CHAPTER

13 Recursive Least-Squares Algorithm

In this chapter we extend the use of the method of least squares to develop a recursive algorithm for the design of adaptive transversal filters such that, given the least-squares estimate of the tap-weight vector of the filter at iteration $n - 1$, we may compute the updated estimate of this vector at iteration n upon the arrival of new data. We refer to the resulting algorithm as the *recursive least-squares (RLS) algorithm.*

The RLS algorithm may be viewed as a special case of the Kalman filter. Indeed, this special relationship between the RLS algorithm and the Kalman filter is considered later in the chapter. Our main mission in this chapter, however, is to develop the basic theory of the RLS algorithm as an important tool for linear adaptive filtering in its own right.

We begin the development of the RLS algorithm by reviewing some basic relations that pertain to the method of least squares. Then, by exploiting a relation in matrix algebra known as the *matrix inversion lemma*, we develop the RLS algorithm. An important feature of the RLS algorithm is that it utilizes information contained in the input data, extending back to the instant of time when the algorithm is initiated. The resulting rate of convergence is therefore typically an order of magnitude faster than the simple LMS algorithm. This improvement in performance, however, is achieved at the expense of a large increase in computational complexity.

13.1 SOME PRELIMINARIES

In *recursive* implementations of the method of least squares, we start the computation with *known initial conditions* and use the information contained in new data samples to *update* the old estimates. We therefore find that the length of observable data is variable. Accordingly, we express the *cost function* to be minimized as $\mathscr{E}(n)$, where n is the variable length of the observable data. Also, it is customary to introduce a *weighting factor* into the definition of the cost function $\mathscr{E}(n)$. We thus write

$$\mathscr{E}(n) = \sum_{i=1}^{n} \beta(n, i)|e(i)|^2 \tag{13.1}$$

where $e(i)$ is the difference between the *desired response* $d(i)$ and the *output* $y(i)$ produced by a transversal filter whose tap inputs (at time i) equal $u(i), u(i-1), \ldots, u(i-M+1)$, as in Fig. 13.1. That is, $e(i)$ is defined by

$$\begin{aligned} e(i) &= d(i) - y(i) \\ &= d(i) - \mathbf{w}^H(n)\mathbf{u}(i) \end{aligned} \tag{13.2}$$

where $\mathbf{u}(i)$ is the *tap-input vector at time i*, defined by

$$\mathbf{u}(i) = [u(i), u(i-1), \ldots, u(i-M+1)]^T \tag{13.3}$$

and $\mathbf{w}(n)$ is the *tap-weight vector at time n*, defined by

$$\mathbf{w}(n) = [w_0(n), w_1(n), \ldots, w_{M-1}(n)]^T \tag{13.4}$$

Note that the tap weights of the transversal filter remain *fixed* during the observation interval $1 \le i \le n$ for which the cost function $\mathscr{E}(n)$ is defined.

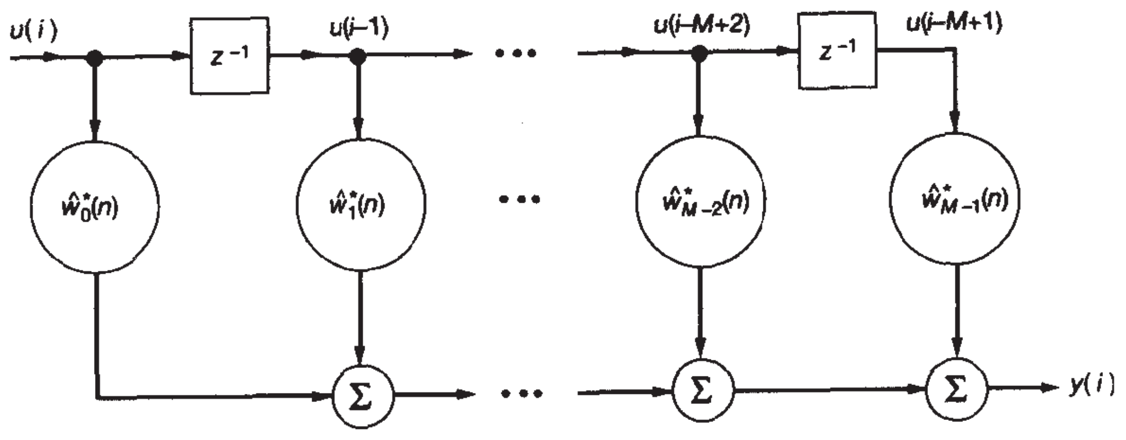

Figure 13.1 Transversal filter.

The weighting factor $\beta(n, i)$, in Eq. (13.1) has the property that

$$0 < \beta(n, i) \le 1, \qquad i = 1, 2, \ldots, n \tag{13.5}$$

The use of the weighting factor $\beta(n, i)$, in general, is intended to ensure that data in the distant past are "forgotten" in order to afford the possibility of following the statistical variations of the observable data when the filter operates in a nonstationary environment. A special form of weighting that is commonly used is the *exponential weighting factor* or *forgetting factor* defined by

$$\beta(n, i) = \lambda^{n-i}, \qquad i = 1, 2, \ldots, n \tag{13.6}$$

where λ is a positive constant close to, but less than, 1. When λ equals 1, we have the ordinary method of least squares. The inverse of $1 - \lambda$ is, roughly speaking, a measure of the *memory* of the algorithm. The special case $\lambda = 1$ corresponds to *infinite memory*. Thus, in the *method of exponentially weighted least squares*, we minimize the cost function

$$\mathscr{E}(n) = \sum_{i=1}^{n} \lambda^{n-i} |e(i)|^2 \tag{13.7}$$

The optimum value of the tap-weight vector, $\hat{\mathbf{w}}(n)$, for which the cost function $\mathscr{E}(n)$ of Eq. (13.7) attains its minimum value is defined by the *normal equations* written in matrix form:

$$\mathbf{\Phi}(n)\hat{\mathbf{w}}(n) = \mathbf{z}(n) \tag{13.8}$$

The M-by-M correlation matrix $\mathbf{\Phi}(n)$ is now defined by

$$\mathbf{\Phi}(n) = \sum_{i=1}^{n} \lambda^{n-i} \mathbf{u}(i)\mathbf{u}^H(i) \tag{13.9}$$

The M-by-1 cross-correlation vector $\mathbf{z}(n)$ between the tap inputs of the transversal filter and the desired response is correspondingly defined by

$$\mathbf{z}(n) = \sum_{i=1}^{n} \lambda^{n-i} \mathbf{u}(i) d^*(i) \tag{13.10}$$

where the asterisk denotes complex conjugation.

The correlation matrix $\mathbf{\Phi}(n)$ of Eq. (13.9) differs from the time-averaged version of Eq. (11.45) in two respects.

1. The matrix product $\mathbf{u}(i)\mathbf{u}^H(i)$ inside the summation on the right-hand side of Eq. (11.45) is weighted by the exponential factor λ^{n-i}, which arises naturally from the adoption of Eq. (13.7) as the cost function.
2. The use of *prewindowing* is assumed, according to which the input data prior to time $i = 1$ are equal to zero, hence the use of $i = 1$ as the lower limit of the summation.

Similar remarks apply to the cross-correlation vector $\mathbf{z}(n)$ compared to its time-averaged counterpart of Chapter 11.

Isolating the term corresponding to $i = n$ from the rest of the summation on the right-hand side of Eq. (13.9), we may write

$$\mathbf{\Phi}(n) = \lambda\left[\sum_{i=1}^{n-1} \lambda^{n-1-i}\,\mathbf{u}(i)\mathbf{u}^H(i)\right] + \mathbf{u}(n)\mathbf{u}^H(n) \tag{13.11}$$

However, by definition, the expression inside the square brackets on the right-hand side of Eq. (13.11) equals the correlation matrix $\mathbf{\Phi}(n-1)$. Hence, we have the following recursion for updating the value of the correlation matrix of the tap inputs:

$$\mathbf{\Phi}(n) = \lambda\mathbf{\Phi}(n-1) + \mathbf{u}(n)\mathbf{u}^H(n) \tag{13.12}$$

where $\mathbf{\Phi}(n-1)$ is the "old" value of the correlation matrix, and the matrix product $\mathbf{u}(n)\mathbf{u}^H(n)$ plays the role of a "correction" term in the updating operation.

Similarly, we may use Eq. (13.10) to derive the following recursion for updating the cross-correlation vector between the tap inputs and the desired response:

$$\mathbf{z}(n) = \lambda\mathbf{z}(n-1) + \mathbf{u}(n)d^*(n) \tag{13.13}$$

To compute the least-square estimate $\hat{\mathbf{w}}(n)$ for the tap-weight vector in accordance with Eq. (13.8), we have to determine the inverse of the correlation matrix $\mathbf{\Phi}(n)$. In practice, however, we usually try to avoid performing such an operation as it can be very time consuming, particularly if the number of tap weights, M, is high. Also, we would like to be able to compute the least-squares estimate $\hat{\mathbf{w}}(n)$ for the tap-weight vector recursively for $n = 1, 2, \ldots, \infty$. We can realize both of these objectives by using a basic result in matrix algebra known as the *matrix inversion lemma*. We assume that the initial conditions have been chosen to ensure the nonsingularity of the correlation matrix $\mathbf{\Phi}(n)$; this issue is discussed later in Section 13.3.

13.2 THE MATRIX INVERSION LEMMA

Let $\mathbf{A}$ and $\mathbf{B}$ be two positive-definite M-by-M matrices related by

$$\mathbf{A} = \mathbf{B}^{-1} + \mathbf{C}\mathbf{D}^{-1}\mathbf{C}^H \tag{13.14}$$

where $\mathbf{D}$ is another positive-definite N-by-M matrix, and $\mathbf{C}$ is an M-by-N matrix. According to the *matrix inversion lemma*, we may express the inverse of the matrix $\mathbf{A}$ as follows:

$$\mathbf{A}^{-1} = \mathbf{B} - \mathbf{B}\mathbf{C}(\mathbf{D} + \mathbf{C}^H\mathbf{B}\mathbf{C})^{-1}\mathbf{C}^H\mathbf{B} \tag{13.15}$$

The proof of this lemma is established by multiplying Eq. (13.14) by (13.15) and recognizing that the product of a square matrix and its inverse is equal to the identity matrix (see Problem 2). The matrix inversion lemma states that if we are given a matrix $\mathbf{A}$ as defined in Eq. (13.14), we can determine its inverse $\mathbf{A}^{-1}$ by using the relation of Eq. (13.15). In effect, the lemma is described by this pair of equations. The matrix inversion lemma is also referred to in the literature as *Woodbury's identity*.[1]

In the next section we show how the matrix inversion lemma can be applied to obtain a recursive equation for computing the least-squares solution $\hat{\mathbf{w}}(n)$ for the tap-weight vector.

13.3 THE EXPONENTIALLY WEIGHTED RECURSIVE LEAST-SQUARES ALGORITHM

With the correlation matrix $\mathbf{\Phi}(n)$ assumed to be positive definite and therefore nonsingular, we may apply the matrix inversion lemma to the recursive equation (13.12). We first make the following identifications:

$$\begin{aligned} \mathbf{A} &= \mathbf{\Phi}(n) \\ \mathbf{B}^{-1} &= \lambda\mathbf{\Phi}(n-1) \\ \mathbf{C} &= \mathbf{u}(n) \\ \mathbf{D} &= 1 \end{aligned}$$

Then, substituting these definitions in the matrix inversion lemma of Eq. (13.15), we obtain the following recursive equation for the inverse of the correlation matrix:

$$\mathbf{\Phi}^{-1}(n) = \lambda^{-1}\mathbf{\Phi}^{-1}(n-1) - \frac{\lambda^{-2}\mathbf{\Phi}^{-1}(n-1)\mathbf{u}(n)\mathbf{u}^H(n)\mathbf{\Phi}^{-1}(n-1)}{1 + \lambda^{-1}\mathbf{u}^H(n)\mathbf{\Phi}^{-1}(n-1)\mathbf{u}(n)} \tag{13.16}$$

For convenience of computation, let

$$\mathbf{P}(n) = \mathbf{\Phi}^{-1}(n) \tag{13.17}$$

and

$$\mathbf{k}(n) = \frac{\lambda^{-1}\mathbf{P}(n-1)\mathbf{u}(n)}{1 + \lambda^{-1}\mathbf{u}^H(n)\mathbf{P}(n-1)\mathbf{u}(n)} \tag{13.18}$$

Using these definitions, we may rewrite Eq. (13.16) as follows:

$$\mathbf{P}(n) = \lambda^{-1}\mathbf{P}(n-1) - \lambda^{-1}\mathbf{k}(n)\mathbf{u}^H(n)\mathbf{P}(n-1) \tag{13.19}$$

[1]The exact origin of the matrix inversion lemma is not known. Householder (1964) attributes it to Woodbury (1950). Nevertheless, application of the matrix inversion lemma in the filtering literature was first made by Kailath, who used a form of this lemma to prove the equivalence of the Wiener filter and the maximum-likelihood procedure for estimating the output of a random linear time-invariant channel that is corrupted by additive white Gaussian noise (Kailath, 1960). Early use of the matrix inversion lemma was also made by Ho (1963). Another interesting application of the matrix inversion lemma was made by Brooks and Reed, who used it to prove the equivalence of the Wiener filter, the maximum signal-to-noise ratio filter, and the likelihood ratio processor for detecting a signal in additive white Gaussian noise (Brooks and Reed, 1972).

The M-by-M matrix $\mathbf{P}(n)$ is referred to as the *inverse correlation matrix*. The M-by-1 vector $\mathbf{k}(n)$ is referred to as the *gain vector* for reasons that will become apparent later in the section. Equation (13.19) is the *Riccati equation* for the RLS algorithm.

By rearranging Eq. (13.18), we have

$$\begin{aligned}\mathbf{k}(n) &= \lambda^{-1}\mathbf{P}(n-1)\mathbf{u}(n) - \lambda^{-1}\mathbf{k}(n)\mathbf{u}^H(n)\mathbf{P}(n-1)\mathbf{u}(n)\\ &= [\lambda^{-1}\mathbf{P}(n-1) - \lambda^{-1}\mathbf{k}(n)\mathbf{u}^H(n)\mathbf{P}(n-1)]\mathbf{u}(n)\end{aligned} \tag{13.20}$$

We see from Eq. (13.19) that the expression inside the brackets on the right-hand side of Eq. (13.20) equals $\mathbf{P}(n)$. Hence. we may simplify Eq. (13.20) to

$$\mathbf{k}(n) = \mathbf{P}(n)\mathbf{u}(n) \tag{13.21}$$

This result, together with $\mathbf{P}(n) = \mathbf{\Phi}^{-1}(n)$, may be used as the definition for the gain vector:

$$\mathbf{k}(n) = \mathbf{\Phi}^{-1}(n)\mathbf{u}(n) \tag{13.22}$$

In other words, the gain vector $\mathbf{k}(n)$ is defined as the tap-input vector $\mathbf{u}(n)$ transformed by the inverse of the correlation matrix $\mathbf{\Phi}(n)$.

Time Update for the Tap-Weight Vector

Next, we wish to develop a recursive equation for updating the least-squares estimate $\hat{\mathbf{w}}(n)$ for the tap-weight vector. To do this, we use Eqs. (13.8), (13.13), and (13.17) to express the least-squares estimate $\hat{\mathbf{w}}(n)$ for the tap-weight vector at iteration n as follows:

$$\begin{aligned}\hat{\mathbf{w}}(n) &= \mathbf{\Phi}^{-1}(n)\mathbf{z}(n)\\ &= \mathbf{P}(n)\mathbf{z}(n)\\ &= \lambda\mathbf{P}(n)\mathbf{z}(n-1) + \mathbf{P}(n)\mathbf{u}(n)d^*(n)\end{aligned} \tag{13.23}$$

Substituting Eq. (13.19) for $\mathbf{P}(n)$ in the first term only in the right-hand side of Eq. (13.23), we get

$$\begin{aligned}\hat{\mathbf{w}}(n) &= \mathbf{P}(n-1)\mathbf{z}(n-1) - \mathbf{k}(n)\mathbf{u}^H(n)\mathbf{P}(n-1)\mathbf{z}(n-1)\\ &\quad + \mathbf{P}(n)\mathbf{u}(n)d^*(n)\\ &= \mathbf{\Phi}^{-1}(n-1)\mathbf{z}(n-1) - \mathbf{k}(n)\mathbf{u}^H(n)\mathbf{\Phi}^{-1}(n-1)\mathbf{z}(n-1)\\ &\quad + \mathbf{P}(n)\mathbf{u}(n)d^*(n)\\ &= \hat{\mathbf{w}}(n-1) - \mathbf{k}(n)\mathbf{u}^H(n)\hat{\mathbf{w}}(n-1) + \mathbf{P}(n)\mathbf{u}(n)d^*(n)\end{aligned} \tag{13.24}$$

Finally, using the fact that $\mathbf{P}(n)\mathbf{u}(n)$ equals the gain vector $\mathbf{k}(n)$, as in Eq. (13.21), we get the desired recursive equation for updating the tap-weight vector:

$$\begin{aligned}\hat{\mathbf{w}}(n) &= \hat{\mathbf{w}}(n-1) + \mathbf{k}(n)[d^*(n) - \mathbf{u}^H(n)\hat{\mathbf{w}}(n-1)]\\ &= \hat{\mathbf{w}}(n-1) + \mathbf{k}(n)\xi^*(n)\end{aligned} \tag{13.25}$$

where $\xi(n)$ is the *a priori estimation error* defined by

$$\begin{aligned}\xi(n) &= d(n) - \mathbf{u}^T(n)\hat{\mathbf{w}}^*(n-1) \\ &= d(n) - \hat{\mathbf{w}}^H(n-1)\mathbf{u}(n)\end{aligned} \tag{13.26}$$

The inner product $\hat{\mathbf{w}}^H(n - 1)\mathbf{u}(n)$ represents an estimate of the desired response $d(n)$, based on the *old* least-squares estimate of the tap-weight vector that was made at time $n - 1$.

Equation (13.25) for the adjustment of the tap-weight vector and Eq. (13.26) for the *a priori* estimation error suggest the block-diagram representation depicted in Fig. 13.2(a) for the *recursive least-squares RLS algorithm.*

The *a priori* estimation error $\xi(n)$ is, in general, different from the *a posteriori estimation error*

$$e(n) = d(n) - \hat{\mathbf{w}}^H(n)\mathbf{u}(n) \tag{13.27}$$

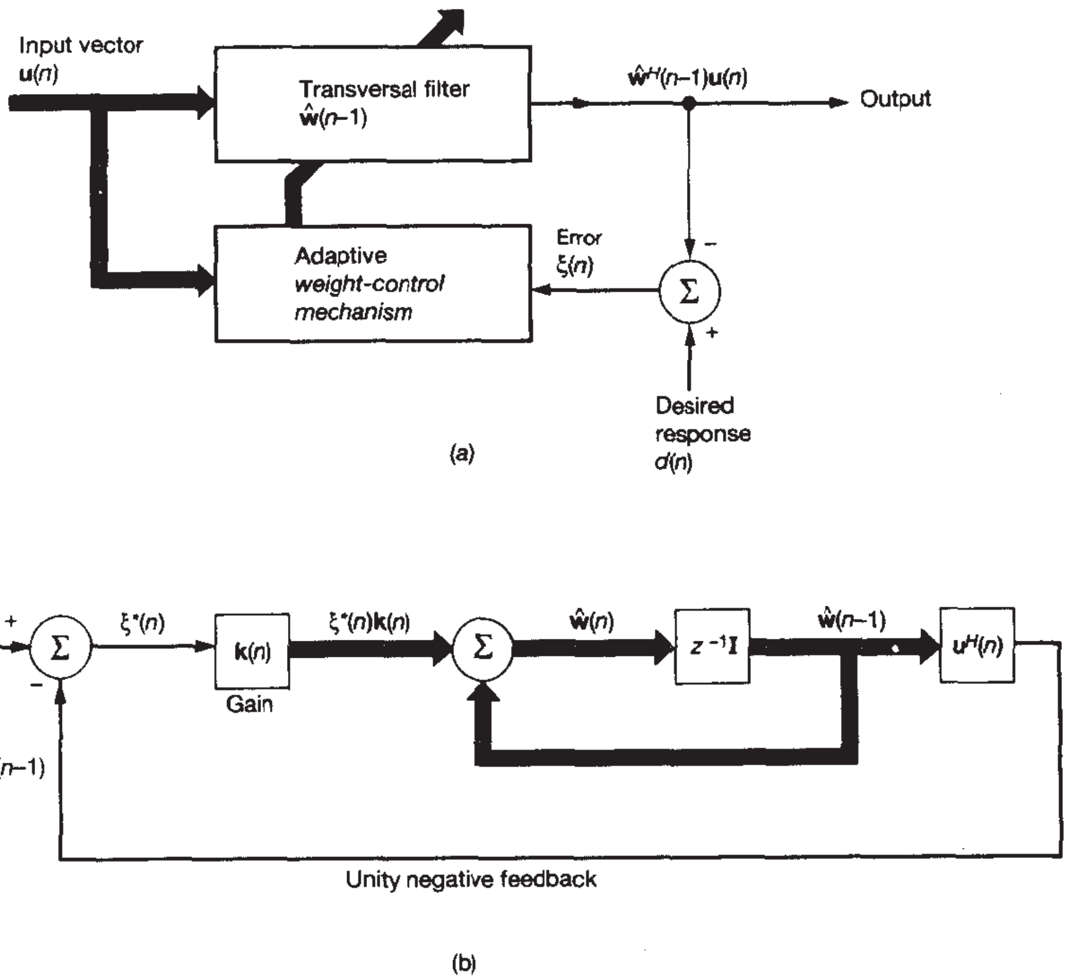

Figure 13.2 Representations of the RLS algorithm: (1) block diagram; (b) signal-flow graph.

TABLE 13.1 SUMMARY OF THE RLS ALGORITHM

Initialize the algorithm by setting

$\mathbf{P}(0) = \delta^{-1}\mathbf{I}, \qquad \delta = \text{small positive constant}$

$\hat{\mathbf{w}}(0) = \mathbf{0}$

For each instant of time, $n = 1, 2, \ldots$, compute

$$\mathbf{k}(n) = \frac{\lambda^{-1}\mathbf{P}(n-1)\mathbf{u}(n)}{1 + \lambda^{-1}\mathbf{u}^H(n)\mathbf{P}(n-1)\mathbf{u}(n)}$$

$$\xi(n) = d(n) - \hat{\mathbf{w}}^H(n-1)\mathbf{u}(n)$$

$$\hat{\mathbf{w}}(n) = \hat{\mathbf{w}}(n-1) + \mathbf{k}(n)\xi^*(n)$$

$$\mathbf{P}(n) = \lambda^{-1}\mathbf{P}(n-1) - \lambda^{-1}\mathbf{k}(n)\mathbf{u}^H(n)\mathbf{P}(n-1)$$

the computation of which involves the *current* least-squares estimate of the tap-weight vector available at time n. Indeed, we may view $\xi(n)$ as a "tentative" value of $e(n)$ before updating the tap-weight vector. Note, however, in the least-squares optimization that led to the recursive algorithm of Eq. (13.25) for the tap-weight vector, we actually minimized a cost function based on $e(n)$ and *not* $\xi(n)$.

Summary of the RLS Algorithm

Equations (13.18), (13.26), (13.25), and (13.19), collectively and in that order, constitute the *RLS algorithm*, as summarized in Table 13.1. We note that, in particular, Eq. (13.26) describes the filtering operation of the algorithm, whereby the transversal filter is excited to compute the *a priori* estimation error $\xi(n)$. Equation (13.25) describes the adaptive operation of the algorithm, whereby the tap-weight vector is updated by incrementing its old value by an amount equal to the complex conjugate of the *a priori* estimation error $\xi(n)$ times the time-varying gain vector $\mathbf{k}(n)$, hence the name "gain vector." Equations (13.18) and (13.19) enable us to update the value of the gain vector itself. An important feature of the RLS algorithm described by these equations is that the inversion of the correlation matrix $\mathbf{\Phi}(n)$ is replaced at each step by a simple scalar division. Figure 13.2(b) depicts a signal-flow-graph representation of the RLS algorithm that complements the block diagram of Fig. 13.2(a).

Initialization of the RLS Algorithm

The applicability of the RLS algorithm requires that we initialize the recursion of Eq. (13.19) by choosing a starting value $\mathbf{P}(0)$ that assures the nonsingularity of the correlation matrix $\mathbf{\Phi}(n)$. We may do this by evaluating the inverse

$$\left[\sum_{i=-n_0}^{0} \lambda^{-i}\mathbf{u}(i)\mathbf{u}^H(i)\right]^{-1}$$

where the tap-weight vector $\mathbf{u}(i)$ is obtained from an initial block of data for $-n_0 \le i \le 0$.

A simpler procedure, however, is to modify the expression slightly for the correlation matrix $\mathbf{\Phi}(n)$ by writing

$$\mathbf{\Phi}(n) = \sum_{i=1}^{n} \lambda^{n-i}\mathbf{u}(i)\mathbf{u}^H(i) + \delta\lambda^n\mathbf{I} \tag{13.28}$$

where $\mathbf{I}$ is the M-by-M identity matrix, and δ is a small positive constant. Thus putting $n = 0$ in Eq. (13.28), we have

$$\mathbf{\Phi}(0) = \delta\mathbf{I}$$

Correspondingly, for the initial value of $\mathbf{P}(n)$ equal to the inverse of the correlation matrix $\mathbf{\Phi}(n)$, we set

$$\mathbf{P}(0) = \delta^{-1}\mathbf{I} \tag{13.29}$$

The initialization described in Eq. (13.29) is equivalent to forcing the unknown data sample $u(-M+1)$ equal to the value $\lambda^{(-M+1)/2}\delta^{1/2}$ instead of zero. In other words, during the initialization period we modify the prewindowing method by writing

$$u(n) = \begin{cases} \lambda^{(-M+1)/2}\delta^{1/2}, & n = -M+1 \\ 0, & n < 0, n \ne -M+1 \end{cases} \tag{13.30}$$

Note that for a transversal filter with M taps, the index $n = M + 1$ refers to the *last* tap in the filter. When the first nonzero data sample $u(i)$ enters the filter, the initializing tap input $u(-M+1)$ leaves the filter and from then on the RLS algorithm takes over.

It only remains for us to choose an initial value for the tap-weight vector. It is customary to set

$$\hat{\mathbf{w}}(0) = \mathbf{0} \tag{13.31}$$

where $\mathbf{0}$ is the M-by-1 null vector.

The initialization procedure incorporating Eqs. (13.29) and (13.31) is referred to as a *soft-constrained initialization*. The positive constant δ is the only parameter required for this initialization. The recommended choice of δ is that it should be small compared to $0.01\sigma_u^2$, where σ_u^2 is the variance of a data sample $u(n)$. Such a choice is based on practical experience with the RLS algorithm, supported by a statistical analysis of the soft-constrained initialization of the algorithm (Hubing and Alexander, 1990). For large data lengths, the exact value of the initializing constant δ has an insignificant effect.

It is important to note that by using the initialization procedure defined by Eqs. (13.29) and (13.31), we are no longer computing a solution that minimizes the cost function $\xi(n)$ of Eq. (13.7) exactly. Instead, we are computing the solution that minimizes the modified cost function:

$$\mathscr{E}(n) = \delta\lambda^n\|\mathbf{w}(n)\|^2 + \sum_{i=1}^{n} \lambda^{n-i}\,|e(i)|^2$$

In other words, the RLS algorithm summarized in Table 13.1 yields the exact recursive solution to the following optimization problem (Sayed and Kailath, 1994):

$$\min_{\mathbf{w}(n)} [\delta\lambda^n \|\mathbf{w}(n)\|^2 + \sum_{i=1}^{n} \lambda^{n-i}|e(i)|^2]$$

where $e(i)$ is defined by Eq. (13.2).

13.4 UPDATE RECURSION FOR THE SUM OF WEIGHTED ERROR SQUARES

The minimum value of the sum of weighted error squares, $\mathscr{E}_{\min}(n)$, results when the tap-weight vector is set equal to the least-squares estimate $\hat{\mathbf{w}}(n)$. To compute $\mathscr{E}_{\min}(n)$, we may therefore use the relation [see first line of Eq. (10.40)]:

$$\mathscr{E}_{\min}(n) = \mathscr{E}_d(n) - \mathbf{z}^H(n)\hat{\mathbf{w}}(n) \tag{13.32}$$

where $\mathscr{E}_d(n)$ is defined by (using the notation of this chapter)

$$\begin{aligned} \mathscr{E}_d(n) &= \sum_{i=1}^{n} \lambda^{n-i}|d(i)|^2 \\ &= \lambda\, \mathscr{E}_d(n-1) + |d(n)|^2 \end{aligned} \tag{13.33}$$

Therefore, substituting Eqs. (13.13), (13.25), and (13.33) in (13.32), we get

$$\begin{aligned} \mathscr{E}_{\min}(n) = \lambda[&\mathscr{E}_d(n-1) - \mathbf{z}^H(n-1)\hat{\mathbf{w}}(n-1)] \\ &+ d(n)[d^*(n) - \mathbf{u}^H(n)\hat{\mathbf{w}}(n-1)] \\ &- \mathbf{z}^H(n)\mathbf{k}(n)\xi^*(n) \end{aligned} \tag{13.34}$$

where, in the last term, we have restored $\mathbf{z}(n)$ to its original form. By definition, the expression inside the first set of brackets on the right-hand side of Eq. (13.34) equals $\mathscr{E}_{\min}(n-1)$. Also, by definition, the expression inside the second set of brackets equals the complex conjugate of the *a priori* estimation error $\xi(n)$. For the last term, we use the definition of the gain vector $\mathbf{k}(n)$ to express the inner product $\mathbf{z}^H(n)\mathbf{k}(n)$ as

$$\begin{aligned} \mathbf{z}^H(n)\mathbf{k}(n) &= \mathbf{z}^H(n)\mathbf{\Phi}^{-1}(n)\mathbf{u}(n) \\ &= [\mathbf{\Phi}^{-1}(n)\mathbf{z}(n)]^H\mathbf{u}(n) \\ &= \hat{\mathbf{w}}^H(n)\mathbf{u}(n) \end{aligned}$$

where (in the second line) we have used the Hermitian property of the correlation matrix $\mathbf{\Phi}(n)$, and (in the third line) we have used the fact that $\mathbf{\Phi}^{-1}(n)\mathbf{z}(n)$ equals the least-squares estimate $\hat{\mathbf{w}}(n)$. Accordingly, we may simplify Eq. (13.34) to

$$\begin{aligned} \mathscr{E}_{\min}(n) &= \lambda\mathscr{E}_{\min}(n-1) + d(n)\xi^*(n) - \hat{\mathbf{w}}^H(n)\mathbf{u}(n)\xi^*(n) \\ &= \lambda\mathscr{E}_{\min}(n-1) + \xi^*(n)[d(n) - \hat{\mathbf{w}}^H(n)\mathbf{u}(n)] \\ &= \lambda\mathscr{E}_{\min}(n-1) + \xi^*(n)e(n) \end{aligned} \tag{13.35}$$