + 2\mu e_k \mathbf{X}_k \tag{10.26b}$$

where

$$e_k = y_k - \mathbf{W}_k^{\mathrm{T}} \mathbf{X}_k$$

When adaptive filters are implemented in the real world, the filter weights, w_k, and the input variables, x_k and y_k, are of necessity represented by a finite number of bits. Similarly, the numerical operations involved are carried out using a finite precision arithmetic. The recursive nature of the LMS algorithm means that the wordlength will grow without limit and so some of the bits must be discarded before each updated weight is stored. Thus the y_k, e_k and $w_k(i)$ may differ significantly from their true values. The use of filter weights and results of arithmetic operations with limited accuracy may introduce errors into the adaptive filter whose effects may include (i) possible non-convergence of the adaptive filter to the optimal solution, leading to an inferior performance (for example, if the filter is used as an interference canceller some residual interference may remain), (ii) the filter outputs may contain noise which will cause it to fluctuate randomly, and (iii) a premature termination of the algorithm may occur. Thus a sufficient number of bits should be used to keep these errors at tolerable levels. Most adaptive systems described in the open literature represent the digital signals, x_{k-i} and y_k, as fixed point numbers of between 8 and 16 bits, with the

coefficients quantized to between 16 and 24 bits. The multipliers used range from 8×8 bit to 24×16 bit, and accumulators of between 16 and 40 bits are used. It appears that for low order filters (up to about 100 coefficients) it is sufficient to store the coefficient to no more than 16-bit accuracy and to use a 16×16 bit multiplier with an accumulator of length 32 bits.

10.4.2.4 Coefficient drift

In the presence of certain types of inputs (for example narrowband signals), the filter coefficients may drift from the optimum values and grow slowly, eventually exceeding the permissible wordlength. This is an inherent problem in the LMS algorithm and leads to a long-term degradation in performance. In practice, coefficient drift is counteracted by introducing a leakage factor which gently nudges the coefficients towards zero. Two such schemes are given in Equations 10.27:

$$w_{k+1}(i) = \delta w_k(i) + 2\mu e_k x_{k-i} \qquad 0 < \delta < 1 \tag{10.27a}$$

$$w_{k+1}(i) = w_k(i) + 2\mu e_k x_{k-i} \pm \delta \qquad 0 < \delta < 1 \tag{10.27b}$$

Small δ, the leakage factor, ensures that drift is contained, but introduces bias in the error term, e_k.

The usefulness of the basic LMS algorithm has been extended by more sophisticated LMS-based algorithms as mentioned before. These include

(1) the complex LMS algorithm which allows the handling of complex data;
(2) the block LMS algorithm which offers substantial computational advantages and in some cases faster convergence; and
(3) time-sequenced LMS algorithms to deal with particular types of non-stationarity.

10.4.3 Other LMS-based algorithms

10.4.3.1 Complex LMS algorithm

The complex LMS algorithm for updating the filter weights is given by (Widrow *et al.*, 1975b)

$$\tilde{\mathbf{W}}_{k+1} = \tilde{\mathbf{W}}_k + 2\mu \tilde{e}_k \tilde{\mathbf{X}}_{k-i} \tag{10.28}$$

where the symbol ˜ denotes a complex variable. The Mitel PDSP16XXX processors are ideally suited to the complex LMS algorithm as they can perform arithmetic operations directly on complex data, which is a distinct advantage over conventional processors.

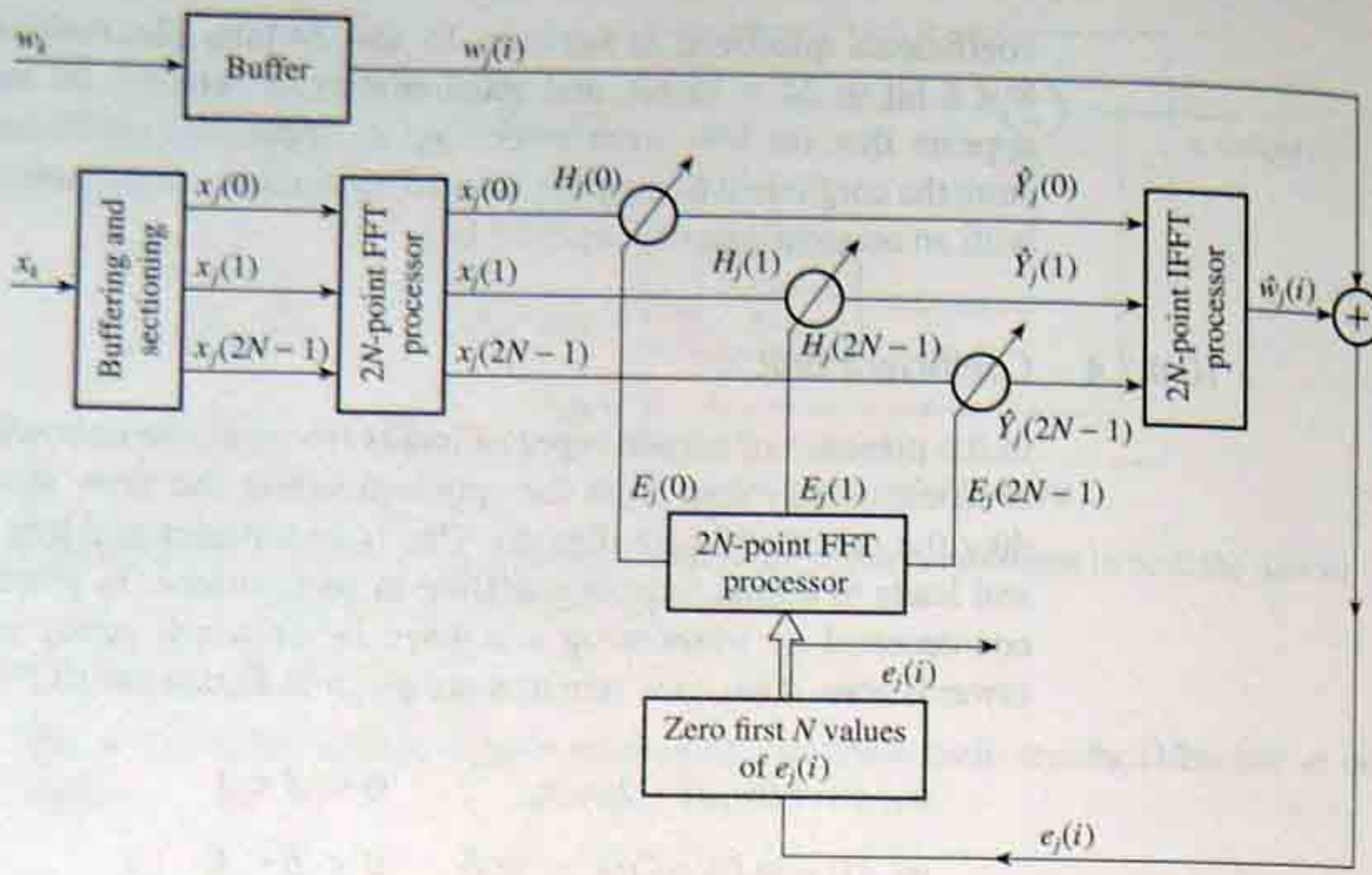


Figure 10.14 Simplified block diagram of a frequency domain LMS filter.

10.4.3.2 Fast LMS algorithms

A number of block LMS algorithms have been proposed which offer substantial computational savings especially when the number of filter coefficients is large. The computational savings result from processing the data in blocks instead of one sample at a time. Frequency domain implementations of the block LMS exploit the computational advantages of the fast Fourier transform (FFT) in performing convolutions (Mansour and Gray, 1982).

An efficient frequency domain filter is depicted in Figure 10.14.

10.5 Recursive least squares algorithm

The RLS algorithm is based on the well-known least squares method (Figure 10.15). An output signal, y_k, is measured at the discrete time, k, in response to a set of input signals, $x_k(i)$, $i = 1, 2, \ldots, n$. The input and output signals are related by the simple regression model

$$y_k = \sum_{i=0}^{n-1} w(i)x_k(i) + e_k \tag{10.29}$$

where e_k represents measurement errors or other effects that cannot be accounted for, and $w(i)$ represents the proportion of the ith input that is contained in the primary

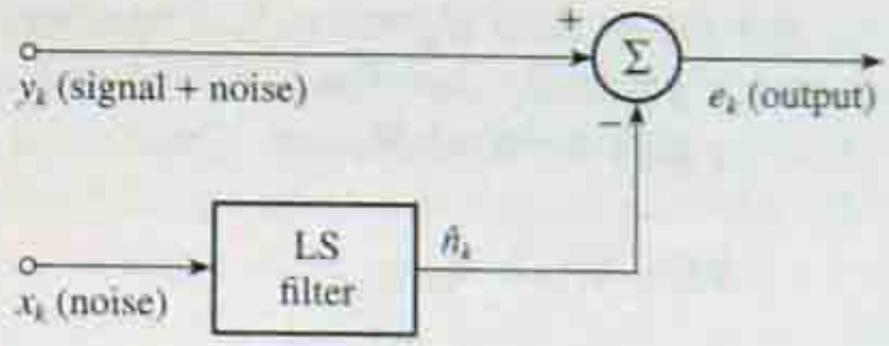


Figure 10.15 An illustration of the basic idea of the least-squares method.

signal, y_k. The problem in the LS method is, given the $x_k(i)$ and y_k above, to obtain estimates of $w(0)$ to $w(n-1)$.

Optimum estimates (in the least squares sense) of the filter weights, $w(i)$, are given by

$$\mathbf{W}_m = [\mathbf{X}_m^{\mathrm{T}}\mathbf{X}_m]^{-1}\mathbf{X}_m^{\mathrm{T}}\mathbf{Y}_m \tag{10.30}$$

where $\mathbf{Y}_m$, $\mathbf{W}_m$ and $\mathbf{X}_m$ are given by

$$\mathbf{Y}_m = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{m-1} \end{bmatrix} \qquad \mathbf{X}_m = \begin{bmatrix} \mathbf{x}^{\mathrm{T}}(0) \\ \mathbf{x}^{\mathrm{T}}(1) \\ \mathbf{x}^{\mathrm{T}}(2) \\ \vdots \\ \mathbf{x}^{\mathrm{T}}(m-1) \end{bmatrix} \qquad \mathbf{W}_m = \begin{bmatrix} w(0) \\ w(1) \\ w(2) \\ \vdots \\ w(n-1) \end{bmatrix}$$

$$\mathbf{x}^{\mathrm{T}}(k) = [x_k(0) \quad x_k(1) \quad \ldots \quad x_k(n-1)], \quad k = 0, 1, \ldots, m-1$$

The suffix m indicates that each matrix above is obtained using all m data points and T indicates transposition. Equation 10.30 gives the OLS estimate of $\mathbf{W}_m$ which can be obtained using any suitable matrix inversion technique. The filter output is then obtained as

$$\hat{n}_k = \sum_{i=0}^{n-1} \hat{w}(i)x_{k-i}, \quad k = 1, 2, \ldots, m \tag{10.31}$$

10.5.1 Recursive least squares algorithm

The computation of $\mathbf{W}_m$ in Equation 10.30 requires the time-consuming computation of the inverse matrix. Clearly, the LS method above is not suitable for real-time or on-line filtering. In practice, when continuous data is being acquired and we wish to improve our estimate of $\mathbf{W}_m$ using the new data, recursive methods are preferred. With the recursive least squares algorithm the estimates of $\mathbf{W}_m$ can be updated for each new set of data acquired without repeatedly solving the time-consuming matrix inversion directly.

A suitable RLS algorithm can be obtained by exponentially weighting the data to remove gradually the effects of old data on $\mathbf{W}_m$ and to allow the tracking of slowly varying signal characteristics. Thus

$$\mathbf{W}_k = \mathbf{W}_{k-1} + \mathbf{G}_k e_k \tag{10.32a}$$

$$\mathbf{P}_k = \frac{1}{\gamma}[\mathbf{P}_{k-1} - \mathbf{G}_k \mathbf{x}^{\mathrm{T}}(k)\mathbf{P}_{k-1}] \tag{10.32b}$$

where

$$\mathbf{G}_k = \frac{\mathbf{P}_{k-1}\mathbf{x}(k)}{\alpha_k}$$

$$e_k = y_k - \mathbf{x}^{\mathrm{T}}(k)\mathbf{W}_{k-1}$$

$$\alpha_k = \gamma + \mathbf{x}^{\mathrm{T}}(k)\mathbf{P}_{k-1}\mathbf{x}(k)$$

$\mathbf{P}_k$ is essentially a recursive way of computing the inverse matrix $[\mathbf{X}_k^{\mathrm{T}}\mathbf{X}_k]^{-1}$.

The argument k emphasizes the fact that the quantities are obtained at each sample point. γ is referred to as the forgetting factor. This weighting scheme reduces to that of the LS when $\gamma = 1$. Typically, γ is between 0.98 and 1. Smaller values assign too much weight to the more recent data, which leads to wildly fluctuating estimates. The number of previous samples that significantly contribute to the value of $\mathbf{W}_k$ at each sample point is called the asymptotic sample length (ASL) given by

$$\sum_{k=0}^{\infty} \gamma^k = \frac{1}{1-\gamma} \tag{10.33}$$

This effectively defines the memory of the RLS filter. When $\gamma = 1$, that is when it corresponds to the LS, the filter has an infinite memory.

10.5.2 Limitations of the recursive least squares algorithm

The RLS method is very efficient and involves exactly the same number of arithmetic operations between samples as $\mathbf{W}_k$ and $\mathbf{P}_k$ in Equation 10.32 have fixed dimensions. This is an important requirement for efficient real-time filtering. There are, however, two main problems that may be encountered when the RLS algorithm is implemented directly. The first, referred to as 'blow-up', results if the signal $x_k(i)$ is zero for a long time, when the matrix $\mathbf{P}_k$ will grow exponentially as a result of division by γ (which is less than unity) at each sample point:

$$\lim_{k\to\infty} \mathbf{P}_k = \lim_{k\to\infty}\left(\frac{\mathbf{P}_{k-1}}{\gamma_{k-1}}\right) \tag{10.34}$$

The second problem with the RLS is its sensitivity to computer roundoff errors, which results in a negative definite **P** matrix and eventually to instability. For successful estimation of **W**, it is necessary that the matrix **P** be positive semi-definite which is equivalent to requiring in the LS method that the matrix $\mathbf{X}^T\mathbf{X}$ be invertible, but, because of differencing of terms in Equation 10.32b, positive definiteness of **P** cannot be guaranteed. This problem can be worse in multiparameter models, especially if the variables are linearly dependent and when the algorithm is implemented on a small system with a finite wordlength. When the algorithm has iterated for a long time the two terms in the parentheses in Equation 10.32b are very nearly equal and subtraction of such terms in a finite wordlength system may lead to errors and a negative definite $\mathbf{P}_k$ matrix.

The problem of numerical instability may be solved by suitably factorizing the matrix **P** such that the differencing of terms in Equation 10.32b is avoided. Such factorization algorithms are numerically better conditioned and have accuracies that are comparable with the RLS algorithm that uses double precision. Two such algorithms are the square root and the UD factorization algorithms. In terms of storage and computation the UD algorithm is more efficient, and is thus preferred. In fact, the UD algorithm is a square-root-free formulation of the square root algorithm and thus shares the same properties as the latter.

10.5.3 Factorization algorithms

10.5.3.1 Square root algorithm

In the square root method, the matrix $\mathbf{P}_k$ is factored as (Peterka, 1975)

$$\mathbf{P}_k = \mathbf{S}_k \mathbf{S}_k^T \tag{10.35}$$

where $\mathbf{S}_k$, an upper triangular matrix, and $\mathbf{S}_k^T$, its transpose, are square roots of $\mathbf{P}_k$. Thus if $\mathbf{S}_k$ instead of $\mathbf{P}_k$ is updated the positive definiteness of $\mathbf{P}_k$ is guaranteed, since the product of two square roots is always positive. $\mathbf{S}_k$ is updated as

$$\mathbf{S}_k = \frac{1}{\gamma^{1/2}} \mathbf{S}_{k-1}\mathbf{H}_{k-1} \tag{10.36}$$

where $\mathbf{H}_k$ is an upper triangular matrix.

10.5.3.2 UD factorization algorithm

In the UD method $\mathbf{P}_k$ is factored as (Bierman, 1976)

$$\mathbf{P}_k = \mathbf{U}_k\mathbf{D}_k\mathbf{U}_k^T$$

where $\mathbf{U}_k$ is a unit upper triangular matrix, $\mathbf{U}_k^T$ is its transpose and $\mathbf{D}_k$ is a diagonal matrix. Thus instead of updating $\mathbf{P}_k$ as in the RLS, its factors **U** and **D** are updated. A C language code for the UD algorithm is given on CD in the companion handbook (see the Preface for details).

10.6 Application example 1 – adaptive filtering of ocular artefacts from the human EEG

10.6.1 The physiological problem

The human electroencephalogram (EEG) is the electrical activity of the brain and contains useful diagnostic information on a variety of neurological disorders. Normal EEG signals are measured from electrodes placed on the scalp, and are often very small in amplitude, of the order of 20 μV. The EEG, like all biomedical signals, is very susceptible to a variety of large signal contaminations or artefacts which reduce its clinical usefulness. For example, blinking or moving the eyes produces large electrical potentials around the eyes called the electrooculogram (EOG). The EOG spreads across the scalp to contaminate the EEG, when it is referred to as an ocular artefact (OA). Examples of measured EOG and the corresponding contaminated EEG are given in Figure 10.16.

Ocular artefacts are a major source of difficulty in distinguishing normal brain activities from abnormal ones. In some cases, for example brain-damaged babies and patients with frontal tumours, it is difficult to distinguish between the associated pathological slow waves in the EEG and OAs. The similarity between the OAs and

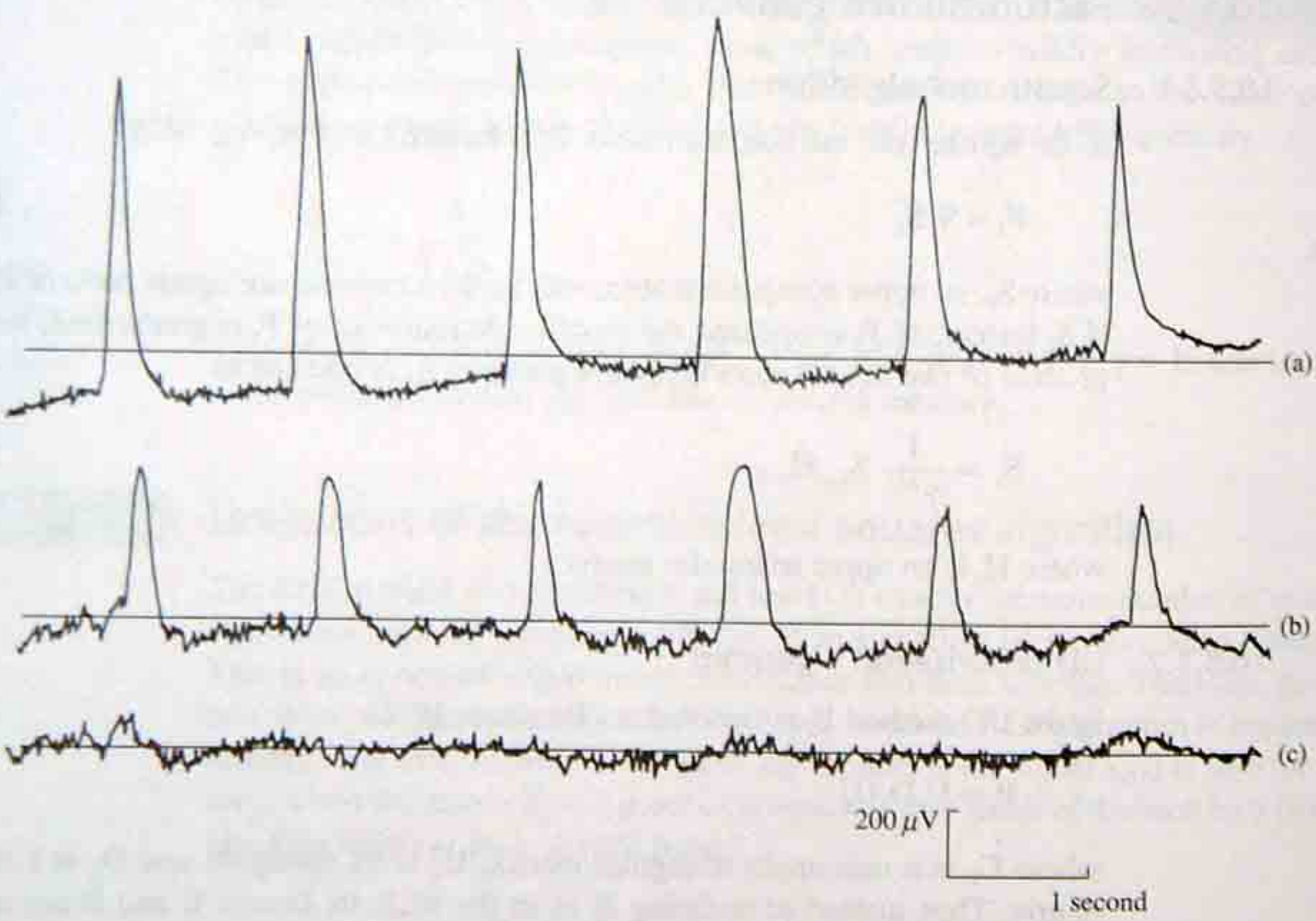


Figure 10.16 The problem of ocular artefacts in electroencephalography: (a) measured EOG, (b) corresponding contaminated EEG signal and (c) EEG signal corrected for artefact.

signals of clinical interest also makes it difficult to automate the analysis of the EEG by computer. In general, neurological disorders often manifest themselves in the EEG as slow waves which unfortunately not only appear similar to OAs but share the same frequency bands as OAs. The problem then is to remove the OAs while preserving the signals of clinical interest.

10.6.2 Artefact processing algorithm

Several methods have been proposed for processing OAs. However, factors such as the requirements of the clinical laboratory, constraints of real-time applications, costs, the random nature of OAs and the spectral overlap between OAs and some signals of cerebral origin dictate that OA processing should be adaptive and in real time.

An adaptive ocular artefact filtering scheme is depicted in Figure 10.17. In this method estimates of OAs are obtained by suitably scaling the EOGs. The OA estimates are then subtracted from the contaminated EEGs to yield 'artefact-free' EEG signals. To illustrate this, consider the simple problem of correcting a single EEG channel for ocular artefact using four EOG signals (Figure 10.17(b)). The information contained in the contaminated EEG, y_k, and the EOGs, $x_k(0)$ to $x_k(3)$, is used to obtain

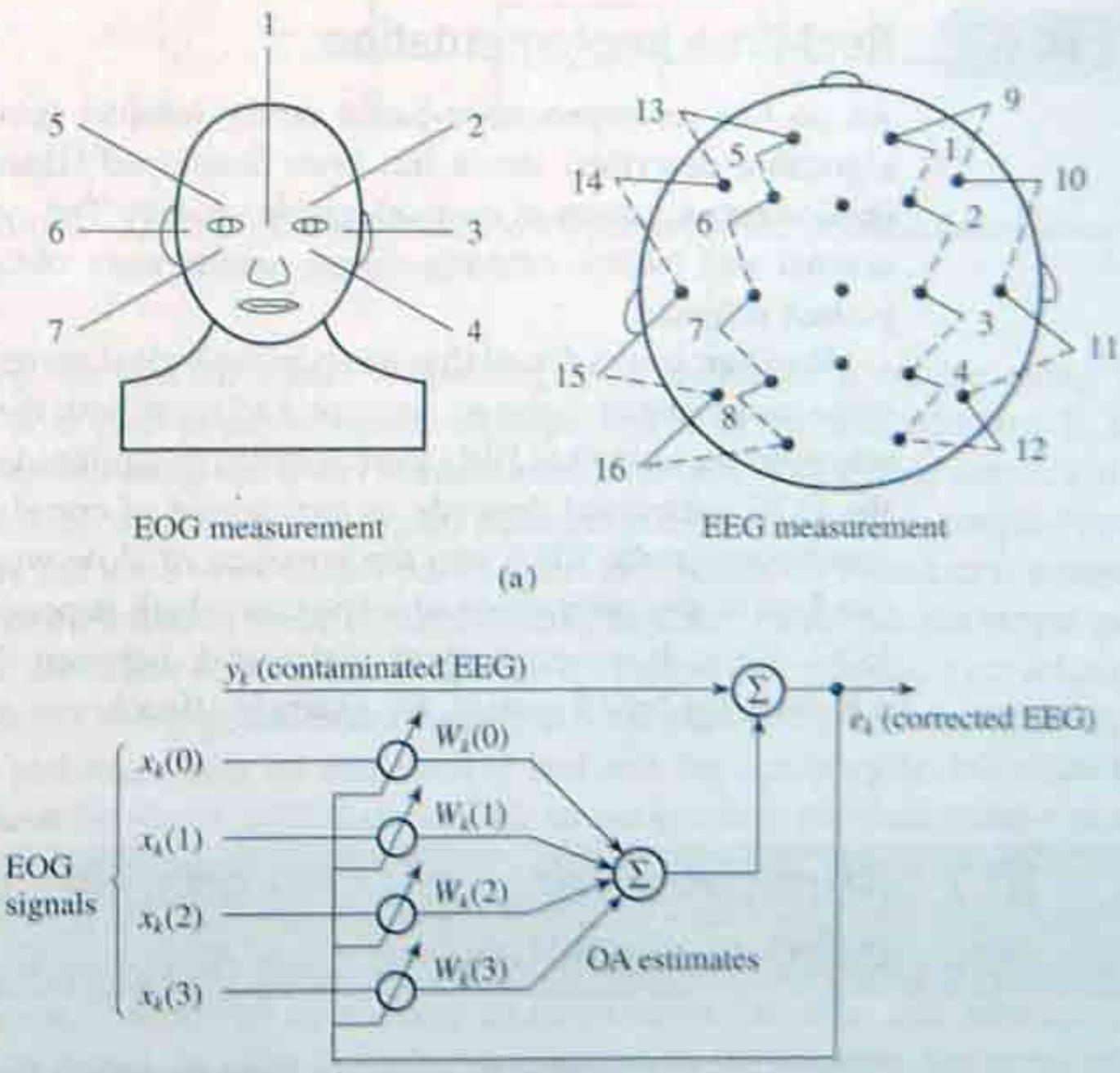


Figure 10.17 Adaptive ocular artefact removal method: (a) possible electrode positions for EOG (ocular movement) and EEG measurements; (b) adaptive ocular artefact filter.

an estimate of the ocular artefact, $\sum_{i=0}^{3} w_k(i)x_k(i)$. The OA estimate is then subtracted from the contaminated EEG to yield an 'artefact-free' EEG, e_k:

$$e_k = y_k - \sum_{i=0}^{n-1} w_k(i)x_k(i) \tag{10.37}$$

where $w_k(i)$, $i = 0, 1, \ldots, n-1$, are the coefficients of the adaptive filter and represent the fractions of the EOGs that reach the EEG as artefacts. e_k is also used to adjust the coefficients (weights) of the adaptive filter, using a numerically stable recursive least squares algorithm, so that optimal estimates of OAs are obtained. Continuous adjustment of $w_k(i)$ is necessary to account for changes in OAs due, for example, to changes in ocular movements.

The adaptive filtering algorithm used to remove the OAs is the UD algorithm described previously. This numerically stable formulation of the RLS algorithm was preferred to the LMS because of its superior convergence time, enabling it to cope better with different OAs each of which requires a different optimum set of coefficients for effective removal. An example of an EEG signal adaptively corrected for artefacts is shown in Figure 10.16(c).

10.6.3 Real-time implementation

An on-line microprocessor-based ocular artefact removal system that uses the UD algorithm described above has been developed (Ifeachor *et al.*, 1986). The system implements a variety of user-selectable models. The system has been tested on several normal and patient subjects. Good results were obtained for various categories of patient subjects.

However, it was found that when pathological waves, such as slow waves, epileptic spike and wave complexes, were picked up at both the EEG and EOG electrodes, the waves in the corrected EEG were reduced in amplitude. This is because the fraction of the EOG subtracted depends on the degree of correlation between the EOG and its component in the EEG, and the presence of slow waves of similar shape to the OA can lead to the subtraction of a fraction which depends on slow waves as well as the EOG. Thus, it is necessary to distinguish between the OA and slow waves, using a knowledge-based system, for example (Ifeachor *et al.*, 1990).

10.7 Application example 2 – adaptive telephone echo cancellation

Echoes arise primarily in communication systems when signals encounter a mismatch in impedance. Figure 10.18(a) shows a simplified long-distance telephone circuit. The hybrid circuit at the exchange converts the two-wire circuit from the customer's premises to a four-wire circuit, and provides separate paths for each direction of transmission. This is largely for economic reasons, for example to allow multiplexing, that is simultaneous transmission of many calls.

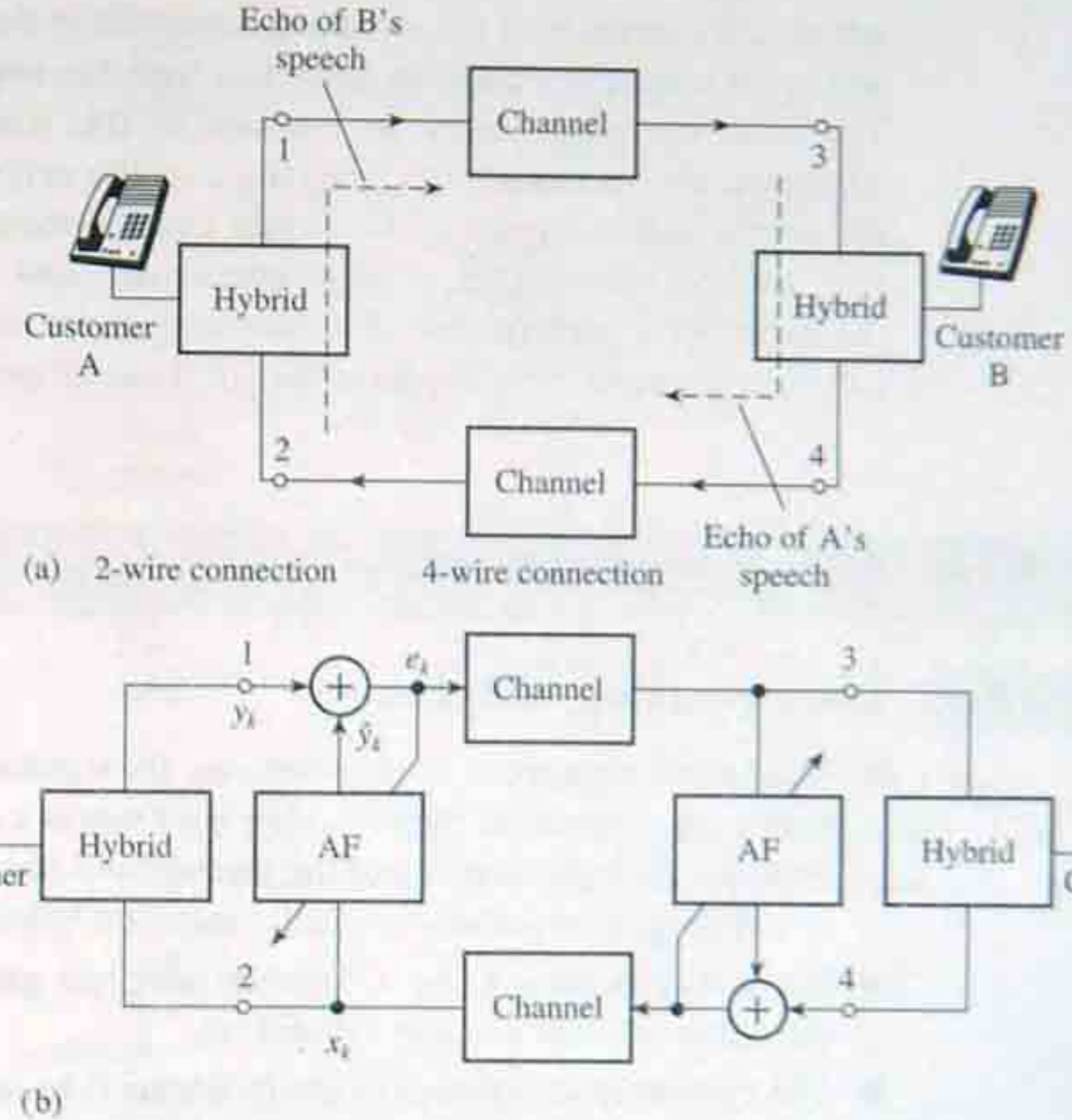


Figure 10.18 (a) Simplified long-distance telephone circuit; (b) echo cancellation in long-distance voice telephony.

Ideally, the speech signal originating from customer A travels along the upper transmission path to the hybrid on the right and from there to customer B, while that from B travels along the lower transmission path to A. The hybrid network at each end should ensure that the speech signal from the distant customer is coupled into its two-wire port and none to its output port. However, because of impedance mismatches the hybrid network allows some of the incoming signals to leak into the output path and to return to the talker as an echo. When the telephone call is made over a long distance (for example using geostationary satellites) the echo may be delayed by as much as 540 ms and represents an impairment that can be annoying to the customers. The impairment increases with distance. To overcome this problem, echo cancellers are installed in the network in pairs, as illustrated in Figure 10.18(b).

At each end of the communication system (Figure 10.18(b)), the incoming signal, x_k, is applied to both the hybrid and the adaptive filter (Duttweiller, 1978). The cancellation is achieved by making an estimate of the echo and subtracting it from the return signal, y_k. The underlying assumption here is that the echo return path (through the hybrid) is linear and time invariant. Thus the return signal at time k may be expressed as

$$y_k = \sum_{i=0}^{N-1} w_k(i) x_{k-i} + s_k \tag{10.38}$$

where x_k are samples of the incoming signal (from the far-end speaker), s_k is the near-end speaker plus any additive noise and w_k is the impulse response of the echo path. The echo canceller makes an estimate of this impulse response and produces a corresponding estimate, $\hat{y}_k = \sum w_k(i)x_{k-j}$, of the echo which is then subtracted from the normal return signal, y_k. Economic considerations place a limit on the sampling rate and the wordlengths of filter coefficients and input data which in turn limits the canceller's performance. Fundamental limits come from misadjustment in the adaptive filter and from nonlinearities in the echo path.

10.8 Other applications

10.8.1 Loudspeaking telephones

- The hybrid network is used to separate the transmit and receive paths (that is, the loudspeaker from the microphone), but there is a significant acoustic coupling between the loudspeaker and the microphone because of their proximity as well as a leakage across the imperfectly matched hybrid network (South *et al.*, 1979).
- The difficulty then is how to provide adequate gain for the receive and transmit directions without causing instability.
- The conventional solution to the problems is to use a voice-activated switch to select the transmit and receive paths, but this is not satisfactory because it does not allow full duplex communication.
- A better solution is to use adaptive filtering techniques to estimate and control the acoustic and hybrid echoes (Figure 10.19(b)). The number of filter coefficients here can be quite large, for example 512, making the use of a fast algorithm attractive.
- In teleconferencing networks (or public address systems) acoustic feedback leads to problems similar to those described above. Adaptive filters used for these may require large numbers of coefficients (250 to 1000), especially in rooms with long reverberation times, and must converge rapidly.

10.8.2 Multipath compensation

- In a type of spread spectrum system each data bit is transmitted as one of two orthogonal M-length pseudorandom sequences of bits. The sequence transmitted depends on whether the data bit is a logic 0 or 1. At the receiver two sequences identical to those used at the transmitter are cross-correlated with the received sequence to determine whether a 1 or 0 is received.
- In the presence of a multipath, the signal travels through separate paths to the receiver. Such effects could occur in mountainous or urban regions through reflection. The received signal is the sum of a number of components whose amplitudes and phases may differ (Figure 10.20). This reduces the performance of the receiver.

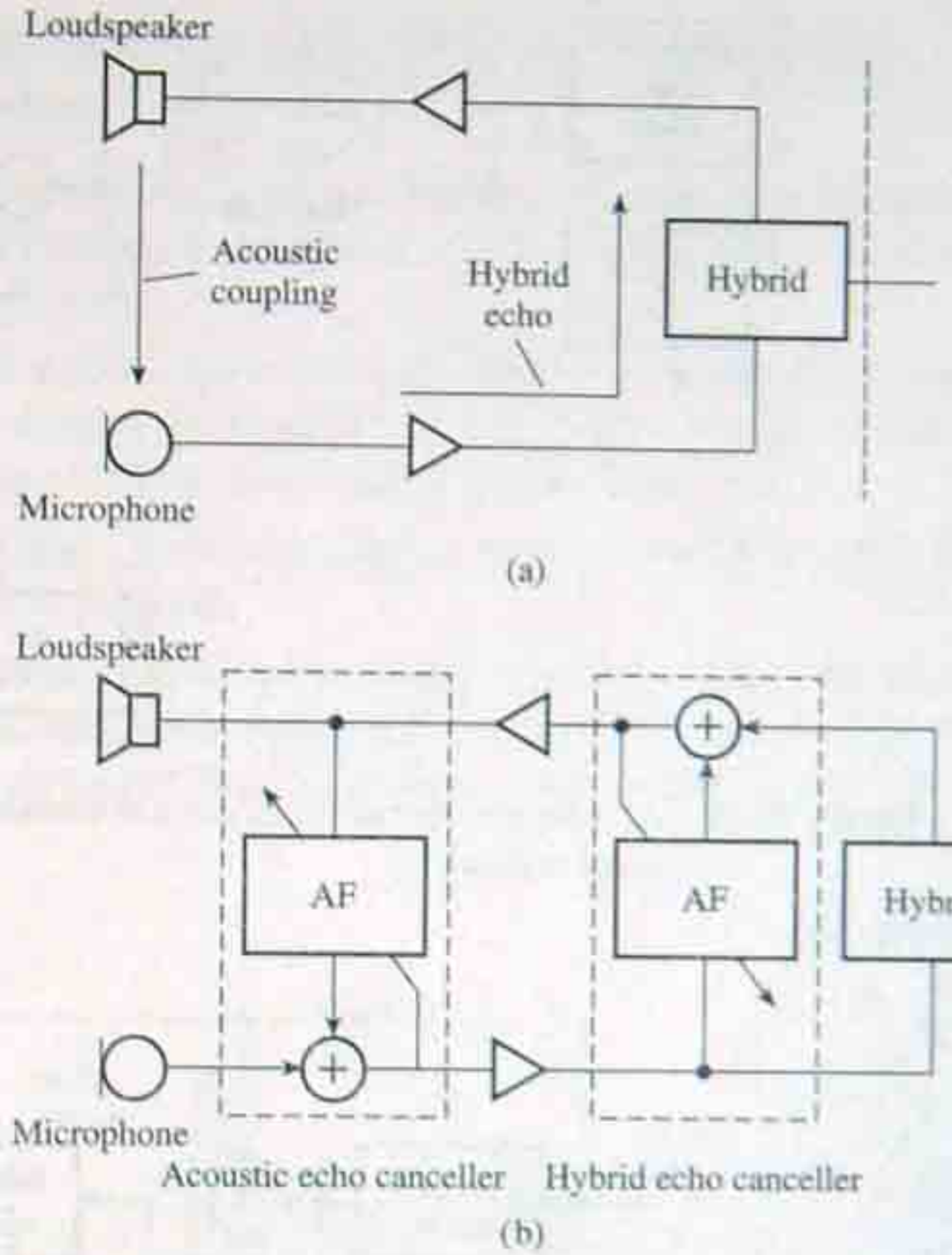


Figure 10.19 (a) Loudspeaking telephone; (b) acoustic and hybrid echo cancellation in loudspeaking telephone.

- The adaptive filter is used to estimate the overall multipath response and to compensate for its effects.

10.8.3 Adaptive jammer suppression

- In direct sequence spread spectrum a need often arises to suppress the effects of a jamming signal at the receiver to improve the performance of the receiver. Adaptive filtering may be used for this purpose (Figure 10.21). In such a system, use is made of the fact that the jammer is highly correlated whereas the pseudorandom code is weakly correlated. Thus the output of the filter, y_k, is an estimate of the jammer. This is subtracted from the received signal, x_k, to yield an estimate of the spread spectrum signal.
- To enhance the performance of the system a two-stage jammer suppressor is used. The adaptive line enhancer, which is essentially another adaptive filter, counteracts the effects of finite correlation which leads to partial cancellation of the desired signal. The number of coefficients required for either filter is moderate (about 16), but the sampling frequency may be well over 400 kHz.

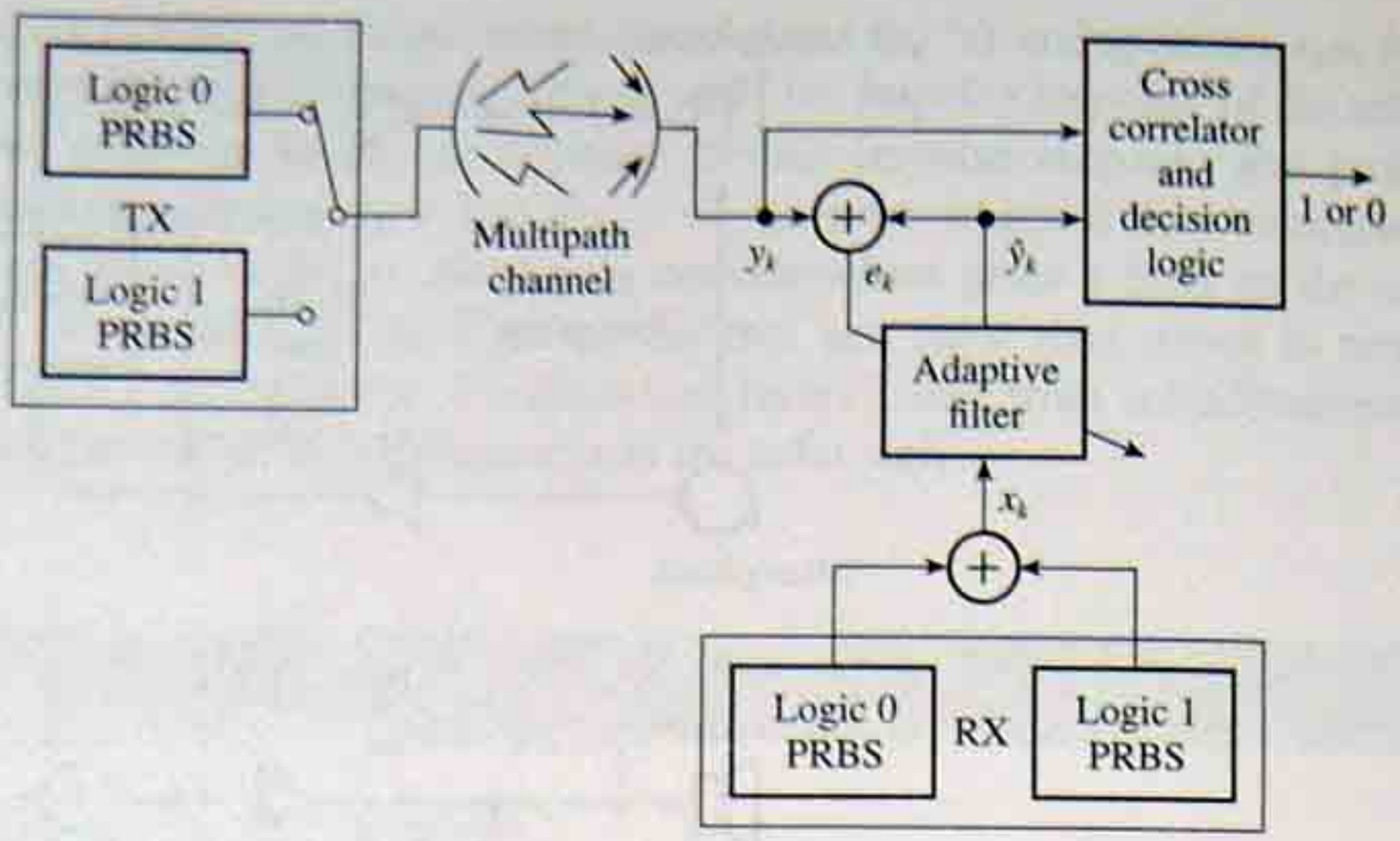


Figure 10.20 An adaptive spread spectrum communication system with multipath-effect compensation.

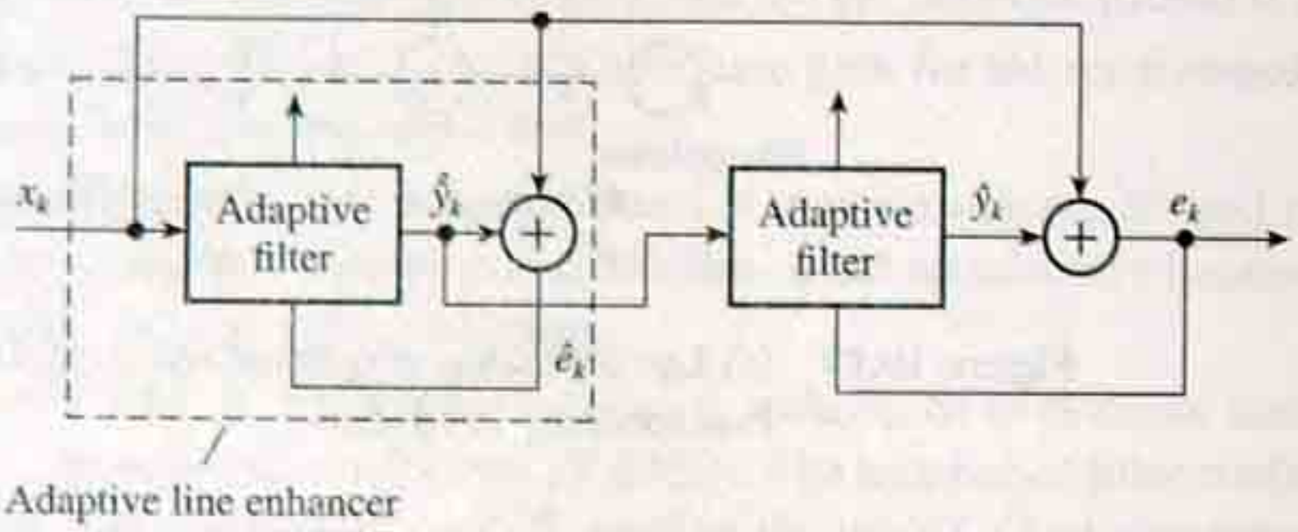


Figure 10.21 Jammer suppression in direct sequence spread spectrum receiver.

10.8.4 Radar signal processing

Adaptive signal processing techniques are widely used to solve a number of problems associated with radar. For example, adaptive filters are used in monostatic radar systems to remove or cancel clutter components from the desired target signals. In HF groundwave radar, adaptive filters are used to reduce co-channel interference which is a major problem in the HF band.

10.8.5 Separation of speech signals from background noise

Acoustic background noise is a serious problem in speech processing. An adaptive filter may be used to enhance the performance of speech systems in noisy environments (for example in fighter aircraft, tanks, or cars) to improve both intelligibility and recognition of speech.

10.8.6 Fetal monitoring – cancelling of maternal ECG during labour

- Information derived from the fetal electrocardiogram (ECG), such as the fetal heart rate pattern, is valuable in assessing the condition of the baby before or during childbirth.
- The ECG derived from electrodes placed on the mother's abdomen is susceptible to contamination from much larger background noise (for example muscle activity and fetal motion) and the mother's own ECG.
- Adaptive filters have been used to derive a 'noise-free' fetal ECG. Figure 10.22 illustrates the concept.
- Four chest leads are used to detect the baby's ECG, and one or more leads to detect the combined mother and baby's ECG. A four-channel adaptive filter, with 32 coefficients per channel, is used to cancel the mother's heartbeat as shown.

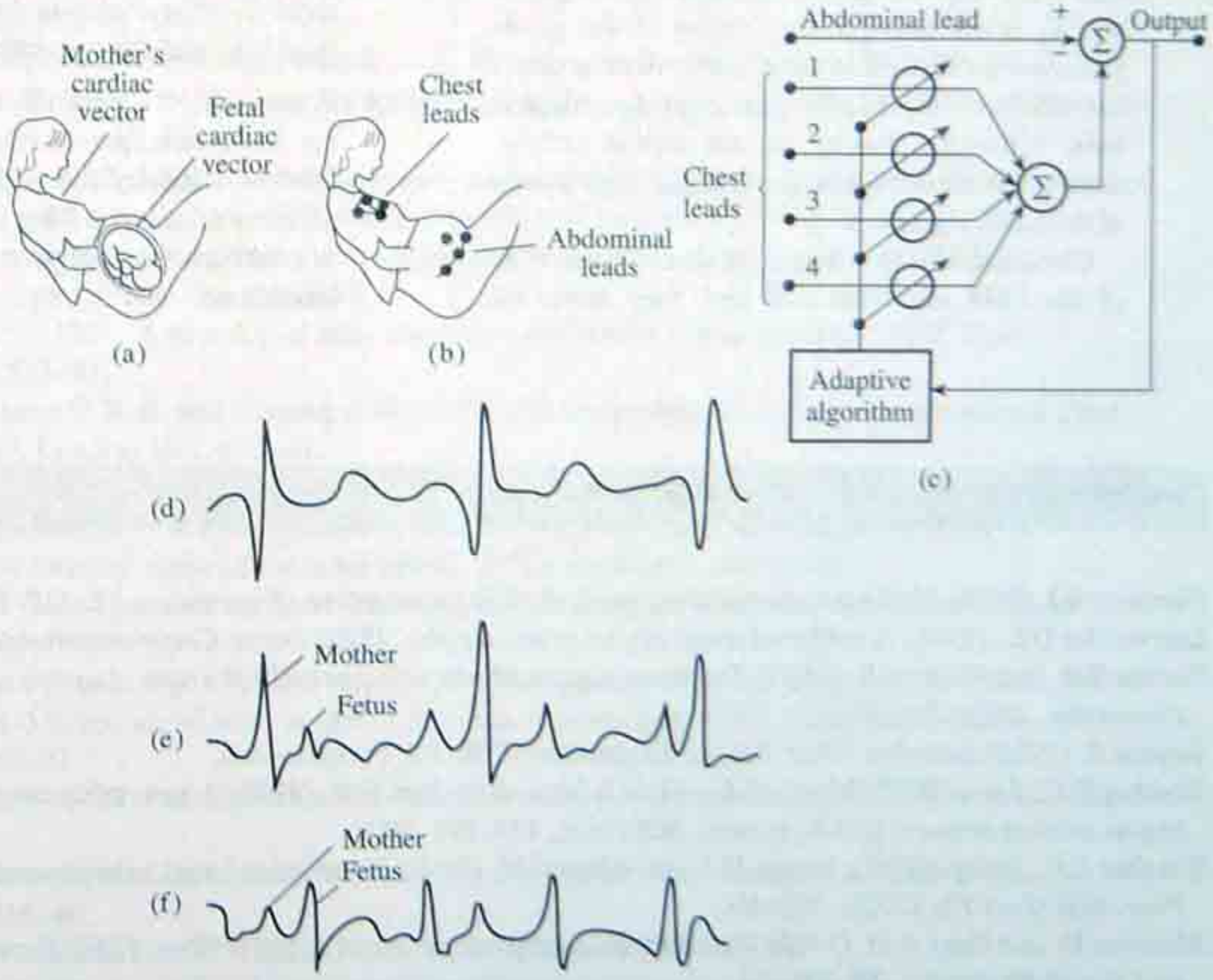


Figure 10.22 Adaptive cancelling of maternal ECG in fetal ECG (after Widrow *et al.*, 1975a): (a) cardiac electric field vectors of mother and fetus; (b) placement of leads; (c) adaptive; (d) idealized mother's ECG (chest leads); (e) idealized contaminated fetal ECG (abdominal lead); (f) output of noise canceller showing reduced mother's ECG.

Problems

10.1 Justify the use of adaptive filters instead of conventional filters in applications such as

(1) the removal of ocular artefacts from human EEGs,
(2) echo cancellation in long distance telephony, and
(3) suppression of jammer signal in spread spectrum communication.

Starting with the steepest descent algorithm:

$$\mathbf{W}_{k+1} = \mathbf{W}_k - \mu\nabla_k$$

where $\mathbf{W}_k$ is the filter weight vector at the discrete time k, μ controls stability and rate of convergence, and ∇_k is the true gradient vector of the error–performance surface at the discrete time k, derive the Widrow–Hopf LMS algorithm for adaptive noise cancelling, stating any reasonable assumptions made. Comment on the practical significance of the LMS algorithm.

Comment on two major practical limitations of the LMS algorithm and how they lower the performance of the algorithm. Suggest how these limitations may be overcome.

10.2 The output signal from an adaptive noise canceller is given by

$$e_k = y_k - \mathbf{X}_k^{\mathrm{T}}\mathbf{W}_k$$

where $\mathbf{W}_k$ is the adaptive filter weight vector and the other variables have the usual meaning. Starting with this equation, derive

(1) the discrete Wiener–Hopf equation and
(2) the basic LMS adaptive algorithm.

State any assumptions made.

10.3 Show that the adaptive filter turns itself off when there is no correlation between the interference signal, x_k, and the contaminated signal y_k.

10.4 Explain briefly, with the aid of a block diagram, the basic concepts of adaptive noise cancelling. Discuss critically the benefits and limitations of adaptive noise cancelling in a real-time application of your choice and suggest ways of overcoming the limitations.

References

Bierman G.J. (1976) Measurement updating using the UD factorization. *Automatica*, **12**, 375–82.

Duttweiller D.L. (1978) A twelve-channel digital echo canceler. *IEEE Trans. Communications*, **26**, 647–53.

Ferrara E.R. and Widrow B. (1981) The time-sequenced adaptive filter. *IEEE Trans. Acoustics, Speech and Signal Processing*, **29**(3), 766–70.

Haykin S. (1986) *Adaptive Filter Theory*. Englewood Cliffs NJ: Prentice-Hall.

Ifeachor E.C., Jervis B.W., Morris E.L., Allen E.M. and Hudson N.R. (1986) A new microcomputer-based on-line ocular artefact removal (OAR) system. *IEE Proc.*, **133**, 291–300.

Ifeachor E.C., Hellyar M.T., Mapps D.J. and Allen E.M. (1990) Knowledge based enhancement of EEG signals. *Proc. IEE (Part F)*, **137**(5), 302–10.

Mansour D. and Gray A.H. (1982) Unconstrained frequency domain adaptive filter. *IEEE Trans. Acoustics, Speech and Signal Processing*, **30**, 726–34.

Peterka P. (1975) A square root filter for real-time multivariate regression. *Kybernetika*, **11**, 53–67.

South C.R., Hoppitt C.E. and Lewis A.V. (1979) Adaptive filters to improve loudspeaker telephone. *Electronics Lett*, **15**, 673–4.

Widrow B. and Winter R. (1988) Neural nets for adaptive filtering and adaptive pattern recognition. *IEEE Computer*, 25–30.

Widrow B., Glover J.R., McCool J.M., Kaunitz J., Williams C.S., Hearn R.H., Zeidler J.R., Dong E. and Goodlin R.C. (1975a) Adaptive noise cancelling: principles and applications. *Proc. IEEE*, **63**, 1692–716.
Widrow B., McCool J.M. and Ball M. (1975b) The complex LMS algorithm. *Proc. IEEE*, 719–20.

Bibliography

Clark G.A., Mitra S.K. and Parker S.R. (1981) Block implementation of adaptive digital filters. *IEEE Trans. Acoustics, Speech and Signal Processing*, **29**, 744–52.
Clark G.A., Parker S.R. and Mitra S.K. (1983) A unified approach to time- and frequency-domain realization of FIR adaptive digital filters. *IEEE Trans. Acoustics, Speech and Signal Processing*, **31**, 1073–83.
Cowan C.F.N. and Grant P.M. (eds) (1985) *Adaptive Filters*. Englewood Cliffs NJ: Prentice-Hall.
De Courville M. and Duhamel P. (1995) Adaptive filtering in subbands using a weighted criterion. In *Proc. ICASSP*, Detroit, MI, Vol. 2, pp. 985–8.
Dentino M., McCool J.M. and Widrow B. (1978) Adaptive filtering in the frequency domain. *Proc. IEEE*, **66**, 1658–9.
Dudek M.T. and Robinson J.M. (1981) A new adaptive circuit for spectrally efficient digital microwave-radio-relay systems. *Electronics and Power*, 397–401.
Falconer D.D. (1982) Adaptive reference echo cancellation. *IEEE Trans. Communications*, **30**, 2083–94.
Ferrara E.R. and Widrow B. (1981) Multichannel adaptive filtering for signal enhancement. *IEEE Trans. Acoustics, Speech and Signal Processing*, **29**, 766–70.
Gilloire A. and Vetterli M. (1992) Adaptive filtering in subbands with critical sampling: analysis, experiments, and applications to acoustic echo cancellation. *IEEE Trans. Circuits and Systems*, **40**, 1862–75.
Harrison W.A., Lim J.S. and Singer E. (1986) A new application of adaptive noise cancellation. *IEEE Trans. Acoustics, Speech and Signal Processing*, **34**, 21–7.
Holte N. and Stueflotten S. (1981) A new digital echo canceler for two-wire subscriber lines. *IEEE Trans. Communications*, **29**, 1573–81.
Lappage R., Clarke J., Palma G.W.R. and Huizing A.G. (1987) The Byson research radar. In *International Conf. Radar 87*, October 1987, London: IEE, 453–61.
Levin M.D. and Cowan C.F.N. (1994) The performance of eight recursive least squares adaptive filtering algorithms in a limited precision environment. In *Proc. European Signal Processing Conf.*, Edinburgh, pp. 1261–4.
Lewis A. (1992) Adaptive filtering applications in telephony. *BT Technology J.*, **10**, 49–63.
Li Y. and Ding Z. (1995) Convergence analysis of finite length blind adaptive equalizers. *IEEE Trans. Signal Processing*, **43**, 2120–9.
Macchi O. (1995) *Adaptive Processing: The LMS Approach with Applications in Transmission*. New York: Wiley.
Messerschmitt D.G. (1984) Echo cancellation in speech and data transmission. *IEEE J. Selected Areas in Communications*, **2**, 283–97.
Mikhael W.B. and Wu F.H. (1987) Fast algorithms for block FIR adaptive digital filtering. *IEEE Trans. Circuits and Systems*, **34**, 1152–60.
Mueller K.H. (1976) A new digital echo canceler for two-wire full-duplex data transmission. *IEEE Trans. Communications*, **24**, 956–62.
Ochia K., Araseki T. and Ogihara T. (1977) *IEEE Trans. Communications*, **25**, 589–94.
Ogue J.C., Saito T. and Hoshiko Y. (1983) A fast convergence frequency domain adaptive filter. *IEEE Trans. Acoustics, Speech and Signal Processing*, **31**, 1312–14.
Reed F.A., Feintuch P.L. and Bershad N.J. (1985) The application of the frequency domain LMS adaptive filter to split array bearing estimation with a sinusoidal signal. *IEEE Trans. Acoustics, Speech and Signal Processing*, **33**, 61–9.
Saulnier G.J., Das P.K. and Milstein L. (1985) An adaptive digital suppression filter for direct sequence spread spectrum communications. *IEEE J. Selected Areas in Communications*, **3**, 676–86.

Sethares W.A., Lawrence D.A., Johnson C.R. and Bitmead R.R. (1986) Parameter drift in LMS adaptive filters. *IEEE Trans. Acoustics, Speech and Signal Processing*, **34**, 868–79.
Sondhi M.M. and Berkley D.A. (1980) Silencing echoes on the telephone network. *Proc. IEEE*, **68**, 948–63.
Tao Y.G., Kolwicz K.D., Gritton C.W.K. and Duttweiller D.L. (1986) A cascadable VLSI echo canceller. *IEEE Trans. Acoustics, Speech and Signal Processing*, **34**, 297–303.
Thornton C.L. and Bierman G.J. (1978) Filtering and error analysis via the UDU covariance factorization. *IEEE Trans. Automatic Control*, **23**, 901–7.
Widrow B. (1966) *Adaptive Filters 1: Fundamentals*. Report SU-SEL-66-126, Stanford Electronics Laboratory, Stanford University, CA.
Widrow B. (1971) Adaptive filters. In *Aspects of Network and System Theory* (Kalman R. and DeClaris N. (eds)), pp. 563–87. New York: Holt, Rinehart and Winston.
Widrow B. (1976) Stationary and nonstationary learning characteristics of the LMS adaptive filter. *Proc. IEEE*, **64**, 1151–62.
Widrow B. and Stearns S.D. (1985) *Adaptive Signal Processing*. Englewood Cliffs NJ: Prentice-Hall.
Widrow B., Mantey P., Griffiths L. and Goode B. (1967) Adaptive antenna systems. *Proc. IEEE*, **55**, 2143–59.

Appendices

10A C language programs for adaptive filtering

Several adaptive algorithms have been implemented in the C language which may be used to explore further the topics covered in this chapter. These are

(1) lmsflt.c, the LMS algorithm,
(2) uduflt.c, the UD algorithm,
(3) sqrflt.c, the square root algorithm and
(4) rlsflt.c, the recursive least squares algorithm.

Only the first program is listed here to limit the size of the book (see Program 10A.1). However, all the programs are available on the CD in the companion handbook *A Practical Guide for MATLAB and C Language Implementations of DSP Algorithms* (see the Preface for details).

Program 10A.1 C language implementation of the LMS algorithm (lmsflt.c).

```
/* ------------------------------------------------------------------------ */
/*      implementation of the LMS algorithm                                 */
/*                                                                          */
/*      manny 6.11.92                                                       */
/*                                                                          */
/*      inputs:                                                             */
/*      x[]     input data vector                                           */
/*      dk      latest input data value                                     */
/*      w[]     coefficient vector                                          */
/*                                                                          */
/*      outputs:                                                            */
/*      ek      error value                                                 */
/*      yk      digital filter output                                       */
/*      w[]     updated coefficient vector                                  */
/* ------------------------------------------------------------------------ */
```

```
double    lmsfit()
{
          int       i;
          double    uek,yk;

          yk = 0;
          for(i=0; i<N; ++i){                          /. digital filtering ./
                    yk=yk+w[i]*x[i];
          }
          ek=dk–yk;                                    /. compute output error ./
          uek=2*mu*ek;                                 /. update the weights ./
          for(i=0; i<N; i++){
                    w[i]=w[i]+uek*x[i];
          }
          return(yk);
}
```

To illustrate how to implement adaptive filters, we will use the program listed in Program 10A.1 to detect a tone in broadband noise.

10A.1 Adaptive enhancement of narrowband signals buried in noise

Adaptive filters are often used to detect or enhance narrowband signals buried in wideband noise. The structure that is commonly used for this purpose is depicted in Figure 10A.1. It consists of a delay element, symbolized by z^{-M}, and an adaptive predictor. The delay element removes any correlation that may exist between the samples of the noise component. The adaptive predictor is essentially an FIR filter with adjustable coefficients and its output, y_k, gives the enhanced narrowband signal. In some applications, the second output of the adaptive filter, e_k, and not y_k is the desired output. The prediction coefficients, $w_k(i)$, are optimized by a suitable adaptive algorithm, which in our case is an LMS algorithm (see Section 10.4 for details).

In the case of the LMS algorithm, the adaptive filter is characterized by the following equations:

$$y_k = \sum_{i=0}^{N-1} w_k(i)x_k(i) \tag{10A.1}$$

$$e_k = d_k - y_k \tag{10A.2}$$

$$w_{k+1}(i+1) = w_k(i) + 2\mu e_k x_k(i), \quad i = 0, 1, \ldots, N-1 \tag{10A.3}$$

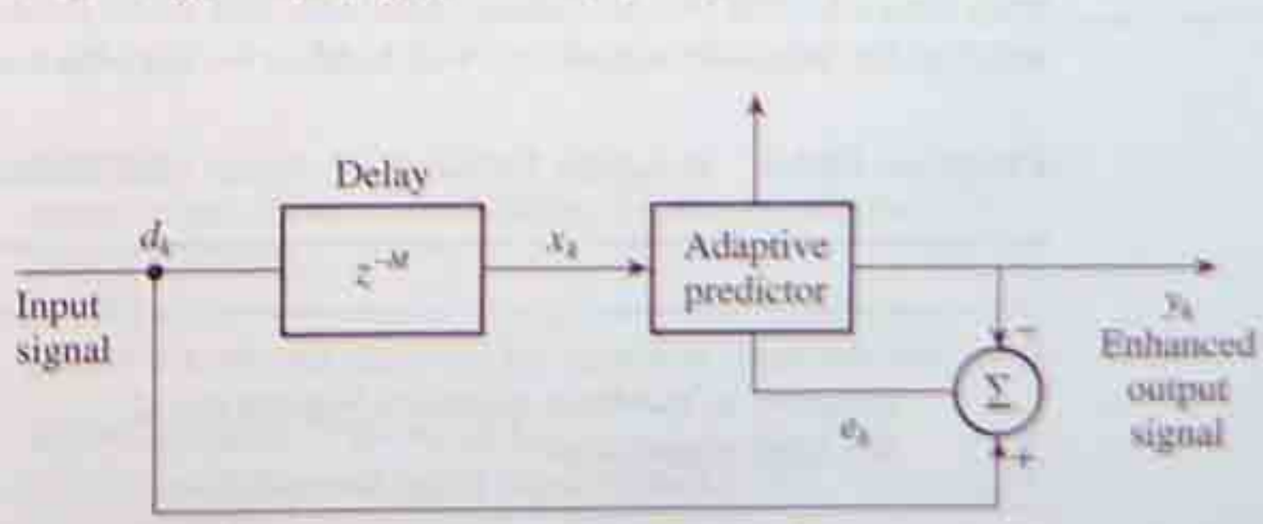


Figure 10A.1 Adaptive signal enhancement.

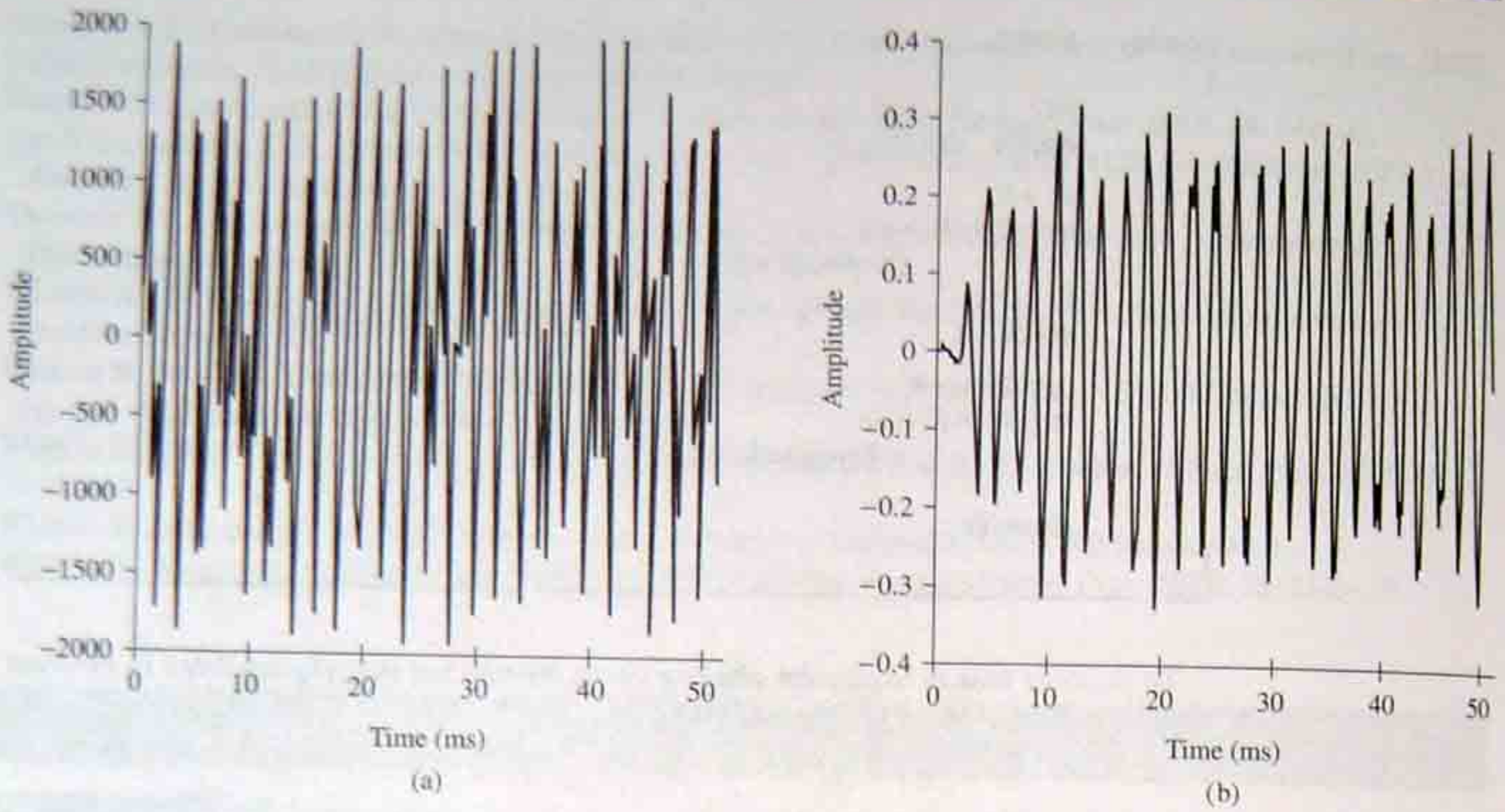


Figure 10A.2 Adaptive enhancement of a narrowband signal: (a) noisy signal; (b) enhanced signal.

where d_k is the noisy narrowband signal sample, $x_k(i)$ is the input data vector, derived from delayed values of d_k, $w_k(i)$ is the prediction coefficient vector at the kth sampling instant, μ is the stability factor, and y_k is the enhanced, narrowband signal.

The function, lmsflt.c, in Program 10A.1 is a C language implementation of the above equations. The program listed in Program 10A.2 illustrates how to use the function lmsflt.c for signal enhancement. To simulate the problem a broadband noise was added to a 500 Hz sine wave signal, and the composite data stored in ASCII format in a file din.dat. The noisy sine wave was then applied to the adaptive filter. To simulate real-time adaptive filtering, the input data is read from the file and applied to the adaptive filter one sample at a time. For very long lengths of data the user may need to read the data in blocks for efficiency. Figure 10A.2 shows the results for the LMS based filters.

As may be evident from Figure 10A.2, to use the adaptive algorithms, the user needs to specify the parameters of the adaptive filters, for example the length of the FIR filter, N, the delay factor, M, and the stability factor, μ. Attention must also be paid to the format of the input data. For example, in some applications the input data may come from a multichannel source, with each element of $x_k(i)$ representing the data value from a channel. In this case, the input data array to the adaptive algorithm will need to be suitably modified.

Program 10A.2 Program for adaptive signal enhancement.

```
/* ------------------------------------------------------------------------ */
/*                                                                          */
/*      program to illustrate adaptive filtering using                      */
/*      the LMS algorithms                                                  */
/*                                                                          */
/*      program name: adfilter.c                                            */
/*                                                                          */
```

```
/.      manny, 7.11.92                                                    ./
/. ------------------------------------------------------------------------ ./

#include    <stdio.h>
#include    <math.h>
#include    <dos.h>

/. constant definitions ./

#define N       30          /. filter length ./
#define M       1           /. delay ./
#define w0      0           /. initial value for adaptive filter coefficients ./
#define npt     N+M
#define SF      2048        /. factor for reducing the data samples – 11 bit ADC assumed ./
#define mu      0.04

double      lmsflt();
void        initlms();
void        update__data__buffers();
void        initfiles();
float       x[npt], d[npt], dk, ek;
double      w[npt];
FILE        .in,.out,.fopen();
char        din[30];

main()
{
            double yk, yk1;

            initfiles();
                                                    /. lms-based adaptive filter ./
            initlms();
            while(fscanf(in,"%f",&dk)!=EOF){
                dk=dk/SF;
                update__data__buffers();
                yk=lmsflt();
                yk1 =SF.yk;
                fprintf(out,"%lf \n",yk1);
            }
            fcloseall();
}

/. ------------------------------------------------------------------------ ./
void        initfiles()
{
            clrscr();
            printf("enter name of file holding data to be filtered \n");
            scanf("%s",din);
            printf("\n");
            printf("the filtered data will be stored in dout.dat \n");

            if((in=fopen(din,"r"))= =NULL){
                printf("cannot open input data file \n");
                exit(1);
            }
```

```
                if ((out = fopen ("dout.dat", "w"))= =NULL){
                        printf("cannot open output data file \n");
                        exit(1);
                }
                return;
}
/.----------------------------------------------------------------------------------------------./

void            update__data__buffers()
{
                long      j, k;

                for(j=1; j<N; + +j){                                    /.update x-data buffer./
                        k=N–j;
                        x[k]=x[k–1];
                }
                x[0]=dk;
                if(M>0)
                        x[0]=d[M–1];
                for(j=1; j<M; + +j){                                    /.update d-data buffer./
                        k=M–j;
                        d[k]=d[k–1];
                }
                d[0]=dk;
}

/.----------------------------------------------------------------------------------------------./
void            initlms()
{
                long      i;
                for(i=0; i<npt; + +i){
                        x[i]=0;
                        d[i]=0;
                        w[i] = w0;
                }
}
/.----------------------------------------------------------------------------------------------./

#include              "lmsflt.c";
```

10B MATLAB programs for adaptive filtering

MATLAB does not explicitly support adaptive signal processing. However, we have developed MATLAB programs for the two basic adaptive algorithms, namely, the LMS and the RLS algorithms:

lmsadf.m – function for performing LMS based adaptive filtering
rlsadf.m – function for performing RLS based adaptive filtering

The programs are available on the web. Illustrative examples of the use of the programs can be found in the companion manual, *A Practical Guide for MATLAB and C Language Implementations of DSP Algorithms*, published by Pearson Education (see the Preface for details).