# *Adaptive Filters*

V. John Mathews
Scott C. Douglas

# Contents

# Chapter 10

# Recursive Least-Squares Adaptive Filters

The stochastic gradient adaptive filters of the previous chapters were developed using a statistical formulation. The LMS adaptive filter and its variants attempt to determine a solution to the MMSE estimation problem using an approximate gradient search. Because the adaptation properties of these filters depend on the statistics of the input signals, they are difficult to use in some applications. This problem motivates us to search for alternate approaches to the stochastic gradient adaptive filters. We describe a class of adaptive filters based on a deterministic formulation of the linear estimation problem in this chapter. These adaptive filters minimize a weighted sum of the squared estimation errors in a recursive manner. For this reason, they are collectively called recursive least-squares (RLS) adaptive filters. One important difference between RLS adaptive filters and stochastic gradient adaptive filters is that the RLS adaptive filters produce the exact solutions to appropriately-formulated optimization problems at each iteration.

Before we delve into the least-squares problem formulation, it is beneficial to study LMS adaptive filters that employ matrix step sizes. We will see that if the matrix step sizes are properly chosen, it is possible, at least to a first approximation, to eliminate the dependence of the behavior of the mean values of the adaptive filter coefficients on the input signal statistics. The RLS adaptive filter is then developed using a mechanism to estimate the "optimal" matrix step size.

## 10.1 LMS Adaptive Filters With Matrix Step Sizes

The LMS adaptive filter with a constant matrix step size $\boldsymbol{\mu}$ is described by

$$e(n) = d(n) - \mathbf{W}^T(n)\mathbf{X}(n) \tag{10.1}$$

and

$$\mathbf{W}(n+1) = \mathbf{W}(n) + \boldsymbol{\mu}\mathbf{X}(n)e(n), \tag{10.2}$$

where $\boldsymbol{\mu}$ is an $L \times L$-element matrix. By assuming stationarity of the operating environment and invoking the independence assumption, we can show that the mean value of the coefficient error vector of this adaptive filter evolves according to

$$E\{\mathbf{V}(n+1)\} = (\mathbf{I} - \boldsymbol{\mu}\mathbf{R}_{\mathbf{XX}})\, E\{\mathbf{V}(n)\}. \tag{10.3}$$

It is easy to see from (10.3) that if we choose the matrix step size as

$$\boldsymbol{\mu} = \mu\mathbf{R}_{\mathbf{XX}}^{-1}, \tag{10.4}$$

the evolution of the mean values of the coefficient errors does not depend on the statistics of the input signal. If $\mu = 1$, the mean value of the coefficient error vector will converge to the zero vector in one iteration for any input signal for which $\mathbf{R}_{\mathbf{XX}}^{-1}$ exists. Furthermore, when $\mu \neq 1$, the coefficient behavior is still independent of the input signal statistics.

While the above result is conceptually straightforward and elegant, there are difficulties in implementing this algorithm in practice. The algorithm with a matrix step size given by (10.4) requires knowledge of the autocorrelation matrix of the input signal statistics at all times. One of the main reasons for using an adaptive filter is because the input signal statistics may be unknown or time-varying.

All is not lost, however. One solution is to estimate $\mathbf{R}_{\mathbf{XX}}$ at each time instant and substitute its inverse in place of $\boldsymbol{\mu}$ in (10.2). Let $\hat{\mathbf{R}}_{\mathbf{XX}}(n)$ be an estimate of the input autocorrelation matrix at time $n$. The corresponding coefficient update equation is given by

$$\mathbf{W}(n+1) = \mathbf{W}(n) + \mu\hat{\mathbf{R}}_{\mathbf{XX}}^{-1}(n)\mathbf{X}(n)e(n). \tag{10.5}$$

If the estimate of the autocorrelation matrix is accurate, the adaptive filter should operate in a manner that is similar to the case in which the exact autocorrelation matrix is employed. However, the above approach requires inverting the estimated autocorrelation matrix at each iteration. Consequently, the computational complexity of this method is high for all but the simplest of cases.

## A Heuristic Explanation for the Input-Dependent Behavior of the LMS Adaptive Filter

The modified version of the LMS adaptive filter in (10.5) provides some additional insights into the behavior of the scalar step size LMS adaptive filter. For performance that is independent of the input statistics, the adaptive filter should use a matrix step size that is proportional to $\mathbf{R}_{\mathbf{XX}}^{-1}$. Scalar step size algorithms satisfy this requirement only when the input signal is white. When the input signal is correlated, the algorithm deviates from this requirement, and its performance deteriorates as a result.

### 10.1.1 Estimation of the Autocorrelation Matrix

Conceptually, the most straightforward way of estimating the autocorrelation matrix is by time averaging as follows:

$$\hat{\mathbf{R}}_{\mathbf{xx}}(n) = \frac{1}{n}\sum_{k=1}^{n}\mathbf{X}(k)\mathbf{X}^T(k). \tag{10.6}$$

The above estimate assumes that the input signal is available from time $n = 1$ and is valid for $n \geq 1$. If the input vector process $\mathbf{X}(n)$ is ergodic, the estimate $\hat{\mathbf{R}}_{\mathbf{xx}}(n)$ will become more accurate as the number of averaged sample vectors increases. However, the estimate in equation (10.6) has infinite memory, *i.e.*, it weighs all the input data samples equally in obtaining the estimates. Consequently, if the input signal is nonstationary, the estimate at any time may be erroneous. If the statistics of the input signals are only slowly-varying, we can compute a local average of the autocorrelation matrix. We discuss two common approaches for computing this average in what follows.

#### Estimation Using Moving Average Filters

In this approach, a moving average of the instantaneous autocorrelation matrix given by $\mathbf{X}(n)\mathbf{X}^T(n)$ is used as the estimated autocorrelation matrix. This estimate is given by

$$\hat{\mathbf{R}}_{\mathbf{xx}}(n) = \frac{1}{K}\sum_{k=0}^{K-1}\mathbf{X}(n-k)\mathbf{X}^T(n-k). \tag{10.7}$$

In the above, $K$ is the number of samples over which the average is computed. If the statistics change very slowly, we can afford to choose $K$ to be very large. On the other hand, if the statistics of the input signals change relatively quickly, we must choose $K$ to be small. A useful rule of thumb for choosing $K$ is that it should be at least a few times (ten is a typical number) the number of coefficients of the adaptive filter in order for the system to provide reliable results. This statement assumes that the input signal statistics do not change significantly within a span of $K$ consecutive samples. If the operating environment is changing quickly, the averaging must be performed over a smaller number of samples than that suggested by the above recommendation. In such cases, the adaptive filter may not provide good results without appropriate modifications of the algorithm.

An interesting interpretation of the above estimation scheme can be found by noting that the moving averaging process of (10.7) corresponds to lowpass filtering of the matrix "signal" $\mathbf{X}(n)\mathbf{X}^T(n)$. The impulse response of this FIR filter is

$$h(n) = \begin{cases} \frac{1}{K}; & n = 0, 1, \ldots, K-1 \\ 0; & \text{otherwise.} \end{cases} \tag{10.8}$$

Thus, $\hat{\mathbf{R}}_{\mathbf{xx}}(n)$ is the output of the lowpass filter described in (10.8) when its input is $\mathbf{X}(n)\mathbf{X}^T(n)$. The magnitude response of the filter for two different values of $K$ is displayed

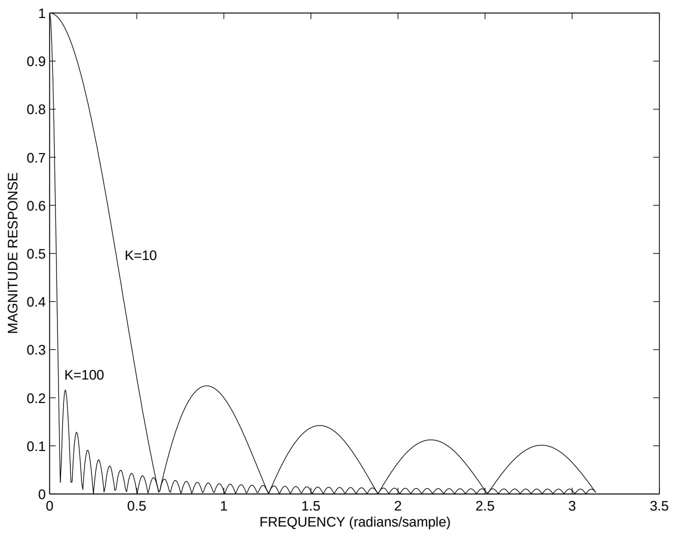


Figure 10.1: The magnitude response of the lowpass filter described by (10.8) for two values of $K$.

in Figure 10.1. As can be seen, the bandwidth of the filter is decreased when the memory of the filter is increased.

**Estimation Using a Single-Pole Lowpass Filter**

The second approach to estimating the autocorrelation matrix arises from the above interpretation of the moving average estimator. Instead of using an FIR lowpass filter, we can use an IIR lowpass filter. Using the IIR filter in place of the moving average filter in the estimation process leads to a certain amount of computational savings. Figure 10.2 shows the block diagram of the IIR filter that is most commonly used to estimate the autocorrelation matrix. The system employs a simple one-pole filter with input-output relationship given by

$$\hat{\mathbf{R}}_{\mathbf{xx}}(n) = \lambda\hat{\mathbf{R}}_{\mathbf{xx}}(n-1) + (1-\lambda)\mathbf{X}(n)\mathbf{X}^T(n). \tag{10.9}$$

In the above, $\lambda$ is a constant between 0 and 1 ($0 < \lambda \leq 1$). The single pole of the filter is located at $z = \lambda$. For the filter in (10.9) to have lowpass characteristics, $\lambda$ must be positive. Also, in order for this filter to be stable, $\lambda$ must be bounded by one.

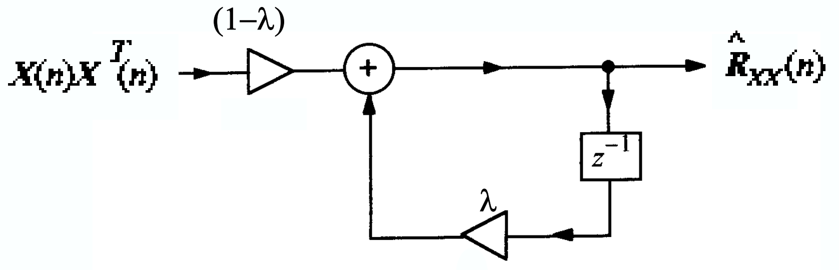


Figure 10.2: Block diagram explaining recursive computation of the input autocorrelation matrix.

### 10.1.2 The "Optimal" Matrix Step Size LMS Adaptive Filter Employing the Single-Pole Lowpass Filter

It is traditional to choose $\mu = 1 - \lambda$. This choice of the step size results in an adaptive filter is described by the following equations:

$$e(n) = d(n) - \mathbf{W}^T(n)\mathbf{X}(n), \tag{10.10}$$

$$\hat{\mathbf{R}}_{\mathbf{XX}}(n) = \lambda\hat{\mathbf{R}}_{\mathbf{XX}}(n-1) + \mathbf{X}(n)\mathbf{X}^T(n) \tag{10.11}$$

and

$$\mathbf{W}(n+1) = \mathbf{W}(n) + \hat{\mathbf{R}}_{\mathbf{XX}}^{-1}(n)\mathbf{X}(n)e(n). \tag{10.12}$$

Comparing (10.11) with (10.9), we see that the input signal to the lowpass filter in (10.11) is scaled by $\mu^{-1} = 1/(1-\lambda)$ when compared with the input signal to the system of (10.9). The inverse of the autocorrelation matrix employed in (10.12) thus effectively employs a step size $\mu = (1-\lambda)$ even though the quantity does not explicitly appear in the coefficient update equations. For reasons that will become clear shortly, the above algorithm is known as the *exponentially-weighted recursive least-squares* adaptive filter. Performance evaluation and derivation of a computationally efficient variation of the above algorithm is the major topic discussed in this chapter.

The magnitude response of the single-pole filter of Figure 10.2 is displayed in Figure 10.3 for two different values of $\lambda$. As $\lambda$ becomes closer to 1, the bandwidth of the filter becomes

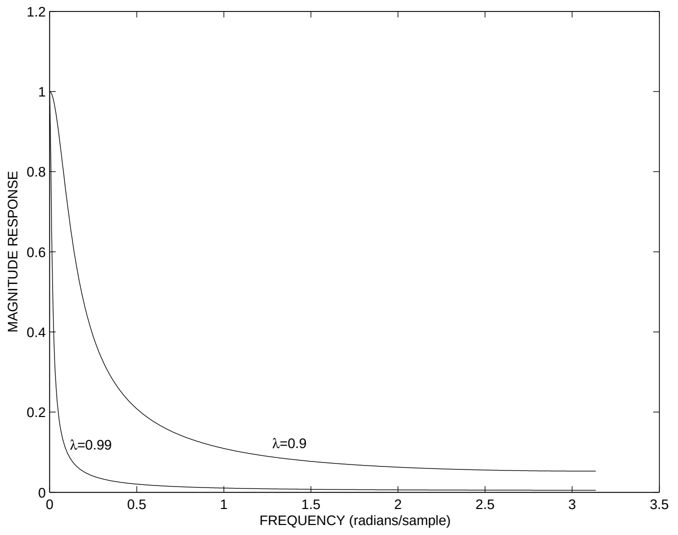


Figure 10.3: Magnitude response of the filter in Figure 10.2 for two values of $\lambda$.

narrower. Analogous to the FIR filter case, when $\lambda$ is closer to 1, the system effectively averages over more number of samples to provide the estimate of the autocorrelation matrix. Similarly, when $\lambda$ is much less than 1, the averaging takes place over a smaller number of samples. We can also make the same inference by examining the unit impulse response function of the lowpass filter, given by

$$h(n) = \begin{cases} \lambda^n; & n \geq 0 \\ 0; & \text{otherwise.} \end{cases} \tag{10.13}$$

The above impulse response signal decays rapidly when $\lambda$ is small, implying that the filter has a short memory span for small values of $\lambda$. Similarly, when $\lambda$ is close to 1, $h(n)$ decays slowly which results in a filter with a relatively long memory.

The impulse response function in (10.13) implies that the estimator in (10.11) can be equivalently written as

$$\hat{\mathbf{R}}_{\mathbf{xx}}(n) = \sum_{k=1}^{n} \lambda^{n-k} \mathbf{X}(k) \mathbf{X}^T(k). \tag{10.14}$$

By choosing $\lambda < 1$, the estimator gives smaller weight to signals that occurred farther in the past. For this reason, $\lambda$ is known as the *forgetting factor* and the output of the lowpass filter in (10.11) or (10.14) is known as an *exponentially-weighted average* of the input matrix.

**Effective Memory Span of the Single-Pole Filter**

We can estimate the effective number of samples over which this averaging is performed by examining the time constant of the lowpass filter used in (10.9). For our purpose, we define the time constant $\tau$ as the approximate value of $n$ for which $h(n) = e^{-1}$, or approximately 36 % of its value at time $n = 0$. This definition is similar to the one employed in Chapter 5. Since $h(n) = \lambda^n h(0)$, we have that

$$\lambda^\tau = e^{-1}. \tag{10.15}$$

Taking the natural logarithm of both sides, we get

$$\tau \ln (\lambda) = -1. \tag{10.16}$$

Since $(1 - \lambda) << 1$ in practice, we can approximate $\ln(\lambda)$ as

$$\ln(\lambda) = \ln(1 - (1 - \lambda)) \approx -(1 - \lambda). \tag{10.17}$$

Substituting the result of (10.17) in (10.16), we get an approximate value for the time constant of the lowpass filter as

$$\tau = \frac{1}{1 - \lambda} \text{ samples.} \tag{10.18}$$

This result agrees with our earlier inference that as $\lambda$ gets closer to 1, the filter effectively averages over more samples.

Up to this point, we have derived a variation of the LMS adaptive filter whose convergence properties are expected to be superior to those of the scalar step size LMS adaptive filter. We now show that the same algorithm is the exact solution of an optimization problem involving the minimization of an exponentially-weighted sum of squared estimation errors. Furthermore, this solution can be evaluated recursively. The name *recursive least-squares adaptive filter* is thus appropriate for this filter.

## 10.2 Exponentially-Weighted RLS Adaptive Filter

Even though several different formulations are possible for the recursive least-squares adaptive filter, we consider only one of them here – the exponentially-weighted RLS adaptive filter. Derivation of RLS adaptive filters employing moving average filters is left as an exercise for the reader. Let $\mathbf{W}(n)$ denote the coefficient vector of the adaptive filter at time $n$. Even though we have not yet formulated the problem, assume that $\mathbf{W}(n)$ is the optimal solution of the problem. Let the estimation error at time $k$ due to the coefficient vector $\mathbf{W}(n)$ be $e_n(k)$, *i.e.*,

$$e_n(k) = d(k) - \mathbf{W}^T(n)\mathbf{X}(k). \tag{10.19}$$

The exponentially-weighted RLS adaptive filter selects the coefficient vector $\mathbf{W}(n)$ so as to minimize the exponentially-weighted sum of the squared errors given by

$$J(n) = \sum_{k=1}^{n} \lambda^{n-k} e_n^2(k). \tag{10.20}$$

Finding the optimal solution is an easy proposition. Substituting (10.19) for $e_n(k)$ into (10.20) and expanding gives

$$\begin{aligned} J(n) = \sum_{k=1}^{n} \lambda^{n-k} d^2(k) + \mathbf{W}^T(n) \left\{ \sum_{k=1}^{n} \lambda^{n-k} \mathbf{X}(k)\mathbf{X}^T(k) \right\} \mathbf{W}(n) \\ -2 \left\{ \sum_{k=1}^{n} \lambda^{n-k} d(k)\mathbf{X}^T(k) \right\} \mathbf{W}(n). \end{aligned} \tag{10.21}$$

Since $J(n)$ is a quadratic function of the coefficients, it has a unique minimum whenever the matrix within brackets in the second term on the right-hand-side of (10.21) is positive definite. In most cases, the matrix $\sum_{k=1}^{n} \lambda^{n-k}\mathbf{X}(k)\mathbf{X}^T(k)$ is positive definite[1] for $n \geq L$.

The optimal coefficient vector can be determined from (10.21) by differentiating $J(n)$ with respect to $\mathbf{W}(n)$ and setting the resulting vector equal to the zero vector. This operation gives

$$2\hat{\mathbf{R}}_{\mathbf{XX}}(n)\mathbf{W}(n) = 2\hat{\mathbf{P}}_{\mathbf{X}\mathrm{d}}(n), \tag{10.22}$$

where $\hat{\mathbf{R}}_{\mathbf{XX}}(n)$ and $\hat{\mathbf{P}}_{\mathbf{X}\mathrm{d}}(n)$ are the exponentially weighted least-squares estimates of the autocorrelation matrix of $\mathbf{X}(n)$ and the cross-correlation vector of $\mathbf{X}(n)$ and $d(n)$, respectively. These quantities are defined as

$$\hat{\mathbf{R}}_{\mathbf{XX}}(n) = \sum_{k=1}^{n} \lambda^{n-k} \mathbf{X}(k)\mathbf{X}^T(k) \tag{10.23}$$

and

$$\hat{\mathbf{P}}_{\mathbf{X}\mathrm{d}}(n) = \sum_{k=1}^{n} \lambda^{n-k} \mathbf{X}(k) d(k), \tag{10.24}$$

respectively. Assuming that $\hat{\mathbf{R}}_{\mathbf{XX}}(n)$ is not a singular matrix, we can solve for $\mathbf{W}(n)$ to get

$$\mathbf{W}(n) = \hat{\mathbf{R}}_{\mathbf{XX}}^{-1}(n)\hat{\mathbf{P}}_{\mathbf{X}d}(n), \tag{10.25}$$

which is exactly the same solution that we obtained in Chapter 2 (see Exercise 2.6) using vector space concepts for linear least-squares estimation.

[1] It is common to assume in analyses of adaptive filters that the input signal satisfies a property called *persistence of excitation*. One of the consequences of persistence of excitation is that the least-squares autocorrelation matrix, defined in (10.23), is positive definite for $n \geq L$. In this book, we will simply assume that this matrix is positive definite and that its eigenvalues are bounded from above and below by finite non-zero constants.

While the solution in (10.25) solves the least-squares estimation problem, it is a computationally expensive solution. Direct implementation of (10.25) requires $O(L^3)$ arithmetic operations at each time instant. A significant factor that contributes to this complexity is the inversion of the estimated autocorrelation matrix that has to be performed at each time instant. The LMS adaptive filter requires only $O(L)$ arithmetic operations per iteration. We now derive a recursive method for computing the solution in (10.25). The implementation, which is known as the *conventional RLS adaptive filter*, requires only $O(L^2)$ arithmetic operations per iteration, and its form resembles that of the algorithm in (10.10) -(10.12).

### 10.2.1 The Conventional RLS Adaptive Filter

The derivation of the conventional RLS algorithm involves a certain amount of straightforward algebraic manipulations. The only result that we must know from matrix algebra is the matrix inversion lemma. Appendix A contains a statement of the matrix inversion lemma.

Our objective is to derive a recursive solution for the least-squares optimization problem of (10.20). Suppose that we have computed the optimal coefficient vector $\mathbf{W}(n-1)$ available at time $n-1$. Given $\mathbf{W}(n-1)$ and all other parameters of the adaptive filters computed at time $n-1$ as well as the new input values $d(n)$ and $\mathbf{X}(n)$ at time $n$, we wish to efficiently estimate $\mathbf{W}(n)$ at time $n$. We start the derivations by finding recursive expressions for $\hat{\mathbf{R}}_{\mathbf{xx}}(n)$ and $\hat{\mathbf{P}}_{\mathbf{x}\mathrm{d}}(n)$. For $\hat{\mathbf{R}}_{\mathbf{xx}}(n)$, we note that

$$\begin{aligned}\hat{\mathbf{R}}_{\mathbf{xx}}(n) &= \sum_{k=1}^{n} \lambda^{n-k}\mathbf{X}(k)\mathbf{X}^T(k)\\ &= \mathbf{X}(n)\mathbf{X}^T(n) + \lambda\sum_{k=1}^{n-1}\lambda^{(n-1)-k}\mathbf{X}(k)\mathbf{X}^T(k)\\ &= \lambda\hat{\mathbf{R}}_{\mathbf{xx}}(n-1) + \mathbf{X}(n)\mathbf{X}^T(n). \qquad (10.26)\end{aligned}$$

In a similar manner, we can show that the cross-correlation vector $\hat{\mathbf{P}}_{\mathbf{x}\mathrm{d}}(n)$ can be recursively computed as

$$\hat{\mathbf{P}}_{\mathbf{x}\mathrm{d}}(n) = \lambda\hat{\mathbf{P}}_{\mathbf{x}\mathrm{d}}(n-1) + \mathbf{X}(n)d(n). \qquad (10.27)$$

In order to compute the coefficient vector $\mathbf{W}(n)$, we require $\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n)$ rather than $\hat{\mathbf{R}}_{\mathbf{xx}}(n)$. Consequently, it is desirable to recursively update the inverse of the estimated autocorrelation matrix. We can derive such a recursive update using the matrix inversion lemma. Applying the matrix inversion lemma gives us the following result:

$$\begin{aligned}\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n) &= \left[\lambda\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n-1) + \mathbf{X}(n)\mathbf{X}^T(n)\right]^{-1}\\ &= \lambda^{-1}\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n-1) - \frac{\lambda^{-2}\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n-1)\mathbf{X}(n)\mathbf{X}^T(n)\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n-1)}{1+\lambda^{-1}\mathbf{X}^T(n)\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n-1)\mathbf{X}(n)}. \qquad (10.28)\end{aligned}$$

It is useful to define a vector $\mathbf{k}(n)$, commonly called the *gain vector,* as

$$\mathbf{k}(n) = \frac{\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n-1)\mathbf{X}(n)}{\lambda + \mathbf{X}^T(n)\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n-1)\mathbf{X}(n)}. \tag{10.29}$$

The gain vector has several useful properties. In particular, we can rewrite (10.28) using $\mathbf{k}(n)$ as

$$\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n) = \frac{1}{\lambda}\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n-1) - \frac{1}{\lambda}\mathbf{k}(n)\mathbf{X}^T(n)\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n-1). \tag{10.30}$$

Now, let us cross-multiply (10.29) and sort out the terms to get

$$\mathbf{k}(n) = \left\{\frac{1}{\lambda}\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n-1) - \frac{1}{\lambda}\mathbf{k}(n)\mathbf{X}^T(n)\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n-1)\right\}\mathbf{X}(n). \tag{10.31}$$

The terms within the brackets above are exactly the same as the right hand side of (10.30), implying that the gain vector can also be written as

$$\mathbf{k}(n) = \hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n)\mathbf{X}(n). \tag{10.32}$$

Substituting (10.27) and (10.30) in (10.25), we get

$$\begin{aligned}
\mathbf{W}(n) &= \left\{\frac{1}{\lambda}\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n-1) - \frac{1}{\lambda}\mathbf{k}(n)\mathbf{X}^T(n)\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n-1)\right\}\left\{\lambda\hat{\mathbf{P}}_{\mathbf{x}\mathrm{d}}(n-1) + \mathbf{X}(n)d(n)\right\} \\
&= \hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n-1)\hat{\mathbf{P}}_{\mathbf{x}\mathrm{d}}(n-1) - \mathbf{k}(n)\mathbf{X}^T(n)\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n-1)\hat{\mathbf{P}}_{\mathbf{x}\mathrm{d}}(n-1) \\
&+ \left\{\frac{1}{\lambda}\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n-1) - \frac{1}{\lambda}\mathbf{k}(n)\mathbf{X}^T(n)\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n-1)\right\}\mathbf{X}(n)d(n).
\end{aligned} \tag{10.33}$$

Recognizing the significance of each term above will lead to the algorithm we are seeking. Recall that

$$\mathbf{W}(n-1) = \hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n-1)\hat{\mathbf{P}}_{\mathbf{x}\mathrm{d}}(n-1). \tag{10.34}$$

Also, the quantitites within the brackets in the third term can be seen to be $\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n)$. Therefore, we can rewrite (10.33) as

$$\mathbf{W}(n) = \mathbf{W}(n-1) - \mathbf{k}(n)\mathbf{X}^T(n)\mathbf{W}(n-1) + \hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n)\mathbf{X}(n)d(n). \tag{10.35}$$

Using the expression for the gain vector from (10.32) in the third term of (10.35) results in

$$\begin{aligned}
\mathbf{W}(n) &= \mathbf{W}(n-1) + \mathbf{k}(n)\left\{d(n) - \mathbf{X}^T(n)\mathbf{W}(n-1)\right\} \\
&= \mathbf{W}(n-1) + \mathbf{k}(n)e_{n-1}(n).
\end{aligned} \tag{10.36}$$

Equations (10.29), (10.30), and (10.36) constitute the conventional recursive least-squares algorithm. Table 10.1 summarizes these recursions. Note that the recursion requires $e_{n-1}(n)$ to update the coefficients. This quantity is denoted by $\varepsilon(n)$ in the table. Similarly, we have employed the notation $e(n)$ to denote the estimation error $e_n(n)$ in the table. A MATLAB program to implement the conventional RLS adaptive filter is given in Table 10.2.

Table 10.1: The conventional RLS adaptive filter

---

**Initialization**

$$\begin{aligned}
\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(0) &= \frac{1}{\delta}\mathbf{I}\ ;\ \ \delta \text{ a small constant} \\
\mathbf{W}(0) &= \mathbf{0}\ ;\ \mathbf{W}(n) \text{ can be initialized arbitrarily if necessary}
\end{aligned}$$

**Recursion**
Given $\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n-1), \mathbf{W}(n-1), \mathbf{X}(n)$ and $d(n)$, compute at time $n$:

$$\mathbf{k}(n) = \frac{\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n-1)\mathbf{X}(n)}{\lambda + \mathbf{X}^T(n)\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n-1)\mathbf{X}(n)}$$

$$\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n) = \frac{1}{\lambda}\left\{\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n-1) - \mathbf{k}(n)\mathbf{X}^T(n)\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n-1)\right\}$$

$$\epsilon(n) = d(n) - \mathbf{W}^T(n-1)\mathbf{X}(n)$$

$$\mathbf{W}(n) = \mathbf{W}(n-1) + \mathbf{k}(n)\epsilon(n)$$

$$e(n) = d(n) - \mathbf{W}^T(n)\mathbf{X}(n)$$

---

Table 10.2: A MATLAB program for implementing the conventional RLS adaptive filter.

```
function [W,dhat,e] = crls2(lambda,delta,W0,x,d);

%  This function adapts a finite-impulse-response (FIR)
%  filter using the conventional recursive least-squares (RLS)
%  adaptive filter.
%
%  Input parameters:
%      lambda   = forgetting factor
%      delta    = initialization constant for autocorrelation matrix
%      W0       = Initial value of W(0) coefficients (L x 1)
%      x        = input signal (num_iter x 1)
%      d        = desired response signal (num_iter x 1)
%
%  Output of program:
%      W        = Evolution of coefficients (L x (num_iter + 1))
%      dhat     = output of adaptive filter (num_iter x 1)
%      e        = error of adaptive filter (num_iter x 1)

L = length(W0);
lambda1 = 1/lambda;
start_iter = 1;
end_iter = min([length(x) length(d)]);
Rxxinv = diag(ones(L,1)/delta);
X = zeros(L,1);
W = zeros(L,end_iter);
e = zeros(end_iter,1);
dhat = zeros(end_iter,1);
W1 = zeros(L,1);
W1(1:L) = W0(1:L);

for n = start_iter:end_iter;

 X = [x(n);X(1:L-1)];
         S = X'*Rxxinv;
         r = 1/(lambda + S*X);
         k = r*Rxxinv*X;
         t = k*S;
         Rxxinv = lambda1*(Rxxinv - t);
```

```
          epsilon = d(n) - W1'*X;
          W(:,n) = W1 + k*epsilon;
          W1 = W(:,n);
         dhat(n) = W1'*X;
         e(n) = d(n) - dhat(n);
end;
```

## 10.2.2 Remarks on the Conventional RLS Adaptive Filter

We make several remarks concerning the conventional RLS adaptive filter in this section.

### ***A priori*** **and** ***A posteriori*** **Errors**

The estimation error signal $e_{n-1}(n)$ in (10.36) is computed by using the optimal coefficient vector at time $n-1$. This error is called the *a priori* estimation error, and is denoted by $\varepsilon(n)$ so that

$$\epsilon(n) = e_{n-1}(n) = d(n) - \mathbf{W}^T(n-1)\mathbf{X}(n) \tag{10.37}$$

Similarly, the *a posteriori* estimation error is defined as the error computed using the optimal coefficient vector at time $n$ and denoted by $e(n)$ as

$$e(n) = e_n(n) = d(n) - \mathbf{W}^T(n)\mathbf{X}(n). \tag{10.38}$$

The *a posteriori* error signal is not used in the coefficient update equations. Consequently, this quantity is not computed in many applications.

### Notational Differences Between RLS and LMS Adaptive Filters

There are a few subtle differences in the notation that we have used for RLS adaptive filters and stochastic gradient adaptive filters. Note that $\mathbf{W}(n+1)$ for the LMS adaptive filter is calculated based on the data available at time $n$. For the RLS adaptive filter, $\mathbf{W}(n+1)$ is calculated based on data at time $n+1$. Consequently, the estimation error

$$e(n) = d(n) - \mathbf{W}^T(n)\mathbf{X}(n). \tag{10.39}$$

as given in (10.1) for the LMS adaptive filter is similar to the *a priori* estimation error as defined in (10.37) for the RLS adaptive filter. It is important to recognize this fact when comparing the two algorithms. In other words, there is a delay of one time sample in the way the coefficients are represented in the LMS and RLS adaptive filters. Even though it would be nice to avoid this confusion and use the same notation for both cases, the two sets of notations are traditional and well-entrenched in the literature. Consequently, we have chosen to follow tradition in this discussion so that comparisons with other works in the literature can be easily made.

### The Conventional RLS Adaptive Filter is the "Optimal" Matrix Step Size LMS Adaptive Filter

Compare the conventional RLS filter with the algorithm given in (10.10) - (10.12). With the help of the interpretation of the gain vector given by (10.32), it becomes clear that the conventional RLS adaptive filter in Table 10.1 and the "optimal" matrix step size stochastic gradient adaptive filter in (10.10)-(10.12) are identical except for the notational difference as explained above. We derived the latter adaptive filter as a stochastic gradient procedure for MMSE estimation. The conventional RLS adaptive filter was derived as an exact solution to the deterministic optimization problem in (10.20). Even though we will use the second interpretation almost exclusively throughout the rest of this chapter, the first interpretation is useful in recognizing the potential advantages that RLS adaptive filters have over traditional stochastic gradient adaptive filters. In particular, we expect that the convergence behaviors of RLS adaptive filters are independent of the statistics of the input signal in general.

### Initialization of the RLS Adaptive Filter

The inverse of the autocorrelation matrix $\hat{\mathbf{R}}_{\mathbf{xx}}(n)$ does not exist for all $n$. In particular, if $\hat{\mathbf{R}}_{\mathbf{xx}}(0)$ is a zero matrix, $\hat{\mathbf{R}}_{\mathbf{xx}}(n)$ is singular for $1 \leq n < L$, since the rank of $\hat{\mathbf{R}}_{\mathbf{xx}}(n)$ is at most $n$. When this is the case, there is no unique solution for the least-squares optimization problem. In fact, since we are trying to estimate the desired response signal samples using $L$ coefficients, and there are only $n < L$ samples to base this estimate on, it is possible to create an exact fit to the data with zero estimation error in an infinite number of ways.

One way of resolving this problem is to set the last $L-n$ coefficients of $\mathbf{W}(n)$ to be zero at time $n$ for $1 \leq n < L$ and then determine the unique solution for the remaining coefficients of $\mathbf{W}(n)$ that gives zero error for the first $n$ iterations. The derivation of the exact initialization procedure for the exponentially-weighted RLS adaptive filter is considered in Exercise 10.8. Even though this approach does give an exact solution to the optimization problem at each time, the coefficients so generated can be greatly influenced by the measurement noise and will most likely not be meaningful. Furthermore, a hardware or software realization of the exact initialization procedure requires additional circuitry or code to implement it. Consequently, most realizations of the conventional RLS adaptive filters initialize the system using

$$\hat{\mathbf{R}}_{\mathbf{xx}}(0) = \delta\mathbf{I}, \tag{10.40}$$

where $\delta$ is a small, positive constant. This form of initialization ensures that the estimated autocorrelation matrix is positive definite during $1 \leq n \leq L-1$. When $\lambda < 1$, the effect of inexact initialization on the system's performance will reduce with time because of the exponentially-fading memory of the adaptive filter. The effect of inexact initialization on the transient response of the adaptive filter is discussed in Section 10.3.3. The coefficient vector is usually initialized to be a vector of all zero elements if nothing is known about the coefficients *a priori*.

### Choosing $\lambda$

As discussed earlier, when $\lambda$ is close to 1, the convergence and tracking of time-varying parameters is slowed. In fact, the value $\lambda = 1$ corresponds to an adaptive filter with infinite memory. Such a system cannot track changes in the parameter values. For fast convergence and tracking of parameters that change quickly, a value of $\lambda$ that is less than one is chosen. When the input signal statistics are sufficiently slowly-varying, the parameter $\lambda$ is usually chosen from the range $1 - 1/10L < \lambda < 1$, so that the memory span of the adaptive filter is at least ten times the number of coefficients.

### Computational Complexity

A count of the number of computations required by the conventional RLS adaptive filter shows that $O(L^2)$ arithmetical operations per iteration are necessary in general. This complexity is an order of magnitude smaller than that required for the direct calculation of the least-squares solution in (10.25). However, the conventional RLS adaptive filter is an order of magnitude more complex than the scalar step size LMS adaptive filter. Some computational simplifications can be achieved by recognizing that $\mathbf{R_{XX}}(n)$ and its inverse are symmetric matrices.

### Another Interpretation of the Gain Vector

The gain vector $\mathbf{k}(n)$ has an interesting and useful interpretation. Define a sequence $\pi_n(k)$ as

$$\pi_n(k) = \begin{cases} 1\ ;\ k = n \\ 0\ ;\ \text{otherwise.} \end{cases} \tag{10.41}$$

Suppose that this sequence is estimated using the input vector sequence $\mathbf{X}(n)$ using the exponentially weighted least-squares criterion. In other words, we desire to minimize

$$J_\pi(n) = \sum_{k=1}^{n} \lambda^{n-k} \left(\pi_n(k) - \mathbf{X}^T(k)\mathbf{W}_\pi(n)\right), \tag{10.42}$$

where $\mathbf{W}_\pi(n)$ is the optimal coefficient vector for the estimate. The optimal coefficient vector for the above problem is given by

$$\begin{aligned} \mathbf{W}_\pi(n) &= \hat{\mathbf{R}}_{\mathbf{XX}}^{-1}(n)\hat{\mathbf{P}}_{\mathbf{X}\pi_n}(n) \\ &= \hat{\mathbf{R}}_{\mathbf{XX}}^{-1}(n)\mathbf{X}(n). \end{aligned} \tag{10.43}$$

Comparing (10.43) with (10.32), we see that $\mathbf{k}(n) = \mathbf{W}_\pi(n)$. The above result means that the gain vector can also be considered as a coefficient vector for the optimal linear least-squares estimate of the sequence $\pi_n(k)$ using $\mathbf{X}(n)$. This interpretation will be useful when we derive computationally efficient $O(L)$ versions of RLS adaptive filters in Chapter 11.

**Example 10.1: Effects of Initialization, Forgetting Factor and Input Signal Statistics**

We consider a two-tap adaptive filter with input correlation matrix and desired response signal as in Example 5.2 It is straightforward to show that the exact initialization of the adaptive filter can be performed as follows:

$$w_0(1) = w_0(2) = \frac{d(1)}{x(1)}, \qquad w_1(1) = 0,$$

and

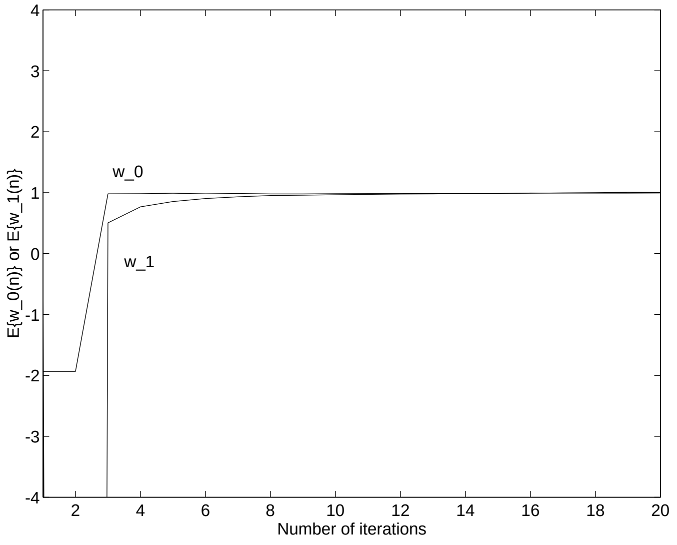
$$w_1(2) = \frac{d(2) - w_0(2)x(2)}{x(1)}.$$

In the first part of the experiments, the coefficients were initialized as above, and at time $n = 2$, the least-squares autocorrelation matrix was formed directly as

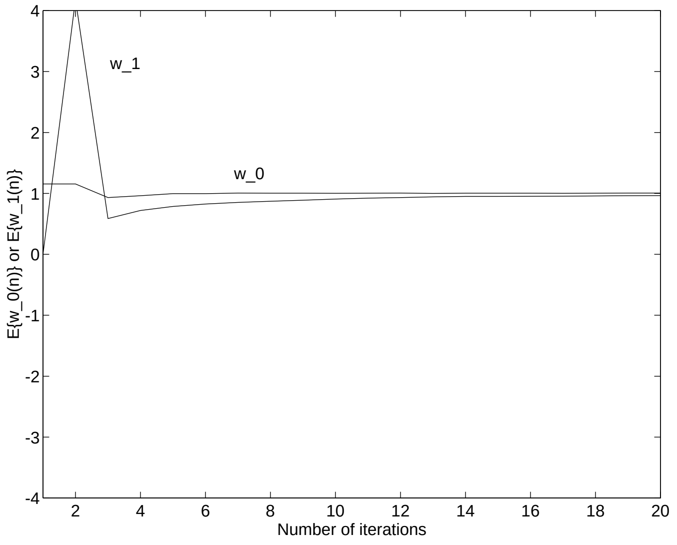
$$\mathbf{R_{XX}}(2) = \lambda \mathbf{X}(1)\mathbf{X}^T(1) + \mathbf{X}(2)\mathbf{X}^T(2)$$

and inverted. Subsequent iterations were performed using the conventional RLS adaptive filter. Figure 10.4 displays the evolution of the two filter coefficients for $\epsilon = 0.1$ and for $\lambda = 0.9$ and 0.999. The corresponding curves for $\epsilon = 0.001$ are shown in Figure 10.5. The curves were obtained by averaging the results of one hundred independent simulations. It can be seen from the figures that the mean values of the adaptive filter coefficients converge to the correct values in two iterations irrespective of the input signal statistics or the choice of the forgetting factor. This observation will be theoretically justified in Section 10.3.3. The small deviations of the mean values of the coefficients from its true values for the first few iterations beyond $n = 2$ is because the coefficients exhibit high variability since they are based on very few input signal samples. This convergence behavior of the RLS adaptive filter exhibited in this example is a significant advantage of such filters over the LMS algorithm. However, since the coefficients are selected based on a minimum number of samples during the initialization procedure, they exhibit very large fluctuations in the first few iterations of the adaptive filter. Consequently, exact initialization is usually not useful in system identification problems.

The next set of experiments involve inexact initialization, which is by far the most common approach for initializing RLS adaptive filters. The evolution of the two coefficients for three different initialization constants are plotted in Figure 10.6 for $\lambda = 0.9$ and $\epsilon = 0.1$. Similar plots for $\lambda = 0.99$ and $\epsilon = 0.1$, $\lambda = 0.9$ and $\epsilon = 0.001$, and $\lambda = 0.99$ and $\epsilon = 0.001$ are shown in Figures 10.7, 10.8 and 10.9, respectively. The coefficients were initialized to $w_0(0) = 3$ and $w_1(0) = -2$ in all the experiments in this part. Unlike the case of exact initialization, the evolution of the coefficients in the early stages does depend on the input signal statistics, the forgetting factor and the initialization parameter $\delta$. The convergence behavior is fast and less dependent on the input signal statistics and the choice of $\lambda$ for the smallest value of $\delta$ selected in the experiments. Consequently, we should select $\delta$ to be as small as possible without introducing numerical stability problems in the RLS adaptive filter. The convergence behavior of the RLS adaptive filter with inexact initialization is theoretically analyzed in Section 10.3.3.

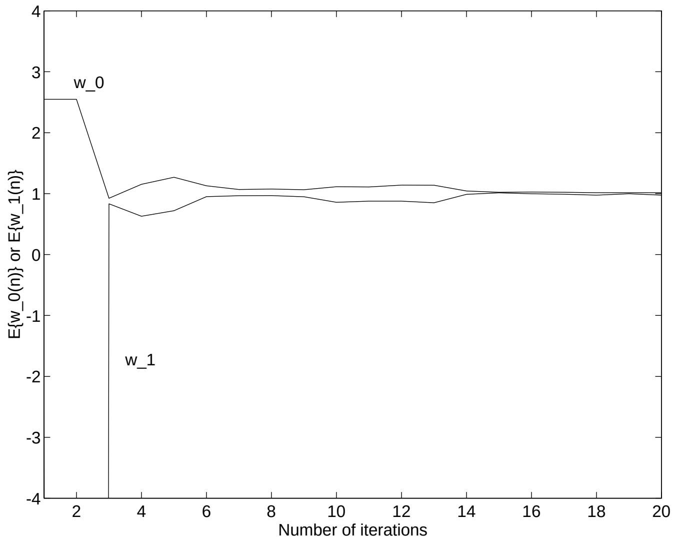


(a) $\lambda = 0.9$

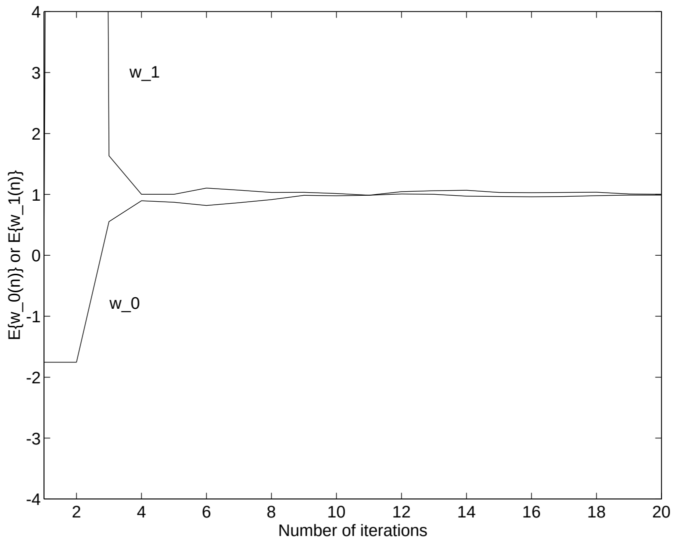


(b) $\lambda = 0.999$

Figure 10.4: Evolution of the filter coefficients in Example 10.1 for $\epsilon = 0.1$ and exact initialization.

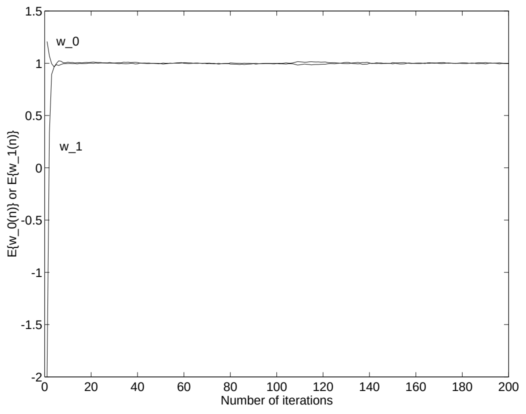


(a) $\lambda = 0.9$

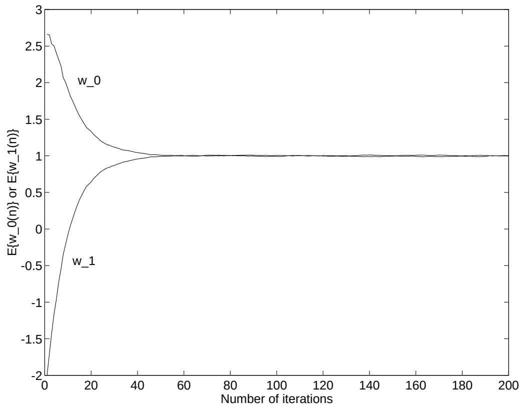


(b) $\lambda = 0.999$

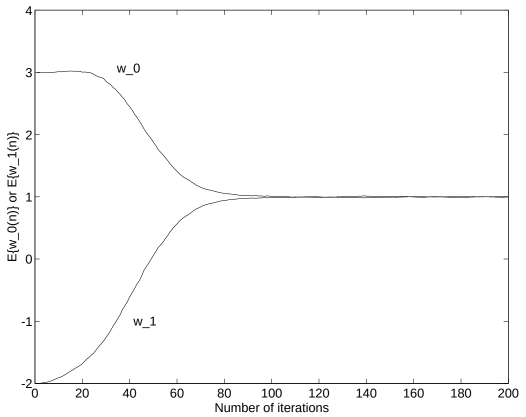
Figure 10.5: Evolution of the filter coefficients in Example 10.1 for $\epsilon = 0.001$ and exact initialization.



(a) $\delta = 0.001$

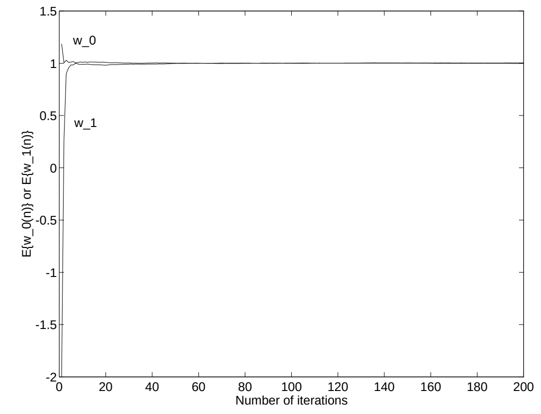


(b) $\delta = 1.0$

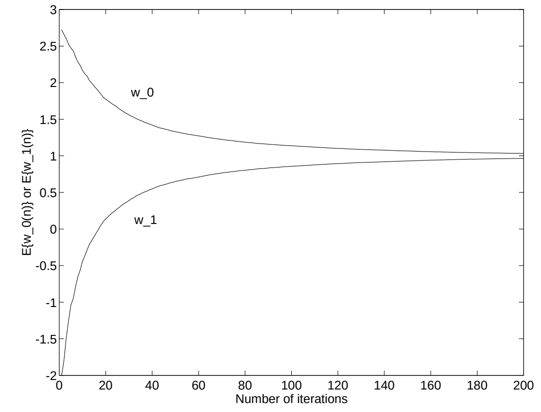


(c) $\delta = 100.0$

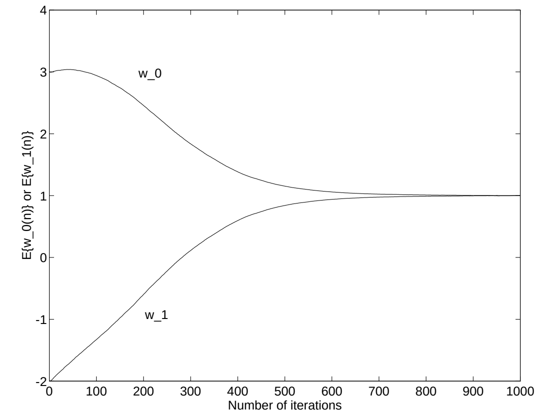
Figure 10.6: Evolution of the filter coefficients in Example 10.1 for $\epsilon = 0.1$, $\lambda = 0.9$, and inexact initialization.



(a) $\delta = 0.001$

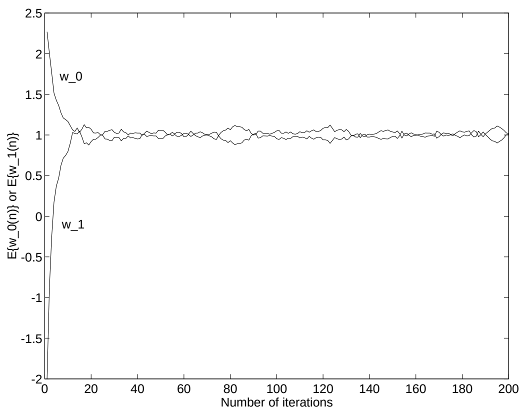


(b) $\delta = 1.0$

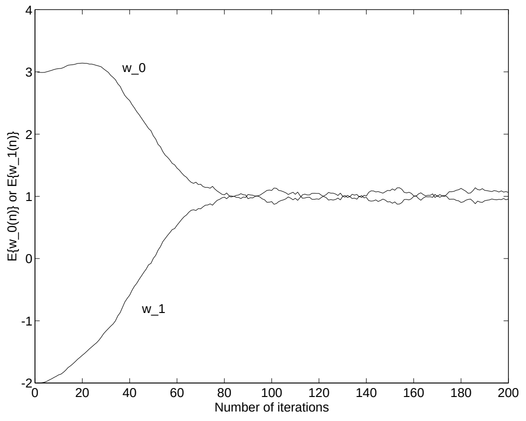


(c) $\delta = 100.0$

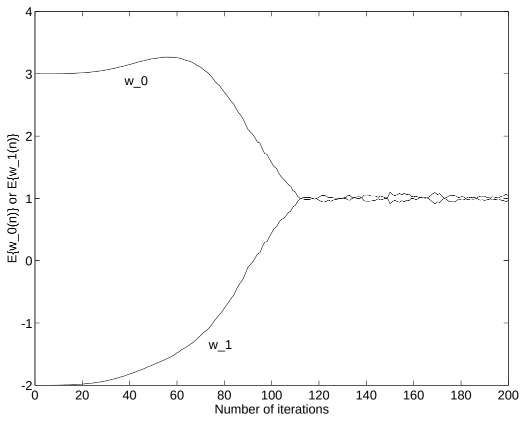
Figure 10.7: Evolution of the filter coefficients in Example 10.1 for $\epsilon = 0.1$, $\lambda = 0.99$, and inexact initialization.



(a) $\delta = 0.001$

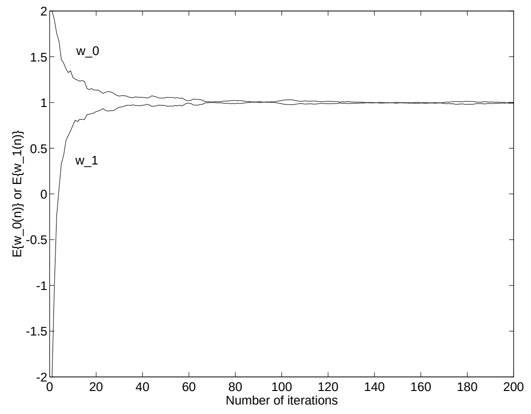


(b) $\delta = 1.0$

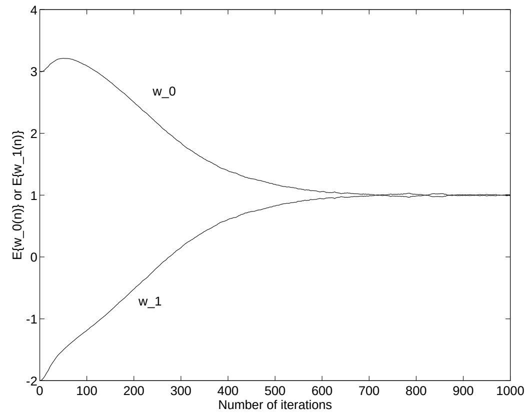


(c) $\delta = 100.0$

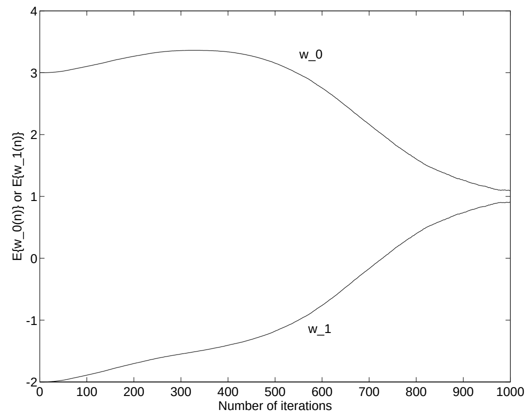
Figure 10.8: Evolution of the filter coefficients in Example 10.1 for $\epsilon = 0.001$, $\lambda = 0.9$, and inexact initialization.



(a) $\delta = 0.001$



(b) $\delta = 1.0$



(c) $\delta = 100.0$

Figure 10.9: Evolution of the filter coefficients in Example 10.1 for $\epsilon = 0.001$, $\lambda = 0.99$, and inexact initialization.

**Example 10.2: Performance of the RLS Adaptive Filter in an Abruptly Changing Environment**

We consider the identification of the two-coefficient system of Example 10.1 with the input signal $x(n)$ generated using $\epsilon = 0.001$. The output of the unknown system was corrupted by additive white Gaussian noise with zero mean value and variance $\sigma_\eta^2 = 0.01$. The coefficients of the unknown system changed from $[1, 1]^T$ to $[2, 0]^T$ abruptly at $n = 2,000$. We compare the performances of the conventional RLS adaptive filter with $\lambda = 0.99$ and $\delta = 0.001$ and the LMS adaptive filter with $\mu = 0.01/1.001$ in this example. The input signal power can be shown to be 1.001. It is shown analytically in Section 10.3.5 that both the algorithms exhibit identical excess mean-square errors for our input signal and the choice of parameters. Figure 10.10a displays the mean values of the two coefficients obtained from one hundred independent experiments for the RLS adaptive filter. The corresponding curves for the LMS adaptive filter are shown in Figure 10.10b. The LMS filter coefficients were initialized to the true values of the coefficients of the unknown system at time $n = 0$. We can see that the coefficients of the RLS adaptive filter converge to their true values after the abrupt change in a uniform manner, unlike those of the LMS adaptive filter which exhibits slow and non-uniform convergence characteristics.

While this example is only an illustration of the superior convergence characteristics of the RLS adaptive filter over those of the LMS adaptive filter in dealing with abruptly changing environments, behavior similar to that seen in this example is typical.

## 10.3 Analysis of the Exponentially-Weighted RLS Adaptive Filter

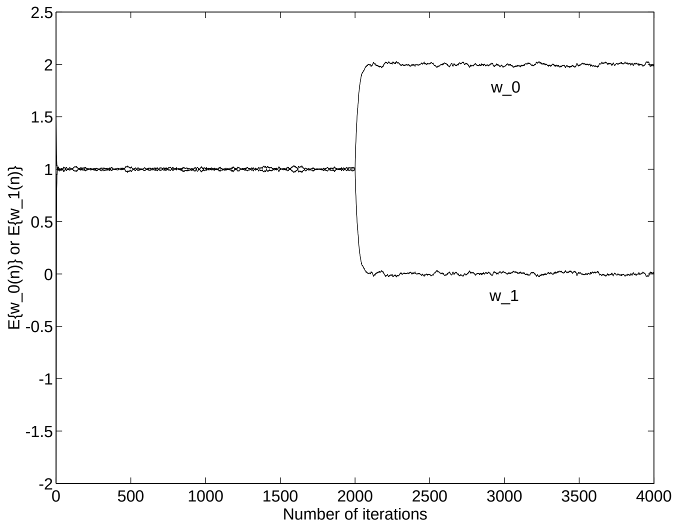
We analyze the convergence, tracking, and steady-state properties of the exponentially weighted RLS adaptive filter in this section. Because the analysis techniques employed are similar to that for the LMS adaptive filter, our derivations are less descriptive. The results of the analyses are used to draw conclusions about the relative merits of the LMS and RLS adaptive filters in certain situations.

### 10.3.1 Major Assumptions

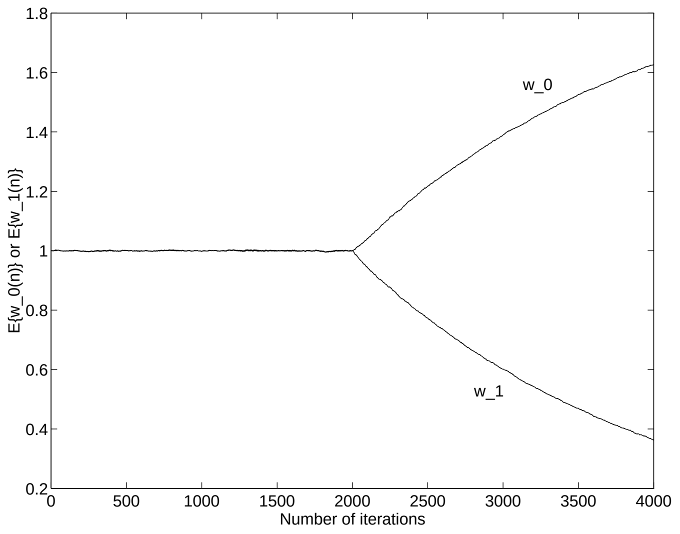
The analysis technique of this section relies on the methods used in [Eleftheriou 1986] and [Macchi 1991]. We employ the following assumptions to make the analysis tractable. Similar assumptions were employed in the analysis of the LMS adaptive filter in Chapter 5.

1. The adaptive filter is operating in the system identification mode such that we can model the desired response signal as

$$d(n) = \mathbf{W}_{opt}^T(n)\mathbf{X}(n) + \eta(n), \tag{10.44}$$



(a) RLS adaptive filter



(b) LMS adaptive filter

Figure 10.10: Evolution of the filter coefficients in Example 10.2 for RLS and LMS adaptive filters.

where the measurement noise $\eta(n)$ is assumed to be i.i.d and independent of $\mathbf{X}(n)$.

2. The sole source of nonstationarity within the system is the random behavior of the optimum coefficient vector $\mathbf{W}_{opt}(n)$. The evolution of $\mathbf{W}_{opt}(n)$ is described by

$$\mathbf{W}_{opt}(n+1) = \mathbf{W}_{opt}(n) + \mathbf{M}(n), \tag{10.45}$$

where $\mathbf{M}(n)$ is a sequence of vectors such that

$$E\{\mathbf{M}(n)\} = \mathbf{0} \tag{10.46}$$

and

$$E\{\mathbf{M}(k)\mathbf{M}^T(n)\} = \sigma_m^2 \mathbf{I}\delta_{k-n}. \tag{10.47}$$

Furthermore, the sequence $\mathbf{M}(n)$ is independent of the input vector sequence $\mathbf{X}(n)$.

3. The forgetting factor $\lambda$ is very close to one so that $1 - \lambda << 1$. This assumption corresponds to the small step size assumption employed in the analysis of the LMS adaptive filter.

## 10.3.2 Convergence of the Coefficients For Exact Initialization

It can be shown using simple algebraic manipulations and the signal model of (10.44) (see Exercise 10.9) that the coefficient error vector $\mathbf{V}(n)$ is given by

$$\mathbf{V}(n) = \lambda^n \hat{\mathbf{R}}_{\mathbf{XX}}^{-1}(n)\hat{\mathbf{R}}_{\mathbf{XX}}(0)\mathbf{V}(0) + \hat{\mathbf{R}}_{\mathbf{XX}}^{-1}(n)\left[\sum_{k=1}^{n} \lambda^{n-k}\mathbf{X}(k)\eta(k)\right] \tag{10.48}$$

when the adaptive filter is operating in a nonstationary environment. Since the signal model assumes that $\eta(n)$ is independent of the input signal and it is a zero-mean process, we can see that the mean value of the coefficient error is zero if the autocorrelation matrix is initialized to be an all-zero matrix and the matrix $\hat{\mathbf{R}}_{\mathbf{XX}}^{-1}(n)$ exists. Recall that $\hat{\mathbf{R}}_{\mathbf{XX}}(n)$ is singular for $1 \leq n < L$. We assume in our derivations that the estimated autocorrelation matrix has full rank for $n \geq L$. In such situations, the mean values of the coefficients converge to their true values in $L$ samples for all values of $\lambda$ in the range $0 < \lambda \leq 1$.

The above analysis clearly shows that RLS adaptive filters have the potential to provide performance that is independent of the statistics of the input signal. This statement is also supported by the experimental results of Figures 10.4 and 10.5. Inexact initialization may reduce or eliminate the ability of the RLS adaptive filter to converge quickly to the correct solution, as we have already seen in Example 10.1. In the following sections, we analyze the convergence characteristics of the RLS adaptive filter when inexact initialization is employed, and also derive expressions for the excess mean-square estimation error in the steady-state using a highly simplified model of adaptation.

### 10.3.3 Evolution of the Mean Coefficient Values For Inexact Initialization

Since $1-\lambda$ is small, the variance of $\hat{\mathbf{R}}_{\mathbf{xx}}(n)$ is small compared with its mean value. Therefore, we approximate $\hat{\mathbf{R}}_{\mathbf{xx}}(n)$ as an *almost deterministic* quantity with mean value given by

$$\begin{aligned} E\{\hat{\mathbf{R}}_{\mathbf{xx}}(n)\} &= \lambda E\{\hat{\mathbf{R}}_{\mathbf{xx}}(n-1)\} + E\{\mathbf{X}(n)\mathbf{X}^T(n)\} \\ &= \lambda E\{\hat{\mathbf{R}}_{\mathbf{xx}}(n-1)\} + \tilde{\mathbf{R}}_{\mathbf{xx}}(n), \end{aligned} \tag{10.49}$$

where the $(i,j)$th element of the matrix $\tilde{\mathbf{R}}_{\mathbf{xx}}(n)$ is given by

$$\tilde{r}_{i,j}(n) = \begin{cases} r_{xx}(i-j) & ; 0 \le i,j < n \\ 0 & ; \text{otherwise} \end{cases} \tag{10.50}$$

That is, $\tilde{\mathbf{R}}_{\mathbf{xx}}(n)$ is identical to $\mathbf{R}_{\mathbf{xx}}$ if $n \ge L$. Otherwise, the $n \times n$ elements on the top left corner of $\tilde{\mathbf{R}}_{\mathbf{xx}}(n)$ are identical to the corresponding elements of $\mathbf{R}_{\mathbf{xx}}$ and the rest of the elements are zero. This definition is necessary during the initialization period $1 \le n \le L-1$ since it takes $L$ iterations to fill all the entries of the input vector $\mathbf{X}(n)$ with non-zero elements. When $\hat{\mathbf{R}}_{\mathbf{xx}}(0)$ is set to $\delta\mathbf{I}$, we can show that

$$E\{\hat{\mathbf{R}}_{\mathbf{xx}}(n)\} = \lambda^n \delta \mathbf{I} + \mathbf{R}'_{\mathbf{xx}}(n) \; ; \; n \ge 1, \tag{10.51}$$

where the $(i,j)$th element of $\mathbf{R}'_{\mathbf{xx}}(n)$ is given by

$$r'_{i,j}(n) = \begin{cases} 0 & ; i \text{ or } j \ge n \\ \dfrac{1-\lambda^{n-\max(i,j)}}{1-\lambda} r_{xx}(i-j) & ; \text{otherwise.} \end{cases} \tag{10.52}$$

Because $\hat{\mathbf{R}}_{\mathbf{xx}}(n)$ is an almost deterministic matrix sequence, we can use the approximation that this matrix process is uncorrelated with $\mathbf{X}(n)$ with reasonable accuracy. Furthermore, we can argue with the help of (10.51) that $\hat{\mathbf{R}}^{-1}_{\mathbf{xx}}(n)$ is also an almost deterministic quantity that evolves as

$$\hat{\mathbf{R}}^{-1}_{\mathbf{xx}}(n) \approx \left[\lambda^n \delta \mathbf{I} + \mathbf{R}'_{\mathbf{xx}}(n)\right]^{-1} \; ; \; n \ge 1. \tag{10.53}$$

As a result of the above approximations, we can express the gain vector $\mathbf{k}(n)$ as

$$\mathbf{k}(n) \approx \left[\lambda^n \delta \mathbf{I} + \mathbf{R}'_{\mathbf{xx}}(n)\right]^{-1} \mathbf{X}(n). \tag{10.54}$$

Therefore, we can approximate the coefficient update equation as

$$\mathbf{W}(n) = \mathbf{W}(n-1) + \boldsymbol{\mu}(n)\mathbf{X}(n)\left(d(n) - \mathbf{W}^T(n-1)\mathbf{X}(n)\right), \tag{10.55}$$

where $\boldsymbol{\mu}(n)$ is a deterministic matrix sequence given by

$$\boldsymbol{\mu}(n) = \left[\lambda^n \delta \mathbf{I} + \mathbf{R}'_{\mathbf{xx}}(n)\right]^{-1}. \tag{10.56}$$

The above result indicates that the conventional RLS adaptive filter is similar to the matrix step size LMS adaptive filter of (10.10) and (10.12) with a time-varying step size sequence $\boldsymbol{\mu}(n)$. We can now use the same analysis technique for the LMS adaptive filter to show that the mean value of the coefficient error vector evolves as

$$E\{\mathbf{V}(n)\} = (\mathbf{I} - \boldsymbol{\mu}(n)\mathbf{R}_{\mathbf{xx}})\, E\{\mathbf{V}(n-1)\}. \tag{10.57}$$

**Effect of Input Signal Statistics**

The behavior of the RLS adaptive filter that is documented in Example 10.1 for inexact initialization can be explained with the help of our analysis in the following manner. Recall that we can compare the RLS adaptive filter with inexact initialization with an LMS adaptive filter with matrix step size given by (10.56). For large $n$, $\boldsymbol{\mu}(n)$ in (10.56) is almost equal to $(1-\lambda)\mathbf{R}_{\mathbf{xx}}^{-1}$. Substituting this in (10.57), we find that the evolution of the mean coefficient values do not depend on the input signal except in the initial stages of adaptation. However, if $\delta$ is very large, the step size is dominated by the diagonal elements during the initial stages, and the algorithm behavior resembles that of the scalar step size LMS adaptive filter. Fortunately, the effect of initialization on the convergence properties decays exponentially with time. The effects of inexact initialization can be reduced by choosing $\delta$ to be as small as possible. The parameter $\delta$ should be such that the condition number of the estimated autocorrelation matrix $\hat{\mathbf{R}}_{\mathbf{xx}}(n)$ is small enough to allow the inversion of this matrix without introducing significant numerical errors during the initialization period. A rule of thumb for the choice of $\delta$ is that it should be no more than a tenth of the input signal power.

## 10.3.4 Steady-State Mean-Square Analysis

We now analyze the RLS adaptive filter for the *a priori* excess mean-square estimation error in the steady-state. We consider the *a priori* error rather than the *a posteriori* error for two reasons:

1. The analysis of the *a priori* estimation error is somewhat simpler than that of the *a posteriori* estimation error. This is a consequence of the fact that $\mathbf{W}(n)$ depends on the input signal pair $\{\mathbf{X}(n), d(n)\}$ explicitly and, therefore, $\mathbf{W}(n)$ and $\mathbf{X}(n)$ are correlated even under conditions in which the independence assumption is true.

2. Such an analysis facilitates a fair comparison of the performances of the LMS and RLS adaptive filters since the analyses of the LMS adaptive filter in previous chapters were based on the *a priori* estimation errors. Recall that $\mathbf{W}(n)$ for the LMS adaptive filter is calculated from signals obtained until time $n-1$.

## Analysis

Since we are interested in the performance of the RLS adaptive filters in the steady-state, we employ the approximate coefficient update equation given by (10.55) in the analysis. We define the *a priori* coefficient error vector as

$$\mathbf{V}(n) = \mathbf{W}(n-1) - \mathbf{W}_{opt}(n). \tag{10.58}$$

This definition is consistent with the definition of the coefficient error vector for the LMS adaptive filter, in view of the differences in the notation employed for the LMS and RLS adaptive filters. Subtracting $\mathbf{W}_{opt}(n+1)$ from both sides of (10.55), and employing the model for the desired response signal in (10.44), we get

$$\mathbf{V}(n+1) = \mathbf{V}(n) - \mathbf{M}(n) + (1-\lambda)\mathbf{R}_{\mathbf{XX}}^{-1}\mathbf{X}(n)\left(\eta(n) - \mathbf{V}^T(n)\mathbf{X}(n)\right). \tag{10.59}$$

We proceed by post-multiplying both sides of this equation by their respective transposes, taking the statistical expectations of each term, and neglecting the term involving the fourth-order expectations of the input signal because it is weighted by $(1-\lambda)^2$. These operations result in

$$\mathbf{K}(n+1) = \mathbf{K}(n) + \sigma_m^2\mathbf{I} + (1-\lambda)^2\sigma_\eta^2\mathbf{R}_{\mathbf{XX}}^{-1} - 2(1-\lambda)\mathbf{K}(n), \tag{10.60}$$

where $\mathbf{K}(n)$ is the autocorrelation matrix of the coefficient error vector as defined in Chapter 5. In the steady-state, $\mathbf{K}(n+1) = \mathbf{K}(n) = \mathbf{K}_{ss}$, and we can solve for the steady-state correlation matrix $\mathbf{K}_{ss}$ as

$$\mathbf{K}_{ss} = \frac{(1-\lambda)}{2}\sigma_\eta^2\mathbf{R}_{\mathbf{XX}}^{-1} + \frac{\sigma_m^2}{2(1-\lambda)}\mathbf{I}. \tag{10.61}$$

The steady-state excess mean-square error is given by

$$\begin{aligned}\xi_{ex,ss} &= \mathrm{tr}[\mathbf{K}_{ss}\mathbf{R}_{\mathbf{XX}}] \\ &= \frac{1}{2}\left((1-\lambda)\sigma_\eta^2 + \frac{\sigma_m^2\sigma_x^2}{1-\lambda}\right)L \qquad (10.62)\end{aligned}$$

for stationary input signals.

The above analysis shows that the steady-state excess mean-square error consists of the adaptation error given by $L(1-\lambda)\sigma_\eta^2/2$ and the lag error given by $L\sigma_m^2\sigma_x^2/(2(1-\lambda))$. The adaptation error is reduced when $\lambda$ is close to one. The lag error, on the other hand, is reduced by making $\lambda$ small. These characteristics are similar to that of the LMS adaptive filter for the nonstationarity model of (10.44).

**Example 10.3: Verification of Analytical Results**

Table 10.3: Comparison of analytical and experimental results of the excess mean-square error in Example 10.3.

| $\lambda$ | $\epsilon$ | $\sigma_\eta^2$ | $\sigma_m^2$ | Analytical Excess MSE | Experimental Excess MSE |
|---|---|---|---|---|---|
| 0.990 | 0.001 | $10^{-1}$ | $10^{-5}$ | 0.0015 | 0.0014 |
| 0.990 | 0.001 | $10^{-1}$ | 0 | 0.0010 | 0.0011 |
| 0.990 | 0.001 | 0 | $10^{-5}$ | 0.0005 | $4.49 \times 10^{-4}$ |
| 0.999 | 0.001 | 1.0 | $10^{-5}$ | 0.0015 | 0.0018 |
| 0.999 | 0.001 | 1.0 | 0 | 0.0010 | 0.0016 |
| 0.999 | 0.001 | 0 | $10^{-5}$ | 0.0005 | $2.88 \times 10^{-4}$ |
| 0.990 | 0.100 | $10^{-1}$ | $10^{-5}$ | 0.0016 | 0.0016 |
| 0.990 | 0.100 | $10^{-1}$ | 0 | 0.0010 | 0.0010 |
| 0.990 | 0.100 | 0 | $10^{-5}$ | $5.50 \times 10^{-4}$ | $5.69 \times 10^{-4}$ |
| 0.999 | 0.100 | 1.0 | $10^{-6}$ | 0.0016 | 0.0016 |
| 0.999 | 0.100 | 1.0 | 0 | 0.0010 | 0.0012 |
| 0.999 | 0.100 | 0 | $10^{-6}$ | $5.50 \times 10^{-4}$ | $4.23 \times 10^{-4}$ |

This example compares the *a priori* excess MSE of the RLS adaptive filter given by (10.62) with experimental results. We use the system identification problem of Example 10.1 with exact initialization. The excess MSE was measured as the average value of the squared difference between the estimation error and the measurement noise over 500 iterations in the range $2501 \leq n \leq 3000$. These values were further averaged over one hundred independent simulations. The unknown system's coefficients at $n = 0$ was $[1 \;\; 1]^T$, and the coefficients varied according to the random walk model of (10.45) with specified values of $\sigma_m^2$ for each set of experiments. Since we are only interested in the steady-state values of the excess mean-square error, the adaptive filter coefficients were initialized to those of the unknown system. Table 10.3 tabulates the experimental and theoretical values for several values of $\epsilon$, $\lambda$, $\sigma_\eta^2$ and $\sigma_m^2$. We can see from this table that the analytical results show good agreement with the experimental results in spite of the highly simplified analysis procedure we employed. The experimental results also show that the contributions of the adaptation error and the tracking error are additive, as predicted by our analysis.

### 10.3.5 Comparison with LMS Adaptive Filter

Recall from (5.111) that, for the LMS adaptive filter with a small step size $\mu$, the steady-state excess mean-square error is given by

$$\xi^{(L)}_{ex,ss} = \frac{1}{2}\left(\mu\sigma^2_\eta\sigma^2_x + \frac{\sigma^2_m}{\mu}\right)L. \tag{10.63}$$

Comparing this result with the expression for the steady-state excess mean-square error in (10.62), we find that the steady-state excess mean-square error of the RLS adaptive filter $\xi^{(R)}_{ex,ss} = \xi^{(L)}_{ex,ss}$ when $\mu = (1-\lambda)/\sigma^2_x$. Thus, for the nonstationary model we analyzed, both the RLS and LMS adaptive filters achieve the same excess mean-square error for properly chosen convergence parameters.

One advantage of the RLS adaptive filters is the relative insensitivity of the convergence speed to the eigenvalue spread of the autocorrelation matrix of the input signal. This advantage becomes significant when $\lambda_{max}/\lambda_{min}$, the ratio of the maximum and minimum eigenvalues, is large. For the same speed of convergence, we can expect the RLS adaptive filter to show smaller steady-state error than the LMS adaptive filter in most situations.

#### Comparison of Tracking Performance

From (10.62) and (10.63), we can show that the optimal choices of $\mu$ and $\lambda$ that minimize the steady-state excess mean-square error for the LMS and RLS adaptive filters are given by

$$\mu_{opt} = \frac{\sigma_m}{\sigma_x\sigma_\eta} \tag{10.64}$$

and

$$\lambda_{opt} = 1 - \frac{\sigma_m\sigma_x}{\sigma_\eta}, \tag{10.65}$$

respectively. The corresponding excess MSEs for these optimal choices are identical for the two adaptive filters. Thus, it appears that both RLS and LMS adaptive filters possess similar tracking capability. However, the value of $\mu_{opt}$ derived above may be outside the range of values of the step size that allows stable operation of the adaptive filter in practice. In such situations, the RLS adaptive filter has a significant tracking advantage over the LMS adaptive filter. As an example, for the optimal step size of the LMS adaptive filter to not exceed the necessary and sufficient bound for mean-square stability given by (5.89) we require that

$$\frac{\sigma_m}{\sigma_x\sigma_\eta} < \frac{2}{3L\sigma^2_x} \tag{10.66}$$

implying that

$$L\frac{\sigma_x\sigma_m}{\sigma_\eta} < \frac{2}{3} \tag{10.67}$$

This condition is met in most practical situations, and therefore, the RLS adaptive filter possesses no tracking advantage over the LMS adaptive filter. An exception to this statement is in certain applications in mobile communication systems where the communication channels exhibit fast fading characteristics corresponding to large increments in the optimal coefficients from one time instant to the next [Ling 1985]. Since the step size bound for the LMS adaptive filter decreases with the number of coefficients, it is possible that the condition of (10.67) is violated for large filter lengths. In such situations, the RLS adaptive filter has superior tracking characteristics over the LMS adaptive filter.

It is important to recognize that the above analysis has been performed for a specific type of nonstationarity. For other types of nonstationarities, either the RLS or the LMS adaptive filter may exhibit tracking advantage over the other adaptive filter. Two such situations are considered in Exercise 10.10.

## 10.4 Error Propagation in Conventional RLS Adaptive Filters

Unlike LMS adaptive filters, RLS adaptive filters propagate the inverse of the least-squares autocorrelation matrix from one iteration to the next. In particular, we see from (10.30) that the inverse matrix is updated as a difference of two matrices. Ideally, $\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n)$ is a symmetric matrix and therefore positive definite for $n \geq L$ for most input signals. However, imprecise calculations during the update procedure may make the matrix singular or indefinite. Even if such an occurence does not lead to catastrophic failures, the resulting coefficient values may deviate significantly from their infinite precision values, and thus may be meaningless. This type of behavior usually happens when the autocorrelation matrix of the input signal has very small eigenvalues, and after a large number of iterations.

In this section, we consider two different implementations of the conventional RLS adaptive filter. When implemented with infinite precision, both realizations produce identical results. When an error is introduced into the calculation of $\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n)$ such that this matrix is made unsymmetric, the first realization propagates the unsymmetric portion of $\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n)$ in an unstable manner. Rather than perform a complete analysis of the numerical properties of the two realizations, we illustrate the above statement by considering the propagation of a single error $\delta\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n-1)$ introduced to the inverse of the least-squares autocorrelation matrix at time $n-1$, when the adaptive filters are implemented with infinite precision. This analysis provides valuable insights into the behavior of the conventional RLS adaptive filters. In particular, we show that, in order to operate in a numerically stable manner, the conventional RLS adaptive filter should be implemented such that $\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n)$ is a symmetric matrix.

Table 10.4: Two realizations of the conventional RLS adaptive filter.

| Quantity | CRLS1 | CRLS2 |
|---|---|---|
| $\mathbf{S}(n)$ | $\mathbf{X}^T(n)\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n-1)$ | $\mathbf{X}^T(n)\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n-1)$ |
| $r(n)$ | $(\lambda+\mathbf{S}(n)\mathbf{X}(n))^{-1}$ | $(\lambda+\mathbf{S}(n)\mathbf{X}(n))^{-1}$ |
| $\mathbf{k}(n)$ | $r(n)\mathbf{S}^T(n)$ | $r(n)\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n-1)\mathbf{X}(n)$ |
| $\mathbf{T}(n)$ | $\mathbf{k}(n)\mathbf{S}(n)$ | $\mathbf{k}(n)\mathbf{S}(n)$ |
| $\mathbf{R}_{\mathbf{xx}}^{-1}(n)$ | $\frac{1}{\lambda}\left(\mathbf{R}_{\mathbf{xx}}^{-1}(n-1)-\mathbf{T}(n)\right)$ | $\frac{1}{\lambda}\left(\mathbf{R}_{\mathbf{xx}}^{-1}(n-1)-\mathbf{T}(n)\right)$ |

**Two Realizations of the Conventional RLS Adaptive Filter**

The iterations to update $\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n)$ using the two realizations denoted by CRLS1 and CRLS2 are tabulated in Table 10.4. The only difference between the two realizations is in the third step that calculates $\mathbf{k}(n)$. Since the CRLS2 realization uses $\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n-1)$ directly in this step, any asymetrical errors introduced into the calculation of $\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n-1)$ decay exponentially fast as they are propagated into future iterations. We show in our analysis that this property is extremely important in the numerically stable operation of the conventional RLS adaptive filter.

### 10.4.1 Error Analysis

Let $\underline{\hat{\mathbf{R}}}_{\mathbf{xx}}^{-1}(n-1)$ denote the distorted version of $\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n)$ so that

$$\underline{\hat{\mathbf{R}}}_{\mathbf{xx}}^{-1}(n-1)=\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n-1)+\delta\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n-1). \tag{10.68}$$

We assume that the error matrix $\delta\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n-1)$ is small so that contributions from terms involving product of two or more elements of this matrix can be neglected in our analysis. Assuming that no other errors are introduced into the system, we wish to determine how the elements of the above error matrix propagate to time $n$. We use $\delta\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n)$ to denote the error matrix at time $n$.

## Error Propagation in CRLS1

We first obtain an expression for $\underline{r}(n)$, the perturbed version of $r(n)$ computed by the adaptive filter:

$$\begin{aligned}\underline{r}(n) &= \frac{1}{\lambda + \mathbf{X}^T(n)(\underline{\hat{\mathbf{R}}}_{\mathbf{xx}}^{-1}(n-1)\mathbf{X}(n)} \\ &= \frac{1}{\lambda + \mathbf{X}^T(n)(\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n-1) + \delta\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n-1)\mathbf{X}(n)} \\ &\approx \frac{1}{\lambda + \mathbf{X}^T(n)\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n-1)\mathbf{X}(n)}\left(1 - \frac{\mathbf{X}^T(n)\delta\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n-1)\mathbf{X}(n)}{\lambda + \mathbf{X}^T(n)\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n-1)\mathbf{X}(n)}\right). \end{aligned} \quad (10.69)$$

Since $\delta\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n-1)$ is assumed to be small, we have ignored terms involving second- and higher-order products of its elements from the last expression in (10.69). Substituting (10.69) in the expression for $\mathbf{k}(n)$, we get the following result describing the perturbed gain vector $\underline{\mathbf{k}}(n)$:

$$\begin{aligned}\underline{\mathbf{k}}(n) &= \underline{r}(n)\underline{\mathbf{S}}^T(n) \\ &= r(n)\left(\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n-1) + \delta\hat{\mathbf{R}}_{\mathbf{xx}}(n-1)\right)^T \mathbf{X}(n) - r^2(n)\mathbf{X}^T(n)\delta\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n-1)\mathbf{X}(n). \end{aligned} \quad (10.70)$$

We can derive an expression for the error in calculating $\mathbf{k}(n)$ by subtracting $\mathbf{k}(n) = r(n)\mathbf{S}^T(n)$ from both sides of the above equation. This operation results in

$$\begin{aligned}\delta\mathbf{k}(n) &= r(n)\left(\delta\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n-1)\right)^T \mathbf{X}(n) \\ &\quad - r^2(n)\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n-1)\mathbf{X}(n)\mathbf{X}^T(n)\delta\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n-1)\mathbf{X}(n). \end{aligned} \quad (10.71)$$

Substituting (10.71) in the expression for computing $\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n)$ gives

$$\begin{aligned}\underline{\hat{\mathbf{R}}}_{\mathbf{xx}}^{-1}(n) &= \frac{1}{\lambda}\left(\left(\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n-1) + \delta\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n-1)\right) - \left(\mathbf{k}(n) + \delta\mathbf{k}(n)\right)\mathbf{X}^T(n)\left(\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n-1)\right.\right. \\ &\quad \left.\left. + \delta\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n-1)\right)\right). \end{aligned} \quad (10.72)$$

Subtracting $\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n)$ from both sides of the above equation after substituting for $\delta\mathbf{k}(n)$ from (10.71) and discarding terms involving products of two or more elements of the error matrix and the error vector results in the desired expression characterizing the propagation of $\delta\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n-1)$ to the next iteration as

$$\begin{aligned}\delta\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n) &= \frac{1}{\lambda}\left(\delta\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n-1) - \mathbf{k}(n)\mathbf{X}^T(n)\delta\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n-1)\right. \\ &- r(n)\left(\delta\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n-1)\right)^T \mathbf{X}(n)\mathbf{X}^T(n)\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n-1) \\ &+ \left. r^2(n)\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n-1)\mathbf{X}(n)\mathbf{X}^T(n)\delta\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n-1)\mathbf{X}(n)\mathbf{X}^T(n)\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n-1)\right). \end{aligned} \quad (10.73)$$

In order to proceed, we consider the symmetric and anti-symmetric parts of $\delta\hat{\mathbf{R}}_{\mathbf{XX}}^{-1}(n)$ separately. Averaging both sides of (10.73) with their respective matrix transposes and simplifying gives the symmetric part of $\delta\hat{\mathbf{R}}_{\mathbf{XX}}^{-1}(n)$ to be

$$\begin{aligned}\left[\delta\hat{\mathbf{R}}_{\mathbf{XX}}^{-1}(n)\right]_s &= \frac{1}{\lambda}\left(\mathbf{I}-\mathbf{k}(n)\mathbf{X}^T(n)\right)\left[\delta\hat{\mathbf{R}}_{\mathbf{XX}}^{-1}(n-1)\right]_s\left(\mathbf{I}-\mathbf{k}(n)\mathbf{X}^T(n)\right)^T \\ &\quad -\mathbf{k}(n)\mathbf{X}^T(n)\left[\delta\hat{\mathbf{R}}_{\mathbf{XX}}^{-1}(n-1)\right]_a+\left[\delta\hat{\mathbf{R}}_{\mathbf{XX}}^{-T}(n-1)\right]_a\mathbf{X}(n)\mathbf{k}^T(n)\end{aligned} \tag{10.74}$$

Similarly, the anti-symmetric part can be shown to be

$$\left[\delta\hat{\mathbf{R}}_{\mathbf{XX}}^{-1}(n)\right]_a = \frac{1}{\lambda}\left[\delta\hat{\mathbf{R}}_{\mathbf{XX}}^{-1}(n-1)\right]_a+\mathbf{k}(n)\mathbf{X}^T(n)\left[\delta\hat{\mathbf{R}}_{\mathbf{XX}}^{-1}(n-1)\right]_a\mathbf{X}(n)\mathbf{k}^T(n). \tag{10.75}$$

In the above equations, the symmetric and anti-symmetric parts of a matrix $\mathbf{A}$ are defined as

$$[\mathbf{A}]_s = \frac{1}{2}\left[\mathbf{A}+\mathbf{A}^T\right] \tag{10.76}$$

and

$$[\mathbf{A}]_a = \frac{1}{2}\left[\mathbf{A}-\mathbf{A}^T\right], \tag{10.77}$$

respectively.

**Propagation of the Antisymmetric Part of the Perturbation**

Since $\left[\delta\hat{\mathbf{R}}_{\mathbf{XX}}^{-1}(n-1)\right]_a$ is an anti-symmetric matrix,

$$\mathbf{X}^T(n)\left[\delta\hat{\mathbf{R}}_{\mathbf{XX}}^{-1}(n-1)\right]_a\mathbf{X}(n) = 0 \tag{10.78}$$

for all choices of $\mathbf{X}(n)$. Consequently, the anti-symmetric part of the perturbation is propagated as

$$\left[\delta\hat{\mathbf{R}}_{\mathbf{XX}}^{-1}(n)\right]_a = \frac{1}{\lambda}\left[\delta\hat{\mathbf{R}}_{\mathbf{XX}}^{-1}(n-1)\right]_a, \tag{10.79}$$

implying that the elements of the anti-symmetric part of the perturbation matrix diverge exponentially for $\lambda < 1$.

**Propagation of the Symmetric Part of the Perturbation**

Equation (10.74) shows that the symmetric part of the propagated error depends on the antisymmetric part. Since the antisymmetric perturbations diverge for $\lambda < 1$, it is clear that the the symmetric part also diverges whenever there is an antisymmetric component in the perturbation. To understand how the symmetric part of the perturbation propagates in time in the absence of an antisymmetric component, we have to characterize the matrix $\left(\mathbf{I} - \mathbf{k}(n)\mathbf{X}^T(n)\right)$that appears in (10.74). Since $\mathbf{k}(n) = \hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n)\mathbf{X}(n)$ and $\hat{\mathbf{R}}_{\mathbf{xx}}(n) = \lambda\hat{\mathbf{R}}_{\mathbf{xx}}(n-1) + \mathbf{X}(n)\mathbf{X}^T(n)$, we can see that

$$\begin{aligned}\mathbf{I} - \mathbf{k}(n)\mathbf{X}^T(n) &= \mathbf{I} - \hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n)\mathbf{X}(n)\mathbf{X}^T(n) \\ &= \hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n)\left[\hat{\mathbf{R}}_{\mathbf{xx}}(n) - \mathbf{X}(n)\mathbf{X}^T(n)\right] \\ &= \lambda\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n)\hat{\mathbf{R}}_{\mathbf{xx}}(n-1) \end{aligned} \tag{10.80}$$

Substituting this result in (10.74) gives

$$\left[\delta\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n)\right]_s = \lambda\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n)\hat{\mathbf{R}}_{\mathbf{xx}}(n-1)\left[\delta\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n-1)\right]_s\hat{\mathbf{R}}_{\mathbf{xx}}(n-1)\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n). \tag{10.81}$$

Assuming that $\hat{\mathbf{R}}_{\mathbf{xx}}(n)$ is not singular for any value of $n$, and that its eigenvalues are bounded from above and below by finite, non-zero values, we can show that the symmetric part of the perturbation propagates after $N$ iterations as

$$\left[\delta\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n+N-1)\right]_s = \lambda^N\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n+N-1)\hat{\mathbf{R}}_{\mathbf{xx}}(n-1)\left[\delta\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n-1)\right]_s\hat{\mathbf{R}}_{\mathbf{xx}}(n-1)\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n+N-1). \tag{10.82}$$

This result implies that the symmetric part of the error decays to zero exponentially fast for $\lambda < 1$ when the perturbation does not contain an antisymmetric component.

Combining the results for the behavior of the symmetric and anti-symmetric parts, we see that in order for the conventional RLS adaptive filter to operate in a stable manner, it is essential that the inverse of the least-squares autocorrelation matrix is constrained to be symmetric, thereby not allowing the propagation of non-zero, anti-symmetric components of any errors introduced into its calculation. The second realization CRLS2 meets this objective.

### Error Propagation in CRLS2

An analysis similar to that done previously for the CRLS1 realization will show that

$$\delta\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n) = \frac{1}{\lambda}\left(\mathbf{I} - \mathbf{k}(n)\mathbf{X}^T(n)\right)\delta\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n-1)\left(\mathbf{I} - \mathbf{k}(n)\mathbf{X}^T(n)\right), \tag{10.83}$$

implying that the perturbations decay exponentially fast for $\lambda < 1$. The details of the derivation are left as an exercise for the reader. The MATLAB program of Table 10.2 implements the CRLS2 realization.

### A Simple Method to Guarantee Symmetry of $\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n)$.

A comparison of the computational complexities of the CRLS1 and CRLS2 realizations shows that the CRLS2 realization is slightly more complex. The simpliest method to ensure the symmetry of $\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n)$ is to compute only the upper (or lower) triangular part of the matrix and then copy the elements to the locations of symmetry.

### Numerical Error Propagation for Systems with $\lambda = 1$

The analysis we just completed indicates that single errors do not grow or decay significantly for $\lambda = 1$. While this is true, numerical errors introduced during each iteration add to those introduced during previous iterations. This phenomenon causes a linear growth in the numerical error in $\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n)$ with time, and eventually causing the estimate to be useless. All variations of the conventional RLS adaptive filter, including the QR-decomposition-based method discussed later in this chapter, suffer from numerical instability when $\lambda$ is chosen to be one. Problems caused by this are usually avoided in practice by periodically resetting the inverse autocorrelation matrix to a diagonal matrix with small entries.

**Example 10.4: Propagation of Errors in Conventional RLS Adaptive Filters**

This example demonstrates some of the analytical results obtained in this section. We consider the system identification problem of Example 10.1, with $\epsilon = 0.01$. The CRLS1 and CRLS2 realizations were implemented for a two-coefficient system with the above input signal and initialized using

$$\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(0) = \begin{bmatrix} 10^3 & 10^{-6} \\ 0 & 10^3 \end{bmatrix}.$$

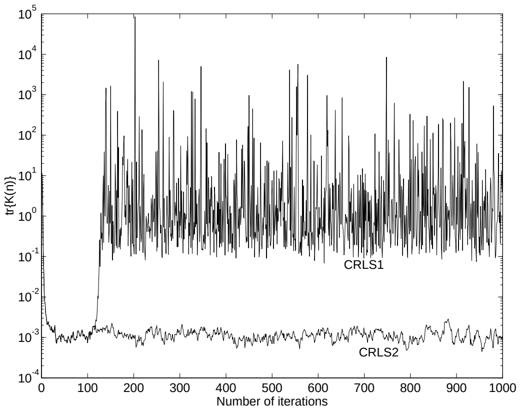
Our analysis indicates that the CRLS1 algorithm will not work in a reliable manner since $\hat{\mathbf{R}}_{\mathbf{xx}}^{-1}(n)$ is not initialized to a symmetric matrix. Figure 10.11 displays the sum of the squared deviation of the coefficients of the two realizations from their optimal values for $\lambda$ =0.9, 0.99 and 0.999. As predicted, the CRLS1 realization gives meaningless results after a short period of time. As would be expected from the theory, the explosive behavior of the CRLS1 realization takes longer to occur for smaller values of $1 - \lambda$. The CRLS2 realization operates in a stable manner for all the experiments shown here. This behavior is also as predicted by our analysis.

## 10.5 QR-Decomposition-Based RLS Adaptive Filters

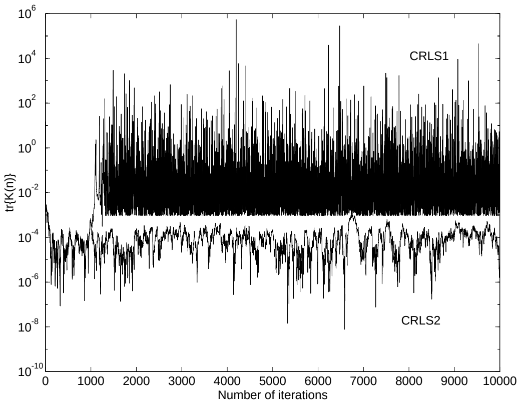
In this section, we derive a different realization of the RLS adaptive filter based on the QR-decomposition (QRD) of a matrix formed from the input signal. Even though we have seen that the conventional RLS adaptive filter possesses good convergence and tracking properties when implemented in a numerically stable manner, there are several reasons why QRD-based RLS adaptive filters are attractive in practical applications.

1. The QRD-based algorithms find the same solution as the conventional RLS adaptive filter. Consequently, they share the good convergence and tracking properties of the conventional RLS algorithm.

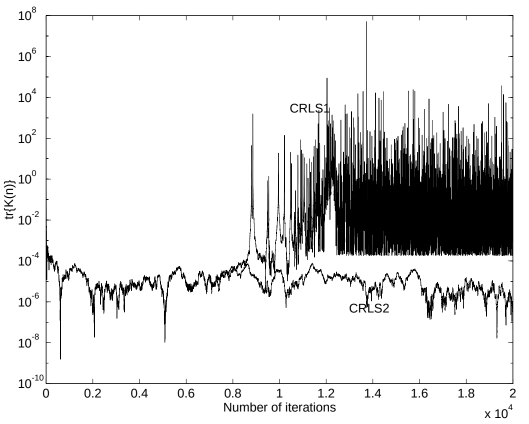
2. The numerical properties of the conventional RLS adaptive filter depend on the condition number of the least-squares autocorrelation matrix $\hat{\mathbf{R}}_{\mathbf{xx}}(n)$. The numerical



(a) $\lambda = 0.9$



(b) $\lambda = 0.99$



(c) $\lambda = 0.999$

Figure 10.11: Propagation of asymmetric errors in the least-squares autocorrelation matrix in the conventional RLS adaptive filter.

properties of the QRD-based methods, on the other hand, depend on the *Cholesky factor* of the autocorrelation matrix. It can be shown that the condition number of the Cholesky factor of a matrix is the square root of the condition number of the matrix. This fact means that we can expect the QRD-based algorithms to have better numerical properties than the conventional RLS adaptive filter.

3. QRD-based algorithms can be implemented using a series of identical transformations known as the *Givens rotations.* This fact is attractive from the point of view of hardware and software implementation, since the adaptive filter can be implemented using identical computational procedures or hardware components that implement the Givens rotations.

### 10.5.1 The Problem Formulation

Let us form an input signal matrix

$$\mathcal{X}(n) = \begin{bmatrix} \lambda^{(n-1)/2}\mathbf{X}^T(1) \\ \lambda^{(n-2)/2}\mathbf{X}^T(2) \\ \vdots \\ \mathbf{X}^T(n) \end{bmatrix} \tag{10.84}$$

and a desired response signal vector

$$\mathcal{D}(n) = \begin{bmatrix} \lambda^{(n-1)/2}d(1) & \lambda^{(n-2)/2}d(2) & \cdots & d(n) \end{bmatrix}^T . \tag{10.85}$$

Both of the above quantities have $n$ rows at time $n$, and therefore the size of the input signal matrix and the desired response signal vector increases with time. The above matrix and vector are related to the corresponding quantities at the previous time instant as

$$\mathcal{X}(n) = \begin{bmatrix} \sqrt{\lambda}\mathcal{X}(n-1) \\ \mathbf{X}^T(n) \end{bmatrix} \tag{10.86}$$

and

$$\mathcal{D}(n) = \begin{bmatrix} \sqrt{\lambda}\mathcal{D}(n-1) \\ d(n) \end{bmatrix}, \tag{10.87}$$

respectively. The exponentially-weighted least-squares autocorrelation matrix and cross-correlation vector in (10.23) and (10.24) can be found from $\mathcal{X}(n)$ and $\mathcal{D}(n)$ as

$$\hat{\mathbf{R}}_{\mathbf{xx}}(n) = \mathcal{X}^T(n)\mathcal{X}(n) \tag{10.88}$$

and

$$\hat{\mathbf{P}}_{\mathbf{x}\mathrm{d}}(n) = \mathcal{X}^T(n)\mathcal{D}(n), \tag{10.89}$$

respectively.

The exponentially weighted least-squares adaptive filtering problem can now be reformulated as that of minimizing the squared norm of the estimation error vector given by

$$\begin{aligned} J(n) &= \sum_{k=1}^{n} \lambda^{n-k} \Big(d(k) - \mathbf{W}^T(n)\mathbf{X}(k)\Big)^2 \\ &= \| \mathcal{D}(n) - \mathcal{X}(n)\mathbf{W}(n) \|^2, \end{aligned} \tag{10.90}$$

where $\| (\cdot) \|^2$ denotes the Euclidean norm of the vector $(\cdot)$. Rather than directly find the solution for $\mathbf{W}(n)$, we consider a transformation of the error vector that does not change its length. Let $\mathbf{Q}(n)$ be a $n \times n$-element, orthonormal matrix. Since $\mathbf{Q}^T(n)\mathbf{Q}(n) = \mathbf{I}$, $J(n)$ can also be expressed as

$$\begin{aligned} J(n) &= \| \mathcal{D}(n) - \mathcal{X}(n)\mathbf{W}(n) \|^2 \\ &= \| \mathbf{Q}(n)\left(\mathcal{D}(n) - \mathcal{X}(n)\mathbf{W}(n)\right) \|^2 . \end{aligned} \tag{10.91}$$

The key idea in the development of QRD-based RLS adaptive filters is to choose $\mathbf{Q}(n)$ such that

$$\mathbf{Q}(n)\mathcal{X}(n) = \begin{bmatrix} \mathbf{R}(n) \\ \mathbf{0} \end{bmatrix}, \tag{10.92}$$

where $\mathbf{R}(n)$ is an $L \times L$-element upper triangular matrix and $\mathbf{0}$ is a matrix with all zero elements and appropriate dimensions. The above decomposition of $\mathcal{X}(n)$ into an orthonormal matrix multiplied by an upper triangular matrix is known as the *QR-decomposition* of $\mathcal{X}(n)$. It is left as an exercise for the reader to show that

$$\hat{\mathbf{R}}_{\mathbf{xx}}(n) = \mathbf{R}^T(n)\mathbf{R}(n), \tag{10.93}$$

which implies that $\mathbf{R}(n)$ is indeed a Cholesky or square root factor of $\hat{\mathbf{R}}_{\mathbf{xx}}(n)$.

### Calculation of the Optimal Coefficient Vector

Let

$$\mathbf{Q}(n)\mathcal{D}(n) = \begin{bmatrix} \mathbf{U}_d(n) \\ \mathbf{V}_d(n) \end{bmatrix}, \tag{10.94}$$

where $\mathbf{U}_d(n)$ contains $L$ elements and $\mathbf{V}_d(n)$ contains $n - L$ elements. We can now rewrite the least-squares cost function in (10.91) using (10.94) and (10.92) as

$$\begin{aligned} J(n) &= \left\| \begin{bmatrix} \mathbf{U}_d(n) \\ \mathbf{V}_d(n) \end{bmatrix} - \begin{bmatrix} \mathbf{R}(n)\mathbf{W}(n) \\ \mathbf{0} \end{bmatrix} \right\|^2 \\ &= \| \mathbf{U}_d(n) - \mathbf{R}(n)\mathbf{W}(n) \|^2 + \| \mathbf{V}_d(n) \|^2 . \end{aligned} \tag{10.95}$$

Since $\mathbf{V}_d(n)$ does not depend on the coefficient vector, the above cost function can be minimized by choosing $\mathbf{W}(n)$ such that the first term on the right-hand-side of (10.95) is zero. Thus, the solution to the least-squares estimation problem is given by

$$\mathbf{W}(n) = \mathbf{R}^{-1}(n)\mathbf{U}_d(n). \tag{10.96}$$

Furthermore, the least-squares value of the estimation error is given by

$$J_{min}(n) = \| \mathbf{V}_d(n) \|^2 . \tag{10.97}$$

Since $\mathbf{R}(n)$ is an upper triangular matrix, we can solve for the coefficients using back-substitution. Let the $(i,j)$th element of $\mathbf{R}(n)$ be $r_{i,j}(n)$, and let the $j$th element of $\mathbf{U}_d(n)$ be $u_j(n)$. Then, we first compute $w_{L-1}(n)$ as

$$w_{L-1}(n) = \frac{u_{L-1}(n)}{r_{L-1,L-1}(n)}. \tag{10.98}$$

The rest of the coefficients are computed in the reversed order of the index as

$$w_i(n) = \frac{u_i(n) - \sum\limits_{j=i+1}^{L-1} r_{i,j}(n)w_j(n)}{r_{i,i}(n)} \quad ; i = L-2,\ L-3,\ \cdots,\ 0. \tag{10.99}$$

### 10.5.2 Recursive Selection of $\mathbf{Q}(n)$

The idea described above does not constitute a realizable adaptive filter since the complexity of finding an orthonormal matrix $\mathbf{Q}(n)$ that triangularizes $\mathcal{X}(n)$ at each time increases with $n$, making it impossible to perform the triangularization for all but a few time samples. We now describe a method for triangularizing the matrix $\mathcal{X}(n)$ recursively.

Assume that at time $n-1$, the Cholesky factor $\mathbf{R}(n-1)$ as well as the vectors $\mathbf{U}_d(n-1)$ and $\mathbf{V}_d(n-1)$ have been evaluated. Let $\mathbf{Q}(n-1)$ represent the orthonormal matrix that triangularizes $\mathcal{X}(n-1)$. Consider pre-multiplying $\mathcal{X}(n)$ with an augmented version of $\mathbf{Q}(n-1)$ obtained by adding an additional row and column to the matrix as follows:

$$\begin{aligned}
\begin{bmatrix} \mathbf{Q}(n-1) & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \mathcal{X}(n) &= \begin{bmatrix} \mathbf{Q}(n-1) & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} \sqrt{\lambda}\mathcal{X}(n-1) \\ \mathbf{X}^T(n) \end{bmatrix} \\
&= \begin{bmatrix} \sqrt{\lambda}\mathbf{R}(n-1) \\ \mathbf{0} \\ \mathbf{X}^T(n) \end{bmatrix}.
\end{aligned} \tag{10.100}$$

In the above equation, we have used the same notation $\mathbf{0}$ to denote zero matrices and vectors of various orders. The sizes of these matrices and vectors should be evident from the context. In a similar manner to the calculations in (10.100), we operate on $\mathcal{D}(n)$ with the same augmented matrix to obtain

$$\begin{aligned}\begin{bmatrix} \mathbf{Q}(n-1) & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix}\mathcal{D}(n) &= \begin{bmatrix} \mathbf{Q}(n-1) & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix}\begin{bmatrix} \sqrt{\lambda}\mathcal{D}(n-1) \\ d(n) \end{bmatrix} \\ &= \begin{bmatrix} \sqrt{\lambda}\mathbf{U}_d(n-1) \\ \sqrt{\lambda}\mathbf{V}_d(n-1) \\ d(n) \end{bmatrix}. \end{aligned} \tag{10.101}$$

The two equations above provide the means for efficiently implementing the RLS adaptive filter using QR-decomposition. A careful consideration of (10.100) shows that the upper triangular matrix $\mathbf{R}(n)$ can be constructed by a transformation that makes the last row zero of the matrix on the right-hand-side of (10.100) without affecting those rows that are already zero. The transformation we seek is such that it does not change the $(L+1)$ through $(n-1)$th rows of any matrix or vector it operates upon. The same transformation, when applied to the vector $[\sqrt{\lambda}\mathbf{U}_d(n-1)\ \ \sqrt{\lambda}\mathbf{V}_d(n-1)\ \ d(n)]^T$, does not change $\sqrt{\lambda}\mathbf{V}_d(n-1)$, but transforms $\sqrt{\lambda}\mathbf{U}_d(n-1)$ and $d(n)$ to generate $\mathbf{U}_d(n)$. Let $\hat{\mathbf{Q}}(n)$ denote an $(L+1)\times(L+1)$-element orthonormal matrix such that

$$\hat{\mathbf{Q}}(n)\begin{bmatrix} \sqrt{\lambda}\mathbf{R}(n-1) \\ \mathbf{X}^T(n) \end{bmatrix} = \begin{bmatrix} \mathbf{R}(n) \\ \mathbf{0} \end{bmatrix}. \tag{10.102}$$

Then, $\mathbf{U}_d(n)$ is obtained by operating on $[\sqrt{\lambda}\mathbf{U}_d(n-1)\ \ d(n)]^T$ with the same matrix, *i.e.*,

$$\hat{\mathbf{Q}}(n)\begin{bmatrix} \sqrt{\lambda}\mathbf{U}_d(n-1) \\ d(n) \end{bmatrix} = \begin{bmatrix} \mathbf{U}_d(n) \\ \tilde{e}(n) \end{bmatrix}, \tag{10.103}$$

where $\tilde{e}(n)$ is the last element of the transformed vector. The reader should verify that

$$\mathbf{V}_d(n) = \begin{bmatrix} \sqrt{\lambda}\mathbf{V}_d(n-1) \\ \tilde{e}(n) \end{bmatrix} \tag{10.104}$$

and that the least-squares value of the estimation error can be recursively computed as

$$J_{min}(n) = \lambda J_{min}(n-1) + \tilde{e}^2(n). \tag{10.105}$$

Note that there is no need to evaluate $\mathbf{V}_d(n)$ explicitly for estimating the coefficients or for updating the least-squares value of the estimation error. Furthermore, the size of the matrices and vectors involved in the recursions do not change as $n$ increases. Therefore, the method described above is a realizable algorithm.

### 10.5.3 Computation of $\hat{\mathbf{Q}}(n)$

The objective of the matrix $\hat{\mathbf{Q}}(n)$ is to modify the matrix $[\sqrt{\lambda}\mathbf{R}^T(n-1)\ \ \mathbf{X}(n)]^T$ it pre-multiplies in (10.102) by making it upper triangular. We show now that the matrix multiplication described above can be accomplished by a series of $L$ Givens rotations.

## Givens Rotation

Consider a two-element vector of the form $[x \quad y]^T$. We wish to find an orthonormal matrix that operates on this vector in such a manner as to make the second element of the transformed vector zero. One choice for such a matrix is

$$\hat{\mathbf{Q}} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}, \tag{10.106}$$

where

$$\cos\theta = \frac{x}{\sqrt{x^2+y^2}} \tag{10.107}$$

and

$$\sin\theta = \frac{y}{\sqrt{x^2+y^2}}. \tag{10.108}$$

The transformation using this matrix results in

$$\hat{\mathbf{Q}} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \sqrt{x^2+y^2} \\ 0 \end{bmatrix}. \tag{10.109}$$

The above transformation is said to *annihilate* $y$ by *rotating it into* $x$. The transformation itself is known as *Givens rotation*, named after its inventor [Givens 1958]. The variable $\theta$ is known as the *angle of rotation*.

The adaptation of the above idea to our problem is straightforward. In our problem, we require a sequence of $L$ Givens rotations. The first rotation annihilates $x(n)$ by rotating it into $\sqrt{\lambda}r_{0,0}(n-1)$. This transformation is performed by an $(L+1)\times(L+1)$-element matrix of the form

$$\hat{\mathbf{Q}}_1(n) = \begin{bmatrix} \cos\theta_1(n) & \mathbf{0} & \sin\theta_1(n) \\ \mathbf{0} & \mathbf{I} & \mathbf{0} \\ -\sin\theta_1(n) & \mathbf{0} & \cos\theta_1(n) \end{bmatrix}, \tag{10.110}$$

where

$$\cos\theta_1(n) = \frac{\sqrt{\lambda}r_{0,0}(n-1)}{\sqrt{\lambda r_{0,0}^2(n-1)+x^2(n)}} \tag{10.111}$$

and

$$\sin\theta_1(n) = \frac{x(n)}{\sqrt{\lambda r_{0,0}^2(n-1)+x^2(n)}}. \tag{10.112}$$

Rotating $x(n)$ into $\sqrt{\lambda}r_{0,0}(n-1)$ using this matrix changes all the elements in the first row of $\sqrt{\lambda}\mathbf{R}(n-1)$ and the most recent input vector $\mathbf{X}(n)$. Obviously, the first element of $\mathbf{X}(n)$ becomes zero in the process. Let $\mathbf{X}^{(i-1)}(n)$ represent the transformed input vector after $i-1$ rotations. The first $i-1$ elements of this vector are zero. Let the $j$th element of this vector be $x_j^{(i-1)}(n)$. Then the $i$th rotation matrix $\hat{\mathbf{Q}}_i(n)$ is given by the $(L+1)\times(L+1)$-element matrix

$$\hat{\mathbf{Q}}_i(n) = \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \cos\theta_i(n) & \mathbf{0} & \sin\theta_i(n) \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & -\sin\theta_i(n) & \mathbf{0} & \cos\theta_i(n) \end{bmatrix}. \tag{10.113}$$

The identity matrices on the top left corner and the third row of the matrix above contain $(i-1)\times(i-1)$ and $(L-i)\times(L-i)$ elements, respectively. The number of elements in the zero matrices and vectors denoted by $\mathbf{0}$ depends on their location. The angle of rotation $\theta_i(n)$ is defined by

$$\cos\theta_i(n) = \frac{\sqrt{\lambda} r_{i-1,i-1}(n-1)}{\sqrt{\lambda r_{i-1,i-1}^2(n-1) + \left(x_i^{(i-1)}(n)\right)^2}} \tag{10.114}$$

and

$$\sin\theta_i(n) = \frac{x_i^{(i-1)}(n)}{\sqrt{\lambda r_{i-1,i-1}^2(n-1) + \left(x_i^{(i-1)}(n)\right)^2}}. \tag{10.115}$$

The sequence of $L$ rotations described above corresponds to the overall rotation matrix $\hat{\mathbf{Q}}(n)$ which is given by

$$\hat{\mathbf{Q}}(n) = \hat{\mathbf{Q}}_L(n) \ \cdots \ \hat{\mathbf{Q}}_2(n) \ \hat{\mathbf{Q}}_1(n). \tag{10.116}$$

Applying these rotations in the same order to the vector $[\sqrt{\lambda}\mathbf{U}_d^T(n-1) \ \ d(n)]^T$ results in the new vector $[\mathbf{U}_d^T(n) \ \ \tilde{e}(n)]^T$. The new set of coefficients are then obtained from $\mathbf{R}(n)$ and $\mathbf{U}_d(n)$ by back-substitution.

The QRD-based exponentially-weighted RLS adaptive filter is given in Table 10.5. It is not difficult to see that the computational complexity of this adaptive filter is proportional to $L^2$ arithmetical operations per iteration. Thus, this algorithm's complexity is comparable to that of the conventional RLS adaptive filter. A MATLAB program to implement this adaptive filter is given in Table 10.6.

### 10.5.4 Evaluation of Estimation Error Without Back-Substitution

In many applications such as equalization, noise cancellation and line enhancement, knowledge of the direct form coefficients is not necessary. Only the estimate of the desired response signal or the corresponding estimation error signal is needed in these applications. In such situations, it is possible to evaluate the estimation error without explicitly calculating $\mathbf{W}(n)$ in the QR-RLS adaptive filter. We develop this form of the algorithm in what follows.

#### Another Expression for $\mathbf{U}_d(n)$

Recall from (10.96) that

$$\mathbf{U}_d(n) = \mathbf{R}(n)\mathbf{W}(n). \tag{10.117}$$

Table 10.5: QR-decomposition-based RLS adaptive filter

**Initialization**

$$\begin{aligned}\mathbf{R}(0) &= \sqrt{\delta}\mathbf{I}\\ \mathbf{U}_d(0) &= \mathbf{0}\\ \mathbf{W}(0) &= \mathbf{0}\ ;\ \mathbf{W}(n) \text{ can be initialized arbitrarily if necessary}\end{aligned}$$

**Recursion**

Given $\mathbf{R}(n-1), \mathbf{U}_d(n-1)\mathbf{W}(n-1), \mathbf{X}(n)$ and $d(n)$, compute at time $n$:

$$\begin{aligned}\mathbf{X}^{(0)}(n) &= \mathbf{X}(n)\\ \mathbf{R}^{(0)}(n) &= \sqrt{\lambda}\mathbf{R}(n-1)\\ \mathbf{U}_d^{(0)}(n) &= \sqrt{\lambda}\mathbf{U}_d(n-1)\\ \tilde{e}^{(0)}(n) &= d(n)\end{aligned}$$

For $i = 1, 2, \cdots,$ L,

$$\begin{aligned}\cos\theta_i(n) &= \frac{\sqrt{\lambda}r_{i-1,i-1}^{(i-1)}(n-1)}{\sqrt{\lambda\left(r_{i-1,i-1}^{(i-1)}(n-1)\right)^2 + \left(x_{i-1}^{(i-1)}(n)\right)^2}}\\ \sin\theta_i(n) &= \frac{x_{i-1}^{(i-1)}(n)}{\sqrt{\left(\lambda r_{i-1,i-1}^{(i-1)}(n-1)\right)^2 + \left(x_{i}^{(i-1)}(n)\right)^2}}\\ \hat{\mathbf{Q}}_i(n) &= \begin{bmatrix}\mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \cos\theta_i(n) & \mathbf{0} & \sin\theta_i(n)\\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0}\\ \mathbf{0} & -\sin\theta_i(n) & \mathbf{0} & \cos\theta_i(n)\end{bmatrix}\\ \begin{bmatrix}\mathbf{R}^{(i)}(n) & \mathbf{U}_d^{(i)}(n)\\ \left(\mathbf{X}^{(i)}(n)\right)^T & \tilde{e}^{(i)}(n)\end{bmatrix} &= \hat{\mathbf{Q}}_i(n)\begin{bmatrix}\mathbf{R}^{(i-1)}(n) & \mathbf{U}_d^{(i-1)}(n)\\ \left(\mathbf{X}^{(i-1)}(n)\right)^T & \tilde{e}^{(i-1)}(n)\end{bmatrix}\end{aligned}$$

Solve for the coefficients after the $L$th rotation

$$\begin{aligned}\mathbf{R}(n) &= \mathbf{R}^{(L)}(n)\\ \mathbf{U}_d(n) &= \mathbf{U}_d^{(L)}(n)\\ w_{L-1}(n) &= \frac{u_{L-1}(n)}{r_{L-1,L-1}(n)}\\ w_i(n) &= \frac{u_i(n) - \sum_{j=i+1}^{L-1} r_{i,j}(n)w_j(n)}{r_{i,i}(n)} \quad ; i = L-2,\ L-3,\ \cdots,\ 0\\ e(n) &= d(n) - \sum_{i=0}^{L-1} w_i(n)x(n-i)\end{aligned}$$

Table 10.6: MATLAB program to implement the QR-RLS adaptive filter with back-substitution.

```
function [W,dhat,e] = fir_qrrls1(lambda,delta,L,x,d);

%  This function adapts a finite-impulse-response (FIR)
%  filter using the QR-Decomposition-based recursive least-squares (QR-RLS)
%  adaptive filter.
%
%  Input parameters:
%      lambda   = forgetting factor
%      delta    = initialization constant for autocorrelation matrix
%      L        = number of coefficients
%      x        = input signal (num_iter x 1)
%      d        = desired response signal (num_iter x 1)
%
%  Output of program:
%      W        = Evolution of coefficients (L x (num_iter + 1))
%      dhat     = output of adaptive filter (num_iter x 1)
%      e        = error of adaptive filter (num_iter x 1)
lambda2 = sqrt(lambda);
start_iter = 1;
end_iter = min([length(x) length(d)]);
R = diag(ones(L,1)*sqrt(delta));
U = zeros(L,1);
X = zeros(L,1);
W = zeros(L,end_iter);
e = zeros(end_iter,1);
dhat = zeros(end_iter,1);

for n = start_iter:end_iter,

 X = [x(n);X(1:L-1)];
 R = lambda2*R;
 U = lambda2*U;
 X1 = X;
 etilde = d(n);
         for i=1:L,
               den = sqrt(R(i,i)^2 + X1(i)^2);
               c = R(i,i)/den;
               s = X1(i)/den;
               Q = [c, s; -s, c];
```

```
            R2 = Q*[R(i,i:L);X1(i:L)'];
            R(i,i:L) = R2(1,:);
            X1(i:L) = R2(2,:);
            U2 = Q*[U(i);etilde];
            U(i) = U2(1);
            etilde = U2(2);
        end;
        W1(L) = U(L)/R(L,L);
        for i=L-1:-1:1,
            W1(i) = (U(i) - sum(R(i,i+1:L).*W1(i+1:L)))/R(i,i);
        end;
        W(:,n) = W1';
        dhat(n) = W1*X;
        e(n) = d(n) - W1*X;
end;
```

Since

$$\begin{aligned}\mathbf{W}(n) &= \hat{\mathbf{R}}_{\mathbf{XX}}^{-1}(n)\hat{\mathbf{P}}_{\mathbf{X}\mathrm{d}}(n)\\ &= \left(\mathbf{R}^{-1}(n)\mathbf{R}^{-T}(n)\right)\hat{\mathbf{P}}_{\mathbf{X}\mathrm{d}}(n),\end{aligned} \tag{10.118}$$

we can combine the two results to obtain a new expression for $\mathbf{U}_d(n)$ as

$$\mathbf{U}_d(n) = \mathbf{R}^{-T}(n)\hat{\mathbf{P}}_{\mathbf{X}\mathrm{d}}(n). \tag{10.119}$$

Let us now define a new vector $\mathbf{Z}(n)$ as

$$\mathbf{Z}(n) = \mathbf{R}^{-T}(n)\mathbf{X}(n). \tag{10.120}$$

It immediately follows that

$$\begin{aligned}\mathbf{U}_d^T(n)\mathbf{Z}(n) &= \hat{\mathbf{P}}_{\mathbf{X}\mathrm{d}}(n)\mathbf{R}^{-1}(n)\mathbf{R}^{-T}(n)\mathbf{X}(n)\\ &= \mathbf{W}^T(n)\mathbf{X}(n)\\ &= \hat{d}(n).\end{aligned} \tag{10.121}$$

The key to finding $\hat{d}(n)$ directly without explicitly calculating $\mathbf{W}$ is that of efficiently updating the vector $\mathbf{Z}(n)$.

### Updating $\mathbf{Z}(n)$

To find an approach for updating $\mathbf{Z}(n)$, we relate this vector to the gain vector $\mathbf{k}(n)$. We saw in Section 10.2.2 that $\mathbf{k}(n)$ is the vector of coefficients that estimates $\pi_n(k)$ in (10.41) using $\mathbf{X}(n)$. Since

$$\mathbf{X}(n) = \sum_{k=1}^{n} \lambda^{n-k}\mathbf{X}(k)\pi_n(k), \tag{10.122}$$

we infer that $\mathbf{Z}(n)$ must have the same role in the estimation of $\pi_n(k)$ that $\mathbf{U}_d(n)$ has in the estimation of $d(n)$, since $\mathbf{U}_d(n)$ is updated by rotating $d(n)$ into $\sqrt{\lambda}\mathbf{U}_d(n-1)$. Since $\pi_n(k)$ is zero for all values of $k$ except $k = n$, we conjecture that $\mathbf{Z}(n)$ can be obtained by rotating the value one into a vector of zeroes as

$$\hat{\mathbf{Q}}(n)\left[\begin{array}{c} \mathbf{0} \\ 1 \end{array}\right] = \left[\begin{array}{c} \mathbf{Z}(n) \\ \tilde{\gamma}(n) \end{array}\right], \tag{10.123}$$

where $\tilde{\gamma}(n)$ is the $(L, L)$th element of $\hat{\mathbf{Q}}(n)$.

To verify the conjecture we made, we consider the set of all rotations together arranged in the following block matrix form:

$$\hat{\mathbf{Q}}(n)\left[\begin{array}{ccc} \sqrt{\lambda}\mathbf{R}(n-1) & \sqrt{\lambda}\mathbf{U}_d(n-1) & 0 \\ \mathbf{X}^T(n) & d(n) & 1 \end{array}\right] = \left[\begin{array}{ccc} \mathbf{R}(n) & \mathbf{U}_d(n) & \mathbf{Z}(n) \\ \mathbf{0} & \tilde{e}(n) & \tilde{\gamma}(n) \end{array}\right]. \tag{10.124}$$

To understand the significance of the variables $\tilde{e}(n)$, $\tilde{\gamma}(n)$ and $\mathbf{Z}(n)$, we pre-multiply both sides of the above equation with their respective transposes and equate the corresponding terms on both sides. This operation results in

$$\left[\begin{array}{ccc} \lambda\mathbf{R}(n-1) + \mathbf{X}(n)\mathbf{X}^T(n) & \lambda\mathbf{R}^T(n-1)\mathbf{U}_d(n-1) + \mathbf{X}(n)d(n) & \mathbf{X}(n) \\ \lambda\mathbf{U}_d^T(n-1)\mathbf{R}(n-1) + \mathbf{X}^T(n)d(n) & \lambda\mathbf{U}_d^T(n-1)\mathbf{U}_d(n-1) + d^2(n) & d(n) \\ \mathbf{X}^T(n) & d(n) & 1 \end{array}\right]$$

$$= \left[\begin{array}{ccc} \mathbf{R}^T(n)\mathbf{R}(n) & \mathbf{R}^T(n)\mathbf{U}_d(n) & \mathbf{R}^T(n)\mathbf{Z}(n) \\ \mathbf{U}_d^T(n)\mathbf{R}(n) & \mathbf{U}_d^T(n)\mathbf{U}_d(n) + \tilde{e}^2(n) & \mathbf{U}_d^T(n)\mathbf{Z}(n) + \tilde{e}(n)\tilde{\gamma}(n) \\ \mathbf{Z}^T(n)\mathbf{R}(n) & \mathbf{Z}^T(n)\mathbf{U}_d(n) + \tilde{\gamma}(n)\tilde{e}(n) & \mathbf{Z}^T(n)\mathbf{Z}(n) + \tilde{\gamma}^2(n) \end{array}\right]. \tag{10.125}$$

By equating the $(2, 3)th$ block elements on both sides, we get

$$d(n) = \mathbf{U}_d^T(n)\mathbf{Z}(n) + \tilde{e}(n)\tilde{\gamma}(n). \tag{10.126}$$

Substituting $\hat{d}(n) = \mathbf{U}_d^T(n)\mathbf{Z}(n)$ from (10.121), we see that the estimation error is given by

$$e(n) = \tilde{e}(n)\tilde{\gamma}(n). \tag{10.127}$$

It is left as an exercise to show by direct calculation that

$$\tilde{\gamma}(n) = \prod_{i=1}^{L} \cos\theta_i(n). \tag{10.128}$$

We now have an algorithm for computing the estimation error without explicitly calculating the coefficient vector via the back-substitution step. The complete algorithm is given

Table 10.7: QR-RLS Adaptive Filter Without Back-Substitution.

**Initialization**

$$\mathbf{R}(0) = \sqrt{\delta}\mathbf{I}$$
$$\mathbf{U}_d(0) = \mathbf{0}$$
$$\mathbf{W}(0) = \mathbf{0} \text{ ; } \mathbf{W}(n) \text{ can be initialized arbitrarily if necessary}$$

**Recursion**

Compute rotations as in Table 10.5

$$\hat{\mathbf{Q}}(n)\begin{bmatrix} \sqrt{\lambda}\mathbf{R}(n-1) & \sqrt{\lambda}\mathbf{U}_d(n-1) & 0 \\ \mathbf{X}^T(n) & d(n) & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R}(n) & \mathbf{U}_d(n) & \mathbf{Z}(n) \\ \mathbf{0} & \tilde{e}(n) & \tilde{\gamma}(n) \end{bmatrix}$$

$$e(n) = \tilde{\gamma}(n)\tilde{e}(n)$$

in Table 10.7. Explicit calculations of the angle of rotations are performed as in Table 10.5. The vector $\mathbf{Z}(n)$ need not be updated for updating the other quantities in the algorithm. A MATLAB program to implement the system is given in Table 10.8.

**Example 10.5: Line Enhancement Using the QR-RLS Adaptive Filter**

Line enhancement is an application in which only the estimate of the desired response signal is required, and not the direct form coefficients of the estimator. Consequently, the QR-RLS adaptive filter is ideally suited for this application. We consider the line enhancement problem discussed in Example 2.13. The input signal is of the form

$$x(n) = \cos\left(\frac{\pi}{6}n + \theta\right) + \eta(n),$$

where $\theta$ is a random phase angle uniformly distributed in the range $[-\pi, \pi)$, and $\eta(n)$ is a white Gaussian sequence, independent of $\theta$, with zero mean value and variance $\sigma_\eta^2 = 1$. The adaptive filter set up is shown in Figure 10.12. A QR-RLS adaptive filter with 25 coefficients (recall that this was the number selected by the model order determination procedure in Example $\delta = 0.001$ and $\lambda = 0.999$ was employed to enhance the signal. Figure 10.13a shows plots of the first 500 samples of the input signal and the enhanced signal obtained as the output of the adaptive filter. Figure 10.13b shows the same signals during the time interval $1500 \leq n \leq 2000$. Comparing the enhanced signal with the original input signal, we can see that the adaptive filter is able to quickly adapt to the environment and produce cleaner signals at its output.

Table 10.8: MATLAB program to implement the QR-RLS adaptive filter without back-substitution.

```
function [e,dhat,W] = qrrls2(lambda,delta,L,x,d);

%  This function adapts a finite-impulse-response (FIR)
%  filter using the QR-Decomposition-based recursive least-squares (QR-RLS)
%  adaptive filter.
%
%  Input parameters:
%      lambda   = forgetting factor
%      delta    = initialization constant for autocorrelation matrix
%      x        = input signal (num_iter x 1)
%      d        = desired response signal (num_iter x 1)
%      L        = number of coefficients
%
%  Output of program:
%      W        = Adaptive filter coefficients (L x (num_iter + 1))
%      dhat     = output of adaptive filter (num_iter x 1)
%      e        = error of adaptive filter (num_iter x 1)

lambda2 = sqrt(lambda);
start_iter = 1;
end_iter = min([length(x) length(d)]);
R = diag(ones(L,1)*sqrt(delta));
U = zeros(L,1);
X = zeros(L,1);
W = zeros(L,end_iter);
e = zeros(end_iter,1);
dhat = zeros(end_iter,1);

for n = start_iter:end_iter,
 X = [x(n);X(1:L-1)];
 R = lambda2*R;
 U = lambda2*U;
 X1 = X;
 etilde = d(n);
 gammatilde = 1;
        for i=1:L,
              den = sqrt(R(i,i)^2 + X1(i)^2);
              c = R(i,i)/den;
              s = X1(i)/den;
```

```
               Q = [c, s; -s, c];
               R2 = Q*[R(i,i:L);X1(i:L)'];
               R(i,i:L) = R2(1,:);
               X1(i:L) = R2(2,:);
               U2 = Q*[U(i);etilde];
               U(i) = U2(1);
               etilde = U2(2);
               gammatilde = gammatilde*c;
         end;
     e(n) = etilde*gammatilde;
         dhat(n) = d(n) - e(n);
end;
```

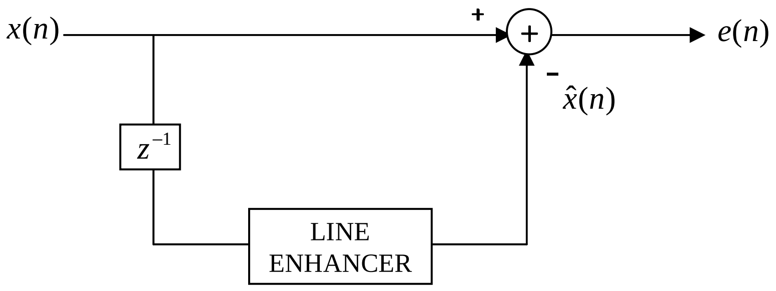
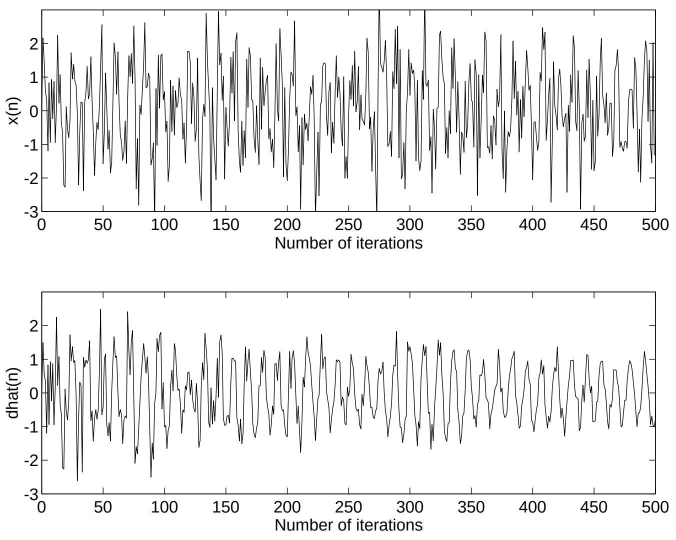
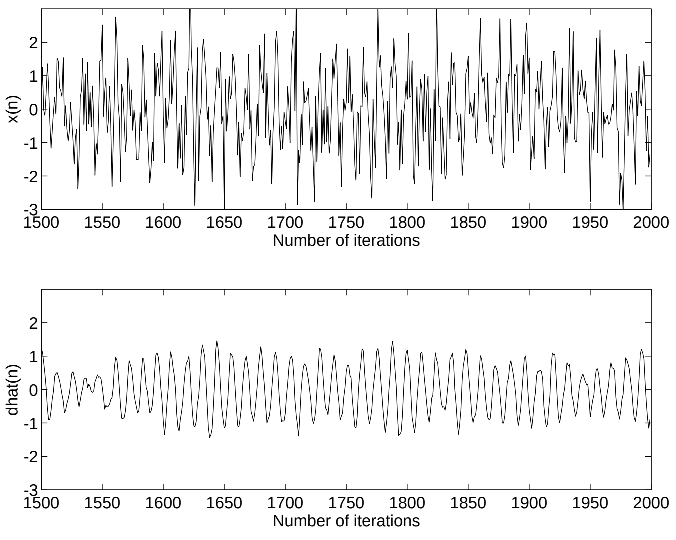


Figure 10.12: Block diagram of the adaptive line enhancer using the conventional RLS adaptive filter.



(a) Results of the first 500 iterations.



(b) Results of iterations in the range $1500 \leq n \leq 2000$.

Figure 10.13: Original and enhanced signals of Example 10.5.

## 10.6 Main Points of This Chapter

- LMS adaptive filters can be implemented using matrix step sizes to combat the difficulties associated with their input signal-dependent performance. In particular, if the step-size is chosen to be $\boldsymbol{\mu} = \mu \mathbf{R}_{\mathbf{xx}}^{-1}$, the mean values of all the coefficients will evolve toward their optimal MMSE values with the same adaptation speed.

- Recursive least-squares adaptive filters find the exact solution of the optimization problem of minimizing a weighted sum of the squared estimation errors at each time instant. This algorithm is identical to the "optimal" matrix step size LMS adaptive filter.

- The conventional RLS adaptive filter requires an order of magnitude smaller number of computations than direct evaluation of the RLS solution at each time instant.

- Using a highly simplified analysis procedure, we showed that the RLS adaptive filter exhibits superior convergence behavior to the LMS adaptive filter. For the models of nonstationarity considered, the RLS adaptive filter outperforms the LMS adaptive filter if i) the filter length $L$ is large, or ii) the level of nonstationarity is high. In other situations, the two adaptive filters have similar tracking performance.

- The conventional RLS adaptive filter, if implemented properly, is numerically stable for $\lambda < 1$. No RLS adaptive filter is numerically stable for $\lambda = 1$ unless the estimate of $\mathbf{R}_{\mathbf{xx}}^{-1}$ is periodically re-initialized.

- The QRD-based adaptive filters belong to the class of RLS algorithms that propagate a Cholesky factor of the least-squares autocorrelation matrix. QR-RLS adaptive filters obtain the same solution as the conventional RLS adaptive filters using orthogonal transformations such as Givens rotations. Because these transformations are numerically stable operations, such algorithms are numerically well-behaved for $\lambda < 1$.

- It is possible to compute the least-squares estimation error without explicitly calculating the filter coefficients in the QR-RLS adaptive filter. Such algorithms are useful for noise cancellation and line enhancement applications, among others.

## 10.7 Bibliographical Notes

**Origins of Recursive Least Squares.** The beginnings of recursive least-squares estimation can be traced to Gauss [Gauss 1821, 1823, 1826]. In his book [Young 1984], Young has included a translation, with commentary, of the relevant pages of Gauss's derivation of the recursive least-squares approach. Gauss invented the method to handle the vast calculations he undertook to locate the asteroid Ceres [Kailath 1974]. Gauss's work was largely unused and forgotten for over a century and a half until it was rediscovered by Plackett in 1950 [Plackett 1950]. Plackett's work derived the corrections one has to make to the least-squares solution when $s$ additional observations are available. Unlike Gauss, who derived his results using scalar analysis, Plackett's work involved more elegant matrix-vector analysis. Kalman, in 1960, derived the recursive least-squares solution in a more sophisticated form [Kalman 1960]. This work, done almost certainly without the knowledge of either Gauss's or Plackett's works, was developed using a state space formulation, and has formed the foundation on which much of recursive least-squares estimation theory has been developed over the last four decades. Even though our presentation in this chapter does not use state space formulation relevant to Kalman filters, there are fundamental and strong connections between the RLS adaptive filters of this chapter and the Kalman filter. Such connections are discussed in [Ho 1963], [Bohlin 1970], [Åström 1971] and more recently in [Sayed 1994]. It appears that the matrix inversion lemma was first derived by Woodbury [Woodbury 1950]. According to [Haykin 1996], Kailath was the first one to use the matrix inversion lemma in filtering literature [Kailath 1960]. Ho also made use of this lemma in [Ho 1963].

**Variations of Conventional RLS Adaptive Filters.** There are several variations of the conventional RLS adaptive filter that propagate the square root or Cholesky factor of the inverse of the least-squares autocorrelation matrix. Examples can be found in the books by Bierman [Bierman 1976], Giordano and Hsu [Giordano 1985], Haykin [Haykin 1996] and the paper by Hsu [Hsu 1982]. The QRD-based adaptive filters discussed in this chapter belong to the above class of algorithms. The QR-RLS algorithm was introduced by Gentleman and Kung [Gentleman 1981]. McWhirther [McWhirther 1983] simplified the algorithm by eliminating the need for back-substitution where only the estimation error is required. One of the greatest attractions of the QR-RLS adaptive filters is the ease of implementing them in VLSI using systolic arrays [Gentleman 1981, McWhirther 1983]. There are several different methods for creating and propagating triangular arrays recursively. Givens rotations [Givens 1958] and Householder transformations [Householder 1958] are the most commonly used tools used for updating the coefficients in the QR-RLS adaptive filters. Books by Lawson and Hanson [Lawson 1974] and Golub and Van Loan [Golub 1983] contain good discussions of both Givens and Householder transformations. A QR-RLS algorithm implemented using only Givens rotations and also finds the direct form coefficient vector without performing the back-substitution operation was derived by Alexander and Ghirnikar [Alexander 1993].

**Analysis of RLS Adaptive Filters.** The tracking and convergence analysis of the exponentially-weighted RLS adaptive filter that we presented in this chapter is an extremely simplified one. A more elaborate analysis along the same lines is given in [Eleftheriou 1986]. Ljung and Ljung studied the error propagation properties of several types of RLS adaptive filters [Ljung 1985]. Verhagen showed in [Verhagen 1989] the importance of preserving the symmetry of the computed inverse of the least-squares autocorrelation matrix to maintain the numerical stability of the conventional RLS adaptive filter. Bottomly and Alexander performed a more complete analysis of the conventional RLS adaptive filter by including all numerical errors introduced by the calculations in their analysis [Bottomly 1989]. An analysis of the numerical error propagation in conventional RLS and LMS adaptive filters implemented in floating point arithmetic is given in [Ardalan 1986]. Fixed point roundoff error analysis of an RLS filter operating in a time-varying environment is given in [Ardalan 1987]. The effect of inexact initialization on the transient behavior of the exponentially-weighted RLS adaptive filters is studied in [Hubing 1991].

**Comparisons with the LMS Adaptive Filter.** An interesting paper that compares the capabilities of the LMS and RLS filters is [Cioffi 1985]. This paper, written based on the results in [Eleftheriou 1986], makes conclusions similar to the ones in Section 10.3.5 in most instances. A comparable set of conclusions have also been made in [Ling 1984] and [McLaughlin 1987]. Gunnarson and Ljung [Gunnarson 1989] analyzed the tracking properties of LMS and RLS adaptive filters in the frequency domain. They concluded that neither of the two methods can claim superior steady state performance over the other. Interestingly, this paper also states that the superiority of transient performance of the RLS algorithm over the LMS filter is guaranteed at all frequencies only if the RLS adaptive filters employ time-varying forgetting factors. Macchi and Bershad have compared the performance of the two adaptive filters in tracking chirped sinusoids [Bershad 1991, Macchi 1991]. They showed that the LMS adaptive filter is superior to the RLS algorithm in this application in low signal to noise ratios and the opposite is true for the high signal-to-noise-ratio case. Similar results are available in [Macchi 1991b] and [Wei 1994]. The book by Benveniste, Metivier and Priouret [Benveniste 1987] contains examples of nonstationary models for which one of the two techniques exhibits superiority over the other. These examples are considered in Exercise 10.10.

## 10.8 Exercises

10.1. *Singularity of the Estimated Autocorrelation Matrix During the Initial Stages of Adaptation:* Show that the estimate of the autocorrelation matrix in (10.6) is singular if $K < L$. Can you make similar conclusions about the estimate in (10.9)?

10.2. *Linear Phase RLS Adaptive Filters:* Consider the problem of deriving a recursive least-squares adaptive filter that employs a linear phase FIR system model of the form

$$\hat{d}(n) = w_0(n)x(n) + \sum_{i=1}^{\frac{L}{2}-1} w_i(n)\left[x(n-i) + x(n-L+i)\right].$$

Derive a set of recursions similar to that of the conventional RLS adaptive filter for this problem.

10.3. *Sliding Window RLS Adaptation:* Derive a recursive solution for the estimation problem that minimizes

$$J(n) = \sum_{k=0}^{M-1} \Big(d(n-k) - \mathbf{W}^T(n)\mathbf{X}(n-k)\Big)^2$$

at each time.

*Hint:* Express $\hat{\mathbf{R}}_{\mathbf{xx}}(n)$ as $\hat{\mathbf{R}}_{\mathbf{xx}}(n-1)+\mathbf{X}(n)\mathbf{X}^T(n)-\mathbf{X}(n-M)\mathbf{X}^T(n-M)$, and apply the matrix inversion lemma twice.

10.4. *Time-Varying Forgetting Factor*: We consider a time-varying forgetting profile of the form $\rho(n,k)$ so that the cost function to be minimized is

$$J(n) = \sum_{k=1}^{n} \rho(n,k)\left(d(k) - \mathbf{W}^T(n)\mathbf{X}(k)\right)^2 .$$

We further restrict the weighting sequence so that $\rho(n,n) = 1$ and $\rho(n,k) = \lambda(n)\rho(n-1,k)$ so that

$$\rho(n,k) = \Pi_{j=k+1}^{n}\lambda(j)$$

Show that the conventional RLS adaptive filter with $\lambda$ replaced with $\lambda(n)$ is a recursive solution to this problem.

10.5. *Initialization using an arbitrary coefficient vector:* Suppose that we wish to initialize the exponentially-weighted RLS adaptive filter with the possibly non-zero coefficient vector $\mathbf{W}(0)$. One approach to deriving the adaptive filter is to minimize the cost function

$$J(n) \ = \lambda^n\delta \parallel \mathbf{W}(n) - \mathbf{W}(0) \parallel^2 + \sum_{k=1}^{n} \lambda^{n-k}\left(d(k) - \mathbf{W}^T(n)\mathbf{X}(k)\right)^2$$

at each time, where $\delta$ represents a positive constant that depends on the level of confidence we have in the initial coefficient vector. Show that the recursive solution to this adaptive filtering problem is identical to that of the conventional RLS adaptive filter initialized using $\mathbf{W}(0)$ and $\mathbf{R}_{\mathbf{xx}}(0) = \delta\mathbf{I}$.

10.6. *Regularized RLS Adaptive Filter:* Consider the cost function given by

$$J_\gamma(n) = \gamma \parallel \mathbf{W}(n) \parallel^2 + \sum_{k=1}^{n} \lambda^{n-k}(d(k) - \mathbf{W}^T(n)\mathbf{X}(k))^2,$$

where $\gamma$ is a positive number. Such a cost function is useful when $\lambda$ is much less than one.

a. Assuming that $\mathbf{X}(n)$ and $d(n)$ are jointly stationary signals, determine the steady-state solution of $\mathbf{W}(n)$ that minimizes $J_\gamma(n)$.

b. Assuming that $\gamma$ is much smaller than the smallest eigenvalue of $\hat{\mathbf{R}}_{\mathbf{xx}}(n)$, derive an expression for the difference between $\mathbf{W}(n)$ and the coefficient vector of the conventional RLS adaptive filter.

c. Derive an $O(L^2)$ algorithm that minimizes $J_\gamma(n)$ recursively in time.

*Hint:* For part b., express $[\gamma\mathbf{I} + \hat{\mathbf{R}}_{\mathbf{xx}}(n)]^{-1}$ as a matrix Taylor series expansion about $\hat{\mathbf{R}}^{-1}_{\mathbf{xx}}(n)]$.

10.7. *Another approach to initialization of the conventional RLS adaptive filter:* We saw in Exercise 10.5 that the exact initialization procedure constrains the adaptive filter to have only $n$ non-zero coefficients at time $n \leq L$. This provides another means for initializing the RLS adaptive filter. We start the iterations with $\hat{r}_{xx}(0) = \delta$. At time $n = 1$, we find the first coefficient $w_0(1)$ using the input samples $x(1)$ and $d(1)$, and the initial value of the input signal power. At the next time, we can increase the number of non-zero coefficients to two, and so on till time $n = L - 1$, when we have all $L$ coefficients to be possibly non-zero.

a. Derive the recursions for the above approach to initializing the RLS adaptive filter.

b. Suppose that we augment the data so that $x(-L) = \sqrt{\delta}$ and $x(n) = 0$ for $n = -L+1, -L+2, \cdots, 0$. Set $d(n) = 0$ for $n \leq 0$. Show that the conventional RLS adaptive filter that minimizes the *soft-constrained* cost function

$$J(n) = \sum_{k=-L}^{n} \lambda^{n-k} \Big(d(k) - \mathbf{W}^T(n)\mathbf{X}(k)\Big)^2$$

is identical to the solution in Part a.

10.8. *Exact Initialization of RLS Adaptive Filters:* Suppose that we wish to initialize the conventional RLS adaptive filter exactly. Our objective is to choose the coefficients $w_0(n)$, $w_1(n)$, $\cdots$, $w_{n-1}(n)$ for $1 \leq n \leq L$ such that

$$d(k) = \sum_{i=0}^{k-1} w_i(n)x(k-i)$$

for $k = 1, 2, \cdots, n$. Find an efficient recursive solution to this problem.

*Hint:* At time $n$, you can choose

$$w_i(n) = w_i(n-1)\ ;\ i = 0, 1, \cdots, n-2.$$

Thus you need to solve for only one coefficient at each time. Assume that you can directly invert

$$\mathbf{R}^{-1}_{\mathbf{xx}}(L) = \sum_{k=1}^{L} \lambda^{L-k}\mathbf{X}(k)\mathbf{X}^T(k)$$

at time $L$.

10.9. *Another Approach For Analyzing the RLS Adaptive Filter:* We assume in this problem that the adaptive filter is operating in a stationary environment.

a. Show that

$$\mathbf{V}(n) = \lambda^n \hat{\mathbf{R}}_{\mathbf{XX}}^{-1}(n)\hat{\mathbf{R}}_{\mathbf{XX}}(0)\mathbf{V}(0) + \hat{\mathbf{R}}_{\mathbf{XX}}^{-1}(n)\left[\sum_{k=1}^{n} \lambda^{n-k}\mathbf{X}(k)\eta(k)\right]$$

for the exponentially weighted RLS adaptive filter, where $\eta(n)$ represents the measurement noise, and the adaptive filter is operating in the system identification mode with an adequate number of coefficients.

b. Assuming that $\eta(n)$ is a zero-mean and i.i.d sequence that is independent of the input signal $x(n)$, derive an expression for the mean value of the coefficient error vector.

c. Explain, using the result in part b, how the evolution of the coefficient error vector depends on the input signal statistics and the initialization.

*Hint:* For part a, first show that

$$\hat{\mathbf{R}}_{\mathbf{XX}}(n)\mathbf{V}(n) = \lambda\hat{\mathbf{R}}_{\mathbf{XX}}(n-1)\mathbf{V}(n-1) + \mathbf{X}(n)\eta(n).$$

10.10. *Comparison of the Steady-State Excess Mean-Squared Errors of the LMS and RLS Adaptive Filters:* Compare the steady-state excess mean-square estimation error of LMS and RLS adaptive filters when the adaptive filters operate in the system identification mode and the nonstationarity model of (10.44) is employed with the exception that

a. $E\{\mathbf{M}(n)\mathbf{M}^T(n)\} = \alpha\mathbf{R}_{\mathbf{XX}}$.

b. $E\{\mathbf{M}(n)\mathbf{M}^T(n)\} = \alpha\mathbf{R}_{\mathbf{XX}}^{-1}$.

In each case, is it possible to show a performance advantage of one of the filters over the other?

10.11. *Transient Mean-Square Response of the Conventional RLS Adaptive Filter:* Using similar types of assumptions and approximations that have been used for analyzing LMS and RLS adaptive filters, derive the evolution equations that describe the behavior of the correlation matrix of the coefficient error vector. Assume that $1-\lambda$ is very small.

10.12. *Error Propagation Analysis for CRLS2:* Derive (10.83) for the CRLS2 realization of the conventional RLS adaptive filter.

10.13. *RLS Adaptive Filters Using the Householder Transform:* Let $\mathbf{V}$ be a vector of $L$ elements that are not all zero. Let

$$\sigma = \begin{cases} +1\,; & \text{if} \quad v_1 \geq 0 \\ -1\,; & \text{if} \quad v_1 < 0, \end{cases}$$

where $v_1$ is the first element of $\mathbf{V}$. Define $\mathbf{U}$ to be

$$\mathbf{U} = \mathbf{V} + \sigma\sqrt{\mathbf{V}^T\mathbf{V}}\mathbf{e}_1,$$

where $\mathbf{e}_1$ is an $L$-element vector whose first entry is 1 and all other entries are zero. Finally, let

$$\mathbf{Q} = \mathbf{I} - \frac{2\mathbf{U}\mathbf{U}^T}{\mathbf{U}^T\mathbf{U}}.$$

The matrix $\mathbf{Q}$ is known as the Householder transformation matrix.

a. Show that

$$\mathbf{Q}\mathbf{V} = -\sigma\sqrt{\mathbf{V}^T\mathbf{V}}\mathbf{e}_1.$$

b. Derive an exponentially-weighted RLS adaptive filter using the Householder transformations. Compare your algorithm with the Givens rotations-based method.

10.14. *Calculation of $\tilde{\gamma}(n)$ in QR-RLS Adaptive Filter:* Show that

$$\tilde{\gamma}(n) = \prod_{i=1}^{L} \cos\theta_i(n)$$

for the QR-RLS adaptive filter.

10.15. *Calculation of $\mathbf{Z}(n)$ in QR-RLS Adaptive Filter:* Find an expression for $\mathbf{Z}(n)$ in terms of the angle of rotations in the QR-RLS adaptive filter.

10.16. *Relationship of $\tilde{\gamma}(n)$ to the Gain Vector:* Show that

$$\tilde{\gamma}^2(n) = 1 - \mathbf{k}^T(n)\mathbf{X}(n).$$

10.17. *Computing Assignment: Comparison of the Exponentially-Weighted RLS Adaptive Filter with the LMS and the Gradient Adaptive Step Size LMS Adaptive Filters.* For this assignment, we consider the use of the three adaptive filters in an adaptive prediction problem. The input signal to the adaptive filters is generated using the model

$$x(n) = 0.44\xi(n) + 1.5x(n-1) - x(n-2) + 0.25x(n-3),$$

where $\xi(n)$ is a zero-mean, i.i.d. Gaussian-distributed random process with unit variance.

a. Generate a 2000-sample sequence using the above model for $x(n)$. Use the conventional RLS adaptive filter to predict $x(n)$ using the most recent three samples. Choose an exponential window with parameter $\lambda = 0.99$. Find the mean values of the three coefficients from fifty independent experiments. Compare the performance of the adaptive filter for several choices of the initialization parameter $\delta$. Estimate the steady-state excess mean-squared error by averaging the mean excess-squared error over the last 500 samples of the fifty experiments.

b. Compare the speed of convergence of the RLS adaptive filter with that of the LMS and the gradient adaptive step size LMS adaptive filters. Choose the step size $\mu$ of the LMS adaptive filter and the parameter $\rho$ of the adaptive step size algorithm such that the excess mean-squared errors for all three algorithms are identical. You may want to perform the experiments for the adaptive step size algorithm using several initial step sizes. Use the theoretical comparison for the small step size case given in the text for selecting $\mu$ for the LMS adaptive filter. Even though the input signal is not white, you may want to choose $\rho$ from equation (45) for the excess mean-squared estimation error for white input signals given in [Mathews 1993].

10.18. *Computing Assignment: Noise Cancellation.* There are several situations in which one needs to eliminate a corrupting sinusoidal signal from a signal. An example is that of an electro-cardiograph (ECG) system that may not be properly isolated from the line voltage. In such a situation, the output ECG signal may be corrupted by a 60 Hz sinusoidal signal. Even if the leakage of the line voltage is a fraction of a percent of the total voltage, the interference can completely mask the desired signal whose amplitude may only be in the order of a few mill or microvolts. This assignment is a demonstration of how an adaptive filter can be used to cancel sinusoidal interference when the user has access to scaled and phase shifted versions of the interfering signal.

Figure 10.14 shows the block diagram of the interference cancellation system. The main idea is that we can use the sinusoidal signal as the input to the adaptive canceller and use the corrupted signal as the desired response signal. Since the information-bearing signal is uncorrelated with the input sinusoid, the adaptive filter will attempt to track the amplitude and phase of the interference, and the estimation error is the "cleaned-up" signal. Since the input is a single sinusoid, we only require two coefficients. (Why?)

Do the following parts for the assignment:

(a) Generate 1,000 samples of a triangular wave form $x(n)$ with 100 samples per period and one-volt peak-to-peak amplitude. Also generate 1,000 samples of a sinusoidal signal $s(n)$ with 10 samples per period and 2 volts peak-to-peak amplitude. Create the interference signal by scaling $s(n)$ by a factor of four, and introducing a random, initial phase shift that is uniformly distributed in the range $[-\pi, \pi)$. Add the interference signal to the triangular signal to create the desired response signal for the adaptive filter.

(b) Implement the conventional RLS adaptive filter with two coefficients and $\lambda = 0.99$ to perform the noise cancellation task. Plot the estimation error as a function of time and observe how the signal changes as the filter adapts to the amplitude and phase of the interference signal.

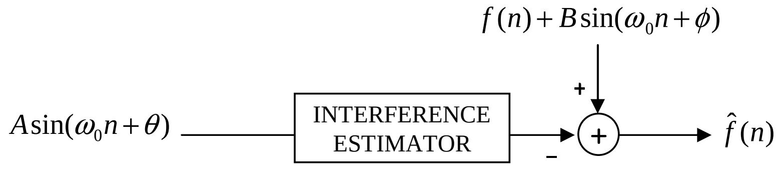

$f(n) + B\sin(\omega_0 n + \phi)$
+
$A\sin(\omega_0 n + \theta)$
INTERFERENCE ESTIMATOR
+
−
$\hat{f}(n)$

Figure 10.14: Interference canceller for Exercise 10.17.

(c) Implement the noise canceller using an LMS adaptive filter. Choose the step size so that the excess mean-square error in steady-state matches that of the RLS adaptive filter. Compare the results obtained with that of part b.

10.19. *Computing Assignment: Tracking Time-Varying Frequencies.* In this assignment, we consider tracking the instantaneous frequency of a sinusoidal input signal with slowly varying frequencies. The input signal is embedded in a bed of white noise, so that it can be modeled as

$$x(n) = A\cos(\phi(n)) + \eta(n),$$

where $\phi(n)$ is the instantaneous phase of the sinusoid. We assume that $\phi(n)$ is the sampled version of a continuously differentiable function. The instantaneous frequency is given by

$$f(n) = \frac{1}{2\pi}\frac{d\phi(t)}{dt}\bigg|_{t=nT},$$

where $T$ is the sampling interval. We consider an instantaneous phase function of the form

$$\phi(t) = \frac{\pi}{6} + 0.00001\pi t^2 + \frac{\pi}{12}t.$$

Assume that $T = 1$.

(a) Plot the instantaneous frequency of the above signal.

(b) To track the frequencies, we employ an adaptive predictor to estimate the most recent sample with the past $L$ samples, and estimate the power spectrum of the input signal using the AR spectrum estimation approach described in Section 2.4.

The instantaneous frequency is determined as the positive frequency at which the spectrum estimate peaks. In order to reduce the computational complexity, you may want to compute the spectrum and estimate the frequency only once every 25 samples or so. Generate 4,000 samples of a signal described by our model above with $A = 1$ and $\eta(n)$ being a zero-mean white, Gaussian sequence with variance 0.01. Plot the instantaneous frequency estimates obtained from an RLS adaptive filter with $\lambda = 0.99$ and $L = 5, 10, 15$ and 20 and compare the results with the true instantaneous frequency. You may find the peak value by evaluating the spectrum at 256 or more equally-spaced samples of frequency values in the range [-0.5, 0.5) cycles/sample.

(c) Repeat the experiments with the LMS adaptive predictor with a similar number of coefficients. As usual, select the step sizes to match the steady-state excess mean-square error of the RLS adaptive filter for each filter length. You can assume that the results we obtained for small $(1 - \lambda)$ and stationary input signals are valid for the input signal in this experiment.