Ali H. Sayed, et. Al. “Recursive Least-Squares Adaptive Filters.”
2000 CRC Press LLC. <http://www.engnetbase.com>.

Recursive Least-Squares Adaptive Filters

21.1 Array Algorithms
Elementary Circular Rotations • Elementary Hyperbolic Rotations • Square-Root-Free and Householder Transformations • A Numerical Example
21.2 The Least-Squares Problem
Geometric Interpretation • Statistical Interpretation
21.3 The Regularized Least-Squares Problem
Geometric Interpretation • Statistical Interpretation
21.4 The Recursive Least-Squares Problem
Reducing to the Regularized Form • Time Updates
21.5 The RLS Algorithm
Estimation Errors and the Conversion Factor • Update of the Minimum Cost
21.6 RLS Algorithms in Array Forms
Motivation • A Very Useful Lemma • The Inverse QR Algorithm • The QR Algorithm
21.7 Fast Transversal Algorithms
The Prewindowed Case • Low-Rank Property • A Fast Array Algorithm • The Fast Transversal Filter
21.8 Order-Recursive Filters
Joint Process Estimation • The Backward Prediction Error Vectors • The Forward Prediction Error Vectors • A Nonunity Forgetting Factor • The QRD Least-Squares Lattice Filter • The Filtering or Joint Process Array
21.9 Concluding Remarks
References

Ali H. Sayed
University of California, Los Angeles

Thomas Kailath
Stanford University

The central problem in estimation is to recover, to good accuracy, a set of unobservable parameters from corrupted data. Several optimization criteria have been used for estimation purposes over the years, but the most important, at least in the sense of having had the most applications, are criteria that are based on quadratic cost functions. The most striking among these is the linear least-squares criterion, which was perhaps first developed by Gauss (ca. 1795) in his work on celestial mechanics. Since then, it has enjoyed widespread popularity in many diverse areas as a result of its attractive computational and statistical properties. Among these attractive properties, the most notable are the facts that least-squares solutions:

- can be explicitly evaluated in closed forms;
- can be recursively updated as more input data is made available, and

- are maximum likelihood estimators in the presence of Gaussian measurement noise.

The aim of this chapter is to provide an overview of adaptive filtering algorithms that result when the least-squares criterion is adopted. Over the last several years, a wide variety of algorithms in this class has been derived. They all basically fall into the following main groups (or variations thereof): recursive least-squares (RLS) algorithms and the corresponding fast versions (known as FTF and FAEST), QR and inverse QR algorithms, least-squares lattice (LSL), and QR decomposition-based least-squares lattice (QRD-LSL) algorithms.

Table 21.1 lists these different variants and classifies them into order-recursive and fixed-order algorithms. The acronyms and terminology are not important at this stage and will be explained as the discussion proceeds. Also, the notation $O(M)$ is used to indicate that each iteration of an algorithm requires of the order of M floating point operations (additions and multiplications). In this sense, some algorithms are fast (requiring only $O(M)$), while others are slow (requiring $O(M^2)$). The value of M is the filter order that will be introduced in due time.

TABLE 21.1 Most Common RLS Adaptive Schemes

Adaptive Algorithm	Order Recursive	Fixed Order	Cost per Iteration
RLS		**x**	$O(M^2)$
QR and Inverse QR		**x**	$O(M^2)$
FTF, FAEST		**x**	$O(M)$
LSL	**x**		$O(M)$
QRD-LSL	**x**		$O(M)$

It is practically impossible to list here all the relevant references and all the major contributors to the rich field of adaptive RLS filtering. The reader is referred to some of the textbooks listed at the end of this chapter for more comprehensive treatments and bibliographies.

Here we wish to stress that, apart from introducing the reader to the fundamentals of RLS filtering, one of our goals in this exposition is to present the different versions of the RLS algorithm in computationally convenient so-called array forms. In these forms, an algorithm is described as a sequence of elementary operations on arrays of numbers. Usually, a prearray of numbers has to be triangularized by a rotation, or a sequence of elementary rotations, in order to yield a postarray of numbers. The quantities needed to form the next prearray can then be read off from the entries of the postarray, and the procedure can be repeated. The explicit forms of the rotation matrices are not needed in most cases.

Such array descriptions are more truly algorithms in the sense that they operate on sets of numbers and provide other sets of numbers, with no explicit equations involved. The rotations themselves can be implemented in a variety of well-known ways: as a sequence of elementary circular or hyperbolic rotations, in square-root- and/or division-free forms, as Householder transformations, etc. These may differ in computational complexity, numerical behavior, and ease of hardware (VLSI) implementation. But, if preferred, explicit expressions for the rotation matrices can also be written down, thus leading to explicit sets of equations in contrast to the array forms.

For this reason, and although the different RLS algorithms that we consider here have already been derived in many different ways in earlier places in the literature, the derivation and presentation in this chapter are intended to provide an alternative unifying exposition that we hope will help a reader get a deeper appreciation of this class of adaptive algorithms.

Notation

We use small boldface letters to denote column vectors (e.g., $\mathbf{w}$) and capital boldface letters to denote matrices (e.g., $\mathbf{A}$). The symbol $\mathbf{I}_n$ denotes the identity matrix of size $n \times n$, while $\mathbf{0}$ denotes a zero column. The symbol ^{T} denotes transposition. This chapter deals with real-valued data. The case of complex-valued data is essentially identical and is treated in many of the references at the end of this chapter.

Square-Root Factors

A symmetric positive-definite matrix $\mathbf{A}$ is one that satisfies $\mathbf{A} = \mathbf{A}^T$ and $\mathbf{x}^T\mathbf{A}\mathbf{x} > 0$ for all nonzero column vectors $\mathbf{x}$. Any such matrix admits a factorization (also known as eigen-decomposition) of the form $\mathbf{A} = \mathbf{U}\Sigma\mathbf{U}^T$, where $\mathbf{U}$ is an orthogonal matrix, namely a square matrix that satisfies $\mathbf{U}\mathbf{U}^T = \mathbf{U}^T\mathbf{U} = \mathbf{I}$, and Σ is a diagonal matrix with real positive entries. In particular, note that $\mathbf{A}\mathbf{U} = \mathbf{U}\Sigma$, which shows that the columns of $\mathbf{U}$ are the right eigenvectors of $\mathbf{A}$ and the entries of Σ are the corresponding eigenvalues.

Note also that we can write $\mathbf{A} = \mathbf{U}\Sigma^{1/2}(\Sigma^{1/2})^T\mathbf{U}^T$, where $\Sigma^{1/2}$ is a diagonal matrix whose entries are (positive) square-roots of the diagonal entries of Σ. Since $\Sigma^{1/2}$ is diagonal, $(\Sigma^{1/2})^T = \Sigma^{1/2}$. If we introduce the matrix notation $\mathbf{A}^{1/2} = \mathbf{U}\Sigma^{1/2}$, then we can alternatively write $\mathbf{A} = (\mathbf{A}^{1/2})(\mathbf{A}^{1/2})^T$. This can be regarded as a square-root factorization of the positive-definite matrix $\mathbf{A}$. Here, the notation $\mathbf{A}^{1/2}$ is used to denote one such square-root factor, namely the one constructed from the eigen-decomposition of $\mathbf{A}$.

Note, however, that square-root factors are not unique. For example, we may multiply the diagonal entries of $\Sigma^{1/2}$ by $\pm 1's$ and obtain a new square-root factor for Σ and, consequently, a new square-root factor for $\mathbf{A}$.

Also, given any square-root factor $\mathbf{A}^{1/2}$, and any orthogonal matrix Θ (satisfying $\Theta\Theta^T = \mathbf{I}$) we can define a new square-root factor for $\mathbf{A}$ as $\mathbf{A}^{1/2}\Theta$ since

$$(\mathbf{A}^{1/2}\Theta)(\mathbf{A}^{1/2}\Theta)^T = \mathbf{A}^{1/2}(\Theta\Theta^T)(\mathbf{A}^{1/2})^T = \mathbf{A}\,.$$

Hence, square factors are highly nonunique. We shall employ the notation $\mathbf{A}^{1/2}$ to denote any such square-root factor. They can be made unique, e.g., by insisting that the factors be symmetric or that they be triangular (with positive diagonal elements). In most applications, the triangular form is preferred. For convenience, we also write

$$\left(\mathbf{A}^{1/2}\right)^T = \mathbf{A}^{T/2},\ \left(\mathbf{A}^{1/2}\right)^{-1} = \mathbf{A}^{-1/2},\ \left(\mathbf{A}^{-1/2}\right)^T = \mathbf{A}^{-T/2}\,.$$

Thus, note the expressions $\mathbf{A} = \mathbf{A}^{1/2}\mathbf{A}^{T/2}$ and $\mathbf{A}^{-1} = \mathbf{A}^{-T/2}\mathbf{A}^{-1/2}$.

21.1 Array Algorithms

The array form is so important that it will be worthwhile to explain its generic form here.

An array algorithm is described via rotation operations on a prearray of numbers, chosen to obtain a certain zero pattern in a postarray. Schematically, we write

$$\begin{bmatrix} x & x & x & x \\ x & x & x & x \\ x & x & x & x \\ x & x & x & x \end{bmatrix}\Theta = \begin{bmatrix} x & 0 & 0 & 0 \\ x & x & 0 & 0 \\ x & x & x & 0 \\ x & x & x & x \end{bmatrix},$$

where Θ is any rotation matrix that triangularizes the prearray. In general, Θ is required to be a $\mathbf{J}-$orthogonal matrix in the sense that it should satisfy the normalization $\Theta\mathbf{J}\Theta^T = \mathbf{J}$, where $\mathbf{J}$

is a given signature matrix with $\pm 1's$ on the diagonal and zeros elsewhere. The orthogonal case corresponds to $\mathbf{J} = \mathbf{I}$ since then $\Theta\Theta^T = \mathbf{I}$.

A rotation Θ that transforms a prearray to triangular form can be achieved in a variety of ways: by using a sequence of elementary Givens and hyperbolic rotations, Householder transformations, or square-root-free versions of such rotations. Here we only explain the elementary forms. The other choices are discussed in some of the references at the end of this chapter.

21.1.1 Elementary Circular Rotations

An elementary 2×2 orthogonal rotation Θ (also known as Givens or circular rotation) takes a row vector $\begin{bmatrix} a & b \end{bmatrix}$ and rotates it to lie along the basis vector $\begin{bmatrix} 1 & 0 \end{bmatrix}$. More precisely, it performs the transformation

$$\begin{bmatrix} a & b \end{bmatrix} \Theta = \begin{bmatrix} \pm\sqrt{|a|^2 + |b|^2} & 0 \end{bmatrix} . \tag{21.1}$$

The quantity $\pm\sqrt{|a|^2 + |b|^2}$ that appears on the right-hand side is consistent with the fact that the prearray, $\begin{bmatrix} a & b \end{bmatrix}$, and the postarray, $\begin{bmatrix} \pm\sqrt{|a|^2 + |b|^2} & 0 \end{bmatrix}$, must have equal Euclidean norms (since an orthogonal transformation preserves the Euclidean norm of a vector).

An expression for Θ is given by

$$\Theta = \frac{1}{\sqrt{1+\rho^2}} \begin{bmatrix} 1 & -\rho \\ \rho & 1 \end{bmatrix} , \quad \rho = \frac{b}{a} , \quad a \neq 0. \tag{21.2}$$

In the trivial case $a = 0$ we simply choose Θ as the permutation matrix,

$$\Theta = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} .$$

The orthogonal rotation (21.2) can also be expressed in the alternative form:

$$\Theta = \begin{bmatrix} c & -s \\ s & c \end{bmatrix} ,$$

where the so-called cosine and sine parameters, c and s, respectively, are defined by

$$c = \frac{1}{\sqrt{1+\rho^2}} \quad , \quad s = \frac{\rho}{\sqrt{1+\rho^2}} .$$

The name *circular rotation* for Θ is justified by its effect on a vector; it rotates the vector along the *circle* of equation $x^2 + y^2 = |a|^2 + |b|^2$, by an angle θ that is determined by the inverse of the above cosine and/or sine parameters, $\theta = \tan^{-1} \rho$, in order to align it with the basis vector $\begin{bmatrix} 1 & 0 \end{bmatrix}$. The trivial case $a = 0$ corresponds to a 90 degrees rotation in an appropriate clockwise (if $b \geq 0$) or counterclockwise (if $b < 0$) direction.

21.1.2 Elementary Hyperbolic Rotations

An elementary 2×2 hyperbolic rotation Θ takes a row vector $\begin{bmatrix} a & b \end{bmatrix}$ and rotates it to lie either along the basis vector $\begin{bmatrix} 1 & 0 \end{bmatrix}$ (if $|a| > |b|$) or along the basis vector $\begin{bmatrix} 0 & 1 \end{bmatrix}$ (if $|a| < |b|$). More precisely, it performs either of the transformations

$$\begin{bmatrix} a & b \end{bmatrix} \Theta = \begin{bmatrix} \pm\sqrt{|a|^2 - |b|^2} & 0 \end{bmatrix} \quad \text{if } |a| > |b| , \tag{21.3}$$

$$\begin{bmatrix} a & b \end{bmatrix} \Theta = \begin{bmatrix} 0 & \pm\sqrt{|b|^2 - |a|^2} \end{bmatrix} \quad \text{if } |a| < |b| . \tag{21.4}$$

The quantity $\sqrt{\pm(|a|^2 - |b|^2)}$ that appears on the right-hand side of the above expressions is consistent with the fact that the prearray, $\begin{bmatrix} a & b \end{bmatrix}$, and the postarrays must have equal hyperbolic "norms." By the hyperbolic "norm" of a row vector $\mathbf{x}^T$ we mean the indefinite quantity $\mathbf{x}^T\mathbf{J}\mathbf{x}$, which can be positive or negative. Here,

$$\mathbf{J} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = (1 \oplus -1) \ .$$

An expression for a hyperbolic rotation Θ that achieves (21.3) or (21.4) is given by

$$\Theta = \frac{1}{\sqrt{1-\rho^2}} \begin{bmatrix} 1 & -\rho \\ -\rho & 1 \end{bmatrix} , \tag{21.5}$$

where

$$\rho = \begin{cases} \frac{b}{a} & \text{when } \ a \neq 0 \ \text{ and } \ |a| > |b| \\ \frac{a}{b} & \text{when } \ b \neq 0 \ \text{ and } \ |b| > |a| \end{cases}$$

The hyperbolic rotation (21.5) can also be expressed in the alternative form:

$$\Theta = \begin{bmatrix} ch & -sh \\ -sh & ch \end{bmatrix} ,$$

where the so-called hyperbolic cosine and sine parameters, ch and sh, respectively, are defined by

$$ch = \frac{1}{\sqrt{1-\rho^2}} \quad , \quad sh = \frac{\rho}{\sqrt{1-\rho^2}} \ .$$

The name *hyperbolic rotation* for Θ is again justified by its effect on a vector; it rotates the original vector along the *hyperbola* of equation $x^2 - y^2 = |a|^2 - |b|^2$, by an angle θ determined by the inverse of the above hyperbolic cosine and/or sine parameters, $\theta = \tanh^{-1}[\rho]$, in order to align it with the appropriate basis vector. Note also that the special case $|a| = |b|$ corresponds to a row vector $\begin{bmatrix} a & b \end{bmatrix}$ with zero hyperbolic norm since $|a|^2 - |b|^2 = 0$. It is then easy to see that there does not exist a hyperbolic rotation that will rotate the vector to lie along the direction of one basis vector or the other.

21.1.3 Square-Root-Free and Householder Transformations

We remark that the above expressions for the circular and hyperbolic rotations involve square-root operations. In many situations, it may be desirable to avoid the computation of square-roots because it is usually expensive. For this and other reasons, square-root- and division-free versions of the above elementary rotations have been developed and constitute an attractive alternative.

Therefore one could use orthogonal or $\mathbf{J}$—orthogonal Householder reflections (for given $\mathbf{J}$) to simultaneously annihilate several entries in a row, e.g., to transform $\begin{bmatrix} x & x & x & x \end{bmatrix}$ directly to the form $\begin{bmatrix} x' & 0 & 0 & 0 \end{bmatrix}$. Combinations of rotations and reflections can also be used.

We omit the details here but the idea is clear. There are many different ways in which a prearray of numbers can be rotated into a postarray of numbers.

21.1.4 A Numerical Example

Assume we are given a 2×3 prearray $\mathbf{A}$,

$$\mathbf{A} = \begin{bmatrix} 0.875 & 0.15 & 1.0 \\ 0.675 & 0.35 & 0.5 \end{bmatrix}, \tag{21.6}$$

and wish to triangularize it via a sequence of elementary circular rotations, i.e., reduce $\mathbf{A}$ to the form

$$\mathbf{A}\Theta = \begin{bmatrix} x & 0 & 0 \\ x & x & 0 \end{bmatrix}. \tag{21.7}$$

This can be obtained, among several different possibilities, as follows. We start by annihilating the $(1, 3)$ entry of the prearray (21.6) by pivoting with its $(1, 1)$ entry. According to expression (21.2), the orthogonal transformation Θ_1 that achieves this result is given by

$$\Theta_1 = \frac{1}{\sqrt{1+\rho_1^2}} \begin{bmatrix} 1 & -\rho_1 \\ \rho_1 & 1 \end{bmatrix} = \begin{bmatrix} 0.6585 & -0.7526 \\ 0.7526 & 0.6585 \end{bmatrix}, \quad \rho_1 = \frac{1}{0.875}.$$

Applying Θ_1 to the prearray (21.6) leads to (recall that we are only operating on the first and third columns, leaving the second column unchanged):

$$\begin{bmatrix} 0.875 & 0.15 & 1 \\ 0.675 & 0.35 & 0.5 \end{bmatrix} \begin{bmatrix} 0.6585 & 0 & -0.7526 \\ 0 & 1 & 0 \\ 0.7526 & 0 & 0.6585 \end{bmatrix} = \begin{bmatrix} 1.3288 & 0.1500 & 0.0000 \\ 0.8208 & 0.3500 & -0.1788 \end{bmatrix}. \tag{21.8}$$

We now annihilate the $(1, 2)$ entry of the resulting matrix in the above equation by pivoting with its $(1, 1)$ entry. This requires that we choose

$$\Theta_2 = \frac{1}{\sqrt{1+\rho_2^2}} \begin{bmatrix} 1 & -\rho_2 \\ \rho_2 & 1 \end{bmatrix} = \begin{bmatrix} 0.9937 & -0.1122 \\ 0.1122 & 0.9937 \end{bmatrix}, \quad \rho_2 = \frac{0.1500}{1.3288}. \tag{21.9}$$

Applying Θ_2 to the matrix on the right-hand side of (21.8) leads to (now we leave the third column unchanged)

$$\begin{bmatrix} 1.3288 & 0.1500 & 0.0000 \\ 0.8208 & 0.3500 & 0.1788 \end{bmatrix} \begin{bmatrix} 0.9937 & -0.1122 & 0 \\ 0.1122 & 0.9937 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1.3373 & 0.0000 & 0.0000 \\ 0.8549 & 0.2557 & 0.1788 \end{bmatrix}. \tag{21.10}$$

We finally annihilate the $(2, 3)$ entry of the resulting matrix in (21.10) by pivoting with its $(2, 2)$ entry. In principle this requires that we choose

$$\Theta_3 = \frac{1}{\sqrt{1+\rho_3^2}} \begin{bmatrix} 1 & -\rho_3 \\ \rho_3 & 1 \end{bmatrix} = \begin{bmatrix} 0.8195 & 0.5731 \\ -0.5731 & 0.8195 \end{bmatrix}, \quad \rho_3 = \frac{0.1788}{-0.2557}, \tag{21.11}$$

and apply it to the matrix on the right-hand side of (21.10), which would then lead to

$$\begin{bmatrix} 1.3373 & 0.0000 & 0.0000 \\ 0.8549 & -0.2557 & 0.1788 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0.8195 & 0.5731 \\ 0 & -0.5731 & 0.8195 \end{bmatrix} = \begin{bmatrix} 1.3373 & 0.0000 & 0.0000 \\ 0.8549 & -0.3120 & 0.0000 \end{bmatrix}. \tag{21.12}$$

Alternatively, this last step could have been implemented without explicitly forming Θ_3. We simply replace the row vector $\begin{bmatrix} -0.2557 & 0.1788 \end{bmatrix}$, which contains the $(2,2)$ and $(2,3)$ entries of the prearray in (21.12), by the row vector $\begin{bmatrix} \pm\sqrt{(-0.2557)^2 + (0.1788)^2} & 0.0000 \end{bmatrix}$, which is equal to $\begin{bmatrix} \pm 0.3120 & 0.0000 \end{bmatrix}$. We choose the positive sign in order to conform with our earlier convention that the diagonal entries of triangular square-root factors are taken to be positive. The resulting postarray is therefore

$$\begin{bmatrix} 1.3373 & 0.0000 & 0.0000 \\ 0.8549 & 0.3120 & 0.0000 \end{bmatrix} . \tag{21.13}$$

We have exhibited a sequence of elementary orthogonal transformations that triangularizes the prearray of numbers (21.6). The combined effect of the sequence of transformations $\{\Theta_1, \Theta_2, \Theta_3\}$ corresponds to the orthogonal rotation Θ required in (21.7). However, note that we do not need to know or to form $\Theta = \Theta_1\Theta_2\Theta_3$.

It will become clear throughout our discussion that the different adaptive RLS schemes can be described in array forms, where the necessary operations are elementary rotations as described above. Such array descriptions lend themselves rather directly to parallelizable and modular implementations. Indeed, once a rotation matrix is chosen, then all the rows of the prearray undergo the same rotation transformation and can thus be processed in parallel. Returning to the above example, where we started with the prearray **A**, we see that once the first rotation is determined, both rows of **A** are then transformed by it, and can thus be processed in parallel, and by the same functional (rotation) block, to obtain the desired postarray. The same remark holds for prearrays with multiple rows.

21.2 The Least-Squares Problem

Now that we have explained the generic form of an array algorithm, we return to the main topic of this chapter and formulate the least-squares problem and its regularized version. Once this is done, we shall then proceed to describe the different variants of the recursive least-squares solution in compact array forms.

Let **w** denote a column vector of n unknown parameters that we wish to estimate, and consider a set of $(N+1)$ noisy measurements $\{d(i)\}$ that are assumed to be linearly related to **w** via the additive noise model

$$d(j) = \mathbf{u}_j^T \mathbf{w} + v(j) ,$$

where the $\{\mathbf{u}_j\}$ are given column vectors. The $(N+1)$ measurements can be grouped together into a single matrix expression:

$$\underbrace{\begin{bmatrix} d(0) \\ d(1) \\ \vdots \\ d(N) \end{bmatrix}}_{\mathbf{d}} = \underbrace{\begin{bmatrix} \mathbf{u}_0^T \\ \mathbf{u}_1^T \\ \vdots \\ \mathbf{u}_N^T \end{bmatrix}}_{\mathbf{A}} \mathbf{w} + \underbrace{\begin{bmatrix} v(0) \\ v(1) \\ \vdots \\ v(N) \end{bmatrix}}_{\mathbf{v}} ,$$

or, more compactly, $\mathbf{d} = \mathbf{A}\mathbf{w} + \mathbf{v}$. Because of the noise component **v**, the observed vector **d** does not lie in the column space of the matrix **A**. The objective of the least-squares problem is to determine the vector in the column space of **A** that is closest to **d** in the least-squares sense.

More specifically, any vector in the range space of **A** can be expressed as a linear combination of its columns, say $\mathbf{A}\hat{\mathbf{w}}$ for some $\hat{\mathbf{w}}$. It is therefore desired to determine the particular $\hat{\mathbf{w}}$ that minimizes the distance between **d** and $\mathbf{A}\hat{\mathbf{w}}$,

$$\min_{\mathbf{w}} \|\mathbf{d} - \mathbf{A}\mathbf{w}\|^2 . \tag{21.14}$$

The resulting $\hat{\mathbf{w}}$ is called the least-squares solution and it provides an estimate for the unknown $\mathbf{w}$. The term $\mathbf{A}\hat{\mathbf{w}}$ is called the linear least-squares estimate (l.l.s.e.) of $\mathbf{d}$.

The solution to (21.14) *always* exists and it follows from a simple geometric argument. The orthogonal projection of $\mathbf{d}$ onto the column span of $\mathbf{A}$ yields a vector $\hat{\mathbf{d}}$ that is the closest to $\mathbf{d}$ in the least-squares sense. This is because the resulting error vector $(\mathbf{d} - \hat{\mathbf{d}})$ will be orthogonal to the column span of $\mathbf{A}$.

In other words, the closest element $\hat{\mathbf{d}}$ to $\mathbf{d}$ must satisfy the orthogonality condition

$$\mathbf{A}^T(\mathbf{d} - \hat{\mathbf{d}}) = \mathbf{0}.$$

That is, and replacing $\hat{\mathbf{d}}$ by $\mathbf{A}\hat{\mathbf{w}}$, the corresponding $\hat{\mathbf{w}}$ must satisfy

$$\mathbf{A}^T\mathbf{A}\hat{\mathbf{w}} = \mathbf{A}^T\mathbf{d}\,.$$

These equations always have a solution $\hat{\mathbf{w}}$. But while a solution $\hat{\mathbf{w}}$ may or may not be unique (depending on whether $\mathbf{A}$ is or is not full rank), the resulting estimate $\hat{\mathbf{d}} = \mathbf{A}\hat{\mathbf{w}}$ is always unique no matter which solution $\hat{\mathbf{w}}$ we pick. This is obvious from the geometric argument because the orthogonal projection of $\mathbf{d}$ onto the span of $\mathbf{A}$ is unique.

If $\mathbf{A}$ is assumed to be a full rank matrix then $\mathbf{A}^T\mathbf{A}$ is invertible and we can write

$$\hat{\mathbf{w}} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{d}\,. \tag{21.15}$$

21.2.1 Geometric Interpretation

The quantity $\mathbf{A}\hat{\mathbf{w}}$ provides an estimate for $\mathbf{d}$; it corresponds to the vector in the column span of $\mathbf{A}$ that is closest in Euclidean norm to the given $\mathbf{d}$. In other words,

$$\hat{\mathbf{d}} = \mathbf{A}\left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T \cdot \mathbf{d} \overset{\Delta}{=} \mathcal{P}_A \cdot \mathbf{d}\,,$$

where $\mathcal{P}_A$ denotes the projector onto the range space of $\mathbf{A}$. Figure 21.1 is a schematic representation of this geometric construction, where $\mathcal{R}(\mathbf{A})$ denotes the column span of $\mathbf{A}$.

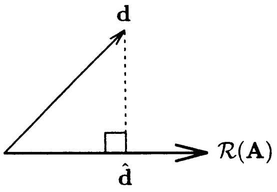

FIGURE 21.1: Geometric interpretation of the least-squares solution.

21.2.2 Statistical Interpretation

The least-squares solution also admits an important statistical interpretation. For this purpose, assume that the noise vector $\mathbf{v}$ is a realization of a vector-valued random variable that is normally distributed with zero mean and identity covariance matrix, written $\mathbf{v} \sim N[\mathbf{0}, \mathbf{I}]$. In this case, the observation vector $\mathbf{d}$ will be a realization of a vector-valued random variable that is also normally distributed with mean $\mathbf{Aw}$ and covariance matrix equal to the identity $\mathbf{I}$. This is because the random vectors are related via the additive model $\mathbf{d} = \mathbf{Aw} + \mathbf{v}$. The probability density function of the observation process $\mathbf{d}$ is then given by

$$\frac{1}{\sqrt{(2\pi)^{(N+1)}}} \cdot \exp\left\{-\frac{1}{2}(\mathbf{d} - \mathbf{Aw})^T (\mathbf{d} - \mathbf{Aw})\right\} . \tag{21.16}$$

It follows, in this case, that the least-squares estimator $\hat{\mathbf{w}}$ is also the maximum likelihood (ML) estimator because it maximizes the probability density function over $\mathbf{w}$, given an observation vector $\mathbf{d}$.

21.3 The Regularized Least-Squares Problem

A more general optimization criterion that is often used instead of (21.14) is the following

$$\min_{\mathbf{w}} \left[(\mathbf{w} - \bar{\mathbf{w}})^T \, \Pi_0^{-1}(\mathbf{w} - \bar{\mathbf{w}}) + \|\mathbf{d} - \mathbf{Aw}\|^2\right] . \tag{21.17}$$

This is still a quadratic cost function in the unknown vector $\mathbf{w}$, but it includes the additional term

$$(\mathbf{w} - \bar{\mathbf{w}})^T \, \Pi_0^{-1}(\mathbf{w} - \bar{\mathbf{w}}) ,$$

where Π_0 is a given positive-definite (weighting) matrix and $\bar{\mathbf{w}}$ is also a given vector. Choosing $\Pi_0 = \infty \cdot \mathbf{I}$ leads us back to the original expression (21.14).

A motivation for (21.17) is that the freedom in choosing Π_0 allows us to incorporate additional *a priori* knowledge into the statement of the problem. Indeed, different choices for Π_0 would indicate how confident we are about the closeness of the unknown $\mathbf{w}$ to the given vector $\bar{\mathbf{w}}$.

Assume, for example, that we set $\Pi_0 = \epsilon \cdot \mathbf{I}$, where ϵ is a very small positive number. Then the first term in the cost function (21.17) becomes dominant. It is then not hard to see that, in this case, the cost will be minimized if we choose the estimate $\hat{\mathbf{w}}$ close enough to $\bar{\mathbf{w}}$ in order to annihilate the effect of the first term. In simple words, a "small" Π_0 reflects a high confidence that $\bar{\mathbf{w}}$ is a good and close enough guess for $\mathbf{w}$. On the other hand, a "large" Π_0 indicates a high degree of uncertainty in the initial guess $\bar{\mathbf{w}}$.

One way of solving the regularized optimization problem (21.17) is to reduce it to the standard least-squares problem (21.14). This can be achieved by introducing the change of variables $\mathbf{w}' = \mathbf{w} - \bar{\mathbf{w}}$ and $\mathbf{d}' = \mathbf{d} - \mathbf{A}\bar{\mathbf{w}}$. Then (21.17) becomes

$$\min_{\mathbf{w}'} \left[(\mathbf{w}')^T \, \Pi_0^{-1}\mathbf{w}' + \left\|\mathbf{d}' - \mathbf{Aw}'\right\|^2\right] ,$$

which can be further rewritten in the equivalent form

$$\min_{\mathbf{w}'} \left\| \begin{bmatrix} \mathbf{0} \\ \mathbf{d}' \end{bmatrix} - \begin{bmatrix} \Pi_0^{-1/2} \\ \mathbf{A} \end{bmatrix} \mathbf{w}' \right\|^2 .$$

This is now of the same form as our earlier minimization problem (21.14), with the observation vector $\mathbf{d}$ in (21.14) replaced by

$$\begin{bmatrix} \mathbf{0} \\ \mathbf{d}' \end{bmatrix} ,$$

and the matrix $\mathbf{A}$ in (21.14) replaced by

$$\begin{bmatrix} \Pi_0^{-1/2} \\ \mathbf{A} \end{bmatrix} .$$

21.3.1 Geometric Interpretation

The orthogonality condition can now be used, leading to the equation

$$\begin{bmatrix} \Pi_0^{-1/2} \\ \mathbf{A} \end{bmatrix}^T \left(\begin{bmatrix} \mathbf{0} \\ \mathbf{d}' \end{bmatrix} - \begin{bmatrix} \Pi_0^{-1/2} \\ \mathbf{A} \end{bmatrix} \hat{\mathbf{w}}' \right) = \mathbf{0} \, ,$$

which can be solved for the optimal estimate $\hat{\mathbf{w}}$,

$$\hat{\mathbf{w}} = \bar{\mathbf{w}} + \left[\Pi_0^{-1} + \mathbf{A}^T \mathbf{A} \right]^{-1} \mathbf{A}^T \left[\mathbf{d} - \mathbf{A}\bar{\mathbf{w}} \right] \, . \tag{21.18}$$

TABLE 21.2 Linear Least-Squares Estimation

Optimization / Problem	Solution
$\{\mathbf{w}, \mathbf{d}\}$	
$\min_{\mathbf{w}} \|\mathbf{d} - \mathbf{A}\mathbf{w}\|^2$	$\hat{\mathbf{w}} = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{d}$
$\mathbf{A}$ full rank	
$\{\mathbf{w}, \mathbf{d}, \bar{\mathbf{w}}, \Pi_0\}$	
$\min_{\mathbf{w}} \left[(\mathbf{w} - \bar{\mathbf{w}})^T \Pi_0^{-1} (\mathbf{w} - \bar{\mathbf{w}}) + \|\mathbf{d} - \mathbf{A}\mathbf{w}\|^2 \right]$	$\hat{\mathbf{w}} = \bar{\mathbf{w}} + \left[\Pi_0^{-1} + \mathbf{A}^T \mathbf{A} \right]^{-1} \mathbf{A}^T \left[\mathbf{d} - \mathbf{A}\bar{\mathbf{w}} \right]$
Π_0 positive-definite	Min. value $= (\mathbf{d} - \mathbf{A}\bar{\mathbf{w}})^T [\mathbf{I} + \mathbf{A}\Pi_0 \mathbf{A}^T]^{-1} (\mathbf{d} - \mathbf{A}\bar{\mathbf{w}})$

Comparing with the earlier expression (21.15), we see that instead of requiring the invertibility of $\mathbf{A}^T \mathbf{A}$, we now require the invertibility of the matrix $\left[\Pi_0^{-1} + \mathbf{A}^T \mathbf{A} \right]$. This is yet another reason in favor of the modified criterion (21.17) because it allows us to relax the full rank condition on $\mathbf{A}$.

The solution (21.18) can also be reexpressed as the solution of the following linear system of equations:

$$\underbrace{\left[\Pi_0^{-1} + \mathbf{A}^T \mathbf{A} \right]}_{\Phi} (\hat{\mathbf{w}} - \bar{\mathbf{w}}) = \underbrace{\mathbf{A}^T \left[\mathbf{d} - \mathbf{A}\bar{\mathbf{w}} \right]}_{\mathbf{s}} \, , \tag{21.19}$$

where we have denoted, for convenience, the coefficient matrix by Φ and the right-hand side by $\mathbf{s}$.

Moreover, it further follows that the value of (21.17) at the minimizing solution (21.18), denoted by $\mathcal{E}_{\text{min}}$, is given by either of the following two expressions:

$$\begin{aligned} \mathcal{E}_{\text{min}} &= \|\mathbf{d} - \mathbf{A}\bar{\mathbf{w}}\|^2 - \mathbf{s}^T (\hat{\mathbf{w}} - \bar{\mathbf{w}}) & (21.20) \\ &= (\mathbf{d} - \mathbf{A}\bar{\mathbf{w}})^T \left[\mathbf{I} + \mathbf{A}\Pi_0 \mathbf{A}^T \right]^{-1} (\mathbf{d} - \mathbf{A}\bar{\mathbf{w}}) . \end{aligned}$$

Expressions (21.19) and (21.20) are often rewritten into the so-called *normal equations*:

$$\begin{bmatrix} \|\mathbf{d} - \mathbf{A}\bar{\mathbf{w}}\|^2 & \mathbf{s}^T \\ \mathbf{s} & \Phi \end{bmatrix} \begin{bmatrix} 1 \\ -(\hat{\mathbf{w}} - \bar{\mathbf{w}}) \end{bmatrix} = \begin{bmatrix} \mathcal{E}_{\text{min}} \\ \mathbf{0} \end{bmatrix} . \tag{21.21}$$

The results of this section are summarized in Table 21.2.

21.3.2 Statistical Interpretation

A statistical interpretation for the regularized problem can be obtained as follows. Given two vector-valued zero-mean random variables $\mathbf{w}$ and $\mathbf{d}$, the minimum-variance unbiased (MVU) estimator of $\mathbf{w}$ given an observation of $\mathbf{d}$ is $\hat{\mathbf{w}} = E(\mathbf{w}|\mathbf{d})$, the conditional expectation of $\mathbf{w}$ given $\mathbf{d}$. If the random variables $(\mathbf{w}, \mathbf{d})$ are jointly Gaussian, then the MVU estimator for $\mathbf{w}$ given $\mathbf{d}$ can be shown to collapse to

$$\hat{\mathbf{w}} = (E\mathbf{w}\mathbf{d}^T)\left(E\mathbf{d}\mathbf{d}^T\right)^{-1}\mathbf{d}. \tag{21.22}$$

Therefore, if $(\mathbf{w}, \mathbf{d})$ are further linearly related, say

$$\mathbf{d} = \mathbf{A}\mathbf{w} + \mathbf{v}\ , \qquad \mathbf{v} \sim N(\mathbf{0}, \mathbf{I})\ , \quad \mathbf{w} \sim N(\mathbf{0}, \Pi_0) \tag{21.23}$$

with a zero-mean noise vector $\mathbf{v}$ that is uncorrelated with $\mathbf{w}$ $(E\mathbf{w}\mathbf{v}^T = \mathbf{0})$, then the expressions for $(E\mathbf{w}\mathbf{d}^T)$ and $(E\mathbf{d}\mathbf{d}^T)$ can be evaluated as

$$E\mathbf{w}\mathbf{d}^T = E\mathbf{w}(\mathbf{A}\mathbf{w}+\mathbf{v})^T = \Pi_0\mathbf{A}^T\ , \quad E\mathbf{d}\mathbf{d}^T = \mathbf{A}\Pi_0\mathbf{A}^T + \mathbf{I}\ .$$

This shows that (21.22) evaluates to

$$\hat{\mathbf{w}} = \Pi_0\mathbf{A}^T(\mathbf{I} + \mathbf{A}\Pi_0\mathbf{A}^T)^{-1}\mathbf{d}\ . \tag{21.24}$$

By invoking the useful matrix inversion formula (for arbitrary matrices of appropriate dimensions and invertible $\mathbf{E}$ and $\mathbf{C}$):

$$(\mathbf{E} + \mathbf{BCD})^{-1} = \mathbf{E}^{-1} - \mathbf{E}^{-1}\mathbf{B}(\mathbf{D}\mathbf{E}^{-1}\mathbf{B} + \mathbf{C}^{-1})^{-1}\mathbf{D}\mathbf{E}^{-1}\ ,$$

we can rewrite expression (21.24) in the equivalent form

$$\hat{\mathbf{w}} = (\Pi_0^{-1} + \mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{d}\ . \tag{21.25}$$

This expression coincides with the regularized solution (21.18) for $\bar{\mathbf{w}} = \mathbf{0}$ (the case $\bar{\mathbf{w}} \neq \mathbf{0}$ follows from similar arguments by assuming a nonzero mean random variable $\mathbf{w}$).

Therefore, the regularized least-squares solution is the minimum variance unbiased (MVU) estimate of $\mathbf{w}$ given observations $\mathbf{d}$ that are corrupted by additive Gaussian noise as in (21.23).

21.4 The Recursive Least-Squares Problem

The recursive least-squares formulation deals with the problem of updating the solution $\hat{\mathbf{w}}$ of a least-squares problem (regularized or not) when new data are added to the matrix $\mathbf{A}$ and to the vector $\mathbf{d}$. This is in contrast to determining afresh the least-squares solution of the new problem. The distinction will become clear as we proceed in our discussions. In this section, we formulate the recursive least-squares problem as it arises in the context of adaptive filtering.

Consider a sequence of $(N+1)$ scalar data points, $\{d(j)\}_{j=0}^N$, also known as *reference* or *desired* signals, and a sequence of $(N+1)$ row vectors $\{\mathbf{u}_j^T\}_{j=0}^N$, also known as *input* signals. Each input vector $\mathbf{u}_j^T$ is a $1 \times M$ row vector whose individual entries we denote by $\{u_k(j)\}_{k=1}^M$, viz.,

$$\mathbf{u}_j^T = \begin{bmatrix} u_1(j) & u_2(j) & \ldots & u_M(j) \end{bmatrix}\ . \tag{21.26}$$

The entries of $\mathbf{u}_j$ can be regarded as the values of M input channels at time j: channels 1 through M.

Consider also a known column vector $\bar{\mathbf{w}}$ and a positive-definite weighting matrix Π_0. The objective is to determine an $M \times 1$ column vector $\mathbf{w}$, also known as the *weight vector*, so as to minimize the weighted error sum:

$$\mathcal{E}(N) = (\mathbf{w} - \bar{\mathbf{w}})^T\left[\lambda^{-(N+1)}\Pi_0\right]^{-1}(\mathbf{w} - \bar{\mathbf{w}}) + \sum_{j=0}^{N}\lambda^{N-j}\left|d(j) - \mathbf{u}_j^T\mathbf{w}\right|^2\ , \tag{21.27}$$

where λ is a positive scalar that is less than or equal to one (usually $0 \ll \lambda \leq 1$). It is often called the forgetting factor since past data is exponentially weighted less than the more recent data. The special case $\lambda = 1$ is known as the *growing memory* case, since, as the length N of the data grows, the effect of past data is not attenuated. In contrast, the *exponentially decaying memory* case ($\lambda < 1$) is more suitable for time-variant environments.

Also, and in principle, the factor $\lambda^{-(N+1)}$ that multiplies Π_0 in the error-sum expression (21.27) can be incorporated into the weighting matrix Π_0. But it is left explicit for convenience of exposition.

We further denote the individual entries of the column vector $\mathbf{w}$ by $\{w(j)\}_{j=1}^{M}$,

$$\mathbf{w} = \text{col}\{w(1), w(2), \ldots, w(M)\}\ .$$

A schematic description of the problem is shown in Fig. 21.2. At each time instant j, the inputs of the M channels are linearly combined via the coefficients of the weight vector and the resulting signal is compared with the desired signal $d(j)$. This results in a residual error $e(j) = d(j) - \mathbf{u}_j^T \mathbf{w}$, for every j, and the objective is to find a weight vector $\mathbf{w}$ in order to minimize the (exponentially weighted and regularized) squared-sum of the residual errors over an interval of time, say from $j = 0$ up to $j = N$.

The linear combiner is said to be of order M since it is determined by M coefficients $\{w(j)\}_{j=1}^{M}$.

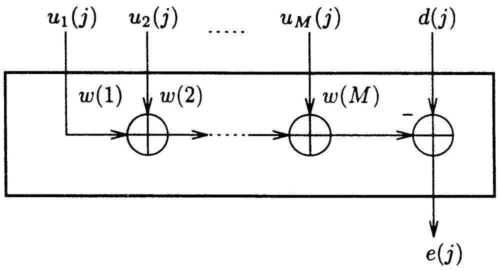

FIGURE 21.2: A linear combiner.

21.4.1 Reducing to the Regularized Form

The expression for the weighted error-sum (21.27) is a special case of the regularized cost function (21.17). To clarify this, we introduce the residual vector $\mathbf{e}_N$, the reference vector $\mathbf{d}_N$, the data matrix $\mathbf{A}_N$, and a diagonal weighting matrix Λ_N,

$$\mathbf{e}_N = \underbrace{\begin{bmatrix} d(0) \\ d(1) \\ d(2) \\ \vdots \\ d(N) \end{bmatrix}}_{\mathbf{d}_N} - \underbrace{\begin{bmatrix} u_1(0) & u_2(0) & \ldots & u_M(0) \\ u_1(1) & u_2(1) & \ldots & u_M(1) \\ u_1(2) & u_2(2) & \ldots & u_M(2) \\ \vdots & & & \vdots \\ u_1(N) & u_2(N) & \ldots & u_M(N) \end{bmatrix}}_{\mathbf{A}_N} \mathbf{w}\ ,$$

$$\Lambda_N^{1/2} = \begin{bmatrix} \left[\lambda^{\frac{1}{2}}\right]^N & & & & \\ & \left[\lambda^{\frac{1}{2}}\right]^{N-1} & & & \\ & & \ddots & & \\ & & & \left[\lambda^{\frac{1}{2}}\right]^2 & \\ & & & & 1 \end{bmatrix} .$$

We now use a subscript $_N$ to indicate that the above quantities are determined by data that is available up to time N.

With these definitions, we can write $\mathcal{E}(N)$ in the equivalent form

$$\mathcal{E}(N) = (\mathbf{w} - \bar{\mathbf{w}})^T \left[\lambda^{-(N+1)}\Pi_0\right]^{-1} (\mathbf{w} - \bar{\mathbf{w}}) + \left\|\Lambda_N^{1/2}\mathbf{e}_N\right\|^2 ,$$

which is a special case of (21.17) with

$$\Lambda_N^{1/2}\mathbf{d}_N \text{ and } \Lambda_N^{1/2}\mathbf{A}_N \tag{21.28}$$

replacing

$$\mathbf{d}_N \text{ and } \mathbf{A}_N , \tag{21.29}$$

respectively, and with $\lambda^{-(N+1)}\Pi_0$ replacing Π_0.

We therefore conclude from (21.19) that the optimal solution $\hat{\mathbf{w}}$ of (21.27) is given by

$$(\hat{\mathbf{w}} - \bar{\mathbf{w}}) = \Phi_N^{-1}\mathbf{s}_N , \tag{21.30}$$

where we have introduced

$$\Phi_N = \left[\lambda^{(N+1)}\Pi_0^{-1} + \mathbf{A}_N^T\Lambda_N\mathbf{A}_N\right] , \tag{21.31}$$

$$\mathbf{s}_N = \mathbf{A}_N^T\Lambda_N\left[\mathbf{d}_N - \mathbf{A}_N\bar{\mathbf{w}}\right] . \tag{21.32}$$

The coefficient matrix Φ_N is clearly symmetric and positive-definite.

21.4.2 Time Updates

It is straightforward to verify that Φ_N and $\mathbf{s}_N$ so defined satisfy simple time-update relations, viz.,

$$\Phi_{N+1} = \lambda\Phi_N + \mathbf{u}_{N+1}\mathbf{u}_{N+1}^T , \tag{21.33}$$

$$\mathbf{s}_{N+1} = \lambda\mathbf{s}_N + \mathbf{u}_{N+1}\left[d(N+1) - \mathbf{u}_{N+1}^T\bar{\mathbf{w}}\right], \tag{21.34}$$

with initial conditions $\Phi_{-1} = \Pi_0^{-1}$ and $\mathbf{s}_{-1} = \mathbf{0}$. Note that Φ_{N+1} and $\lambda\Phi_N$ differ only by a rank-one matrix.

The solution $\hat{\mathbf{w}}$ obtained by solving (21.30) is the optimal weight estimate based on the available data from time $i = 0$ up to time $i = N$. We shall denote it from now on by $\mathbf{w}_N$,

$$\Phi_N(\mathbf{w}_N - \bar{\mathbf{w}}) = \mathbf{s}_N .$$

The subscript $_N$ in $\mathbf{w}_N$ indicates that the data up to, and including, time N were used. This is to differentiate it from the estimate obtained by using a different number of data points.

This notational change is necessary because the main objective of the recursive least-squares (RLS) problem is to show how to update the estimate $\mathbf{w}_N$, which is based on the data up to time N, to the estimate $\mathbf{w}_{N+1}$, which is based on the data up to time $(N+1)$, without the need to solve afresh a new set of linear equations of the form

$$\Phi_{N+1}(\mathbf{w}_{N+1}-\bar{\mathbf{w}})=\mathbf{s}_{N+1}\ .$$

Such a recursive update of the weight estimate should be possible since the coefficient matrices $\lambda\Phi_N$ and Φ_{N+1} of the associated linear systems differ only by a rank-one matrix. In fact, a wide variety of algorithms has been devised for this end and our purpose in this chapter is to provide an overview of the different schemes.

Before describing these different variants, we note in passing that it follows from (21.20) that we can express the minimum value of $\mathcal{E}(N)$ in the form:

$$\mathcal{E}_{\min}(N)=\left\|\Lambda_N^{1/2}(\mathbf{d}_N-\mathbf{A}_N\bar{\mathbf{w}})\right\|^2-\mathbf{s}_N^T(\mathbf{w}_N-\bar{\mathbf{w}})\ . \tag{21.35}$$

21.5 The RLS Algorithm

The first recursive solution that we consider is the famed recursive least-squares algorithm, usually referred to as the RLS algorithm. It can be derived as follows.

Let $\mathbf{w}_{i-1}$ be the solution of an optimization problem of the form (21.27) that uses input data up to time $(i-1)$ [that is, for $N=(i-1)$]. Likewise, let $\mathbf{w}_i$ be the solution of the same optimization problem but with input data up to time i $[N=i]$.

The recursive least-squares (RLS) algorithm provides a recursive procedure that computes $\mathbf{w}_i$ from $\mathbf{w}_{i-1}$. A classical derivation follows by noting from (21.30) that the new solution $\mathbf{w}_i$ should satisfy

$$\mathbf{w}_i-\bar{\mathbf{w}}=\Phi_i^{-1}\mathbf{s}_i=\left[\lambda\Phi_{i-1}+\mathbf{u}_i\mathbf{u}_i^T\right]^{-1}\left(\lambda\mathbf{s}_{i-1}+\mathbf{u}_i\left[d(i)-\mathbf{u}_i^T\bar{\mathbf{w}}\right]\right)\ ,$$

where we have also used the time-updates for $\{\Phi_i,\mathbf{s}_i\}$.

Introduce the quantities

$$\mathbf{P}_i=\Phi_i^{-1}\ ,\quad \mathbf{g}_i=\Phi_i^{-1}\mathbf{u}_i\ . \tag{21.36}$$

Expanding the inverse of $[\lambda\Phi_{i-1}+\mathbf{u}_i\mathbf{u}_i^T]$ by using the matrix inversion formula [stated after (21.24)], and grouping terms, leads after some straightforward algebra to the RLS procedure:

- Initial conditions: $\mathbf{w}_{-1}=\bar{\mathbf{w}}$ and $\mathbf{P}_{-1}=\Pi_0$.
- Repeat for $i\geq 0$:

$$\begin{aligned}\mathbf{w}_i &= \mathbf{w}_{i-1}+\mathbf{g}_i\left[d(i)-\mathbf{u}_i^T\mathbf{w}_{i-1}\right], && (21.37)\\ \mathbf{g}_i &= \frac{\lambda^{-1}\mathbf{P}_{i-1}\mathbf{u}_i}{1+\lambda^{-1}\mathbf{u}_i^T\mathbf{P}_{i-1}\mathbf{u}_i}\ , && (21.38)\\ \mathbf{P}_i &= \lambda^{-1}\left[\mathbf{P}_{i-1}-\mathbf{g}_i\mathbf{u}_i^T\mathbf{P}_{i-1}\right]\ . && (21.39)\end{aligned}$$

- The computational complexity of the algorithm is $O(M^2)$ per iteration.

21.5.1 Estimation Errors and the Conversion Factor

With the RLS problem we associate two residuals at each time instant i: the *a priori* estimation error $e_a(i)$, defined by

$$e_a(i)=d(i)-\mathbf{u}_i^T\mathbf{w}_{i-1}\ ,$$

and the *a posteriori* estimation error $e_p(i)$, defined by

$$e_p(i) = d(i) - \mathbf{u}_i^T \mathbf{w}_i \ .$$

Comparing the expressions for $e_a(i)$ and $e_p(i)$, we see that the latter employs the most recent weight vector estimate.

If we replace $\mathbf{w}_i$ in the definition for $e_p(i)$ by its update expression (21.37), say

$$e_p(i) = d(i) - \mathbf{u}_i^T \left(\mathbf{w}_{i-1} + \mathbf{g}_i \left[d(i) - \mathbf{u}_i^T \mathbf{w}_{i-1}\right]\right) ,$$

some straightforward algebra will show that we can relate $e_p(i)$ and $e_a(i)$ via a factor $\gamma(i)$ known as the *conversion* factor:

$$e_p(i) = \gamma(i) e_a(i) \ ,$$

where $\gamma(i)$ is equal to

$$\gamma(i) = \frac{1}{1 + \lambda^{-1}\mathbf{u}_i^T \mathbf{P}_{i-1}\mathbf{u}_i} = 1 - \mathbf{u}_i^T \mathbf{P}_i \mathbf{u}_i \ . \tag{21.40}$$

That is, the *a posteriori* error is a scaled version of the *a priori* error. The scaling factor $\gamma(i)$ is defined in terms of $\{\mathbf{u}_i, \mathbf{P}_{i-1}\}$ or $\{\mathbf{u}_i, \mathbf{P}_i\}$. Note that $0 \le \gamma(i) \le 1$.

Note further that the expression for $\gamma(i)$ appears in the definition of the so-called *gain* vector $\mathbf{g}_i$ in (21.38) and, hence, we can alternatively rewrite (21.38) and (21.39) in the forms:

$$\mathbf{g}_i = \lambda^{-1}\gamma(i)\mathbf{P}_{i-1}\mathbf{u}_i \ , \tag{21.41}$$

$$\mathbf{P}_i = \lambda^{-1}\mathbf{P}_{i-1} - \gamma^{-1}(i)\mathbf{g}_i\mathbf{g}_i^T \ . \tag{21.42}$$

21.5.2 Update of the Minimum Cost

Let $\mathcal{E}_{\min}(i)$ denote the value of the minimum cost of the optimization problem (21.27) with data up to time i. It is given by an expression of the form (21.35) with N replaced by i,

$$\mathcal{E}_{\min}(i) = \left[\sum_{j=0}^{i} \lambda^{i-j} \left\| d(j) - \mathbf{u}_j^T \bar{\mathbf{w}} \right\|^2 \right] - \mathbf{s}_i^T (\mathbf{w}_i - \bar{\mathbf{w}}) \ .$$

Using the RLS update (21.37) for $\mathbf{w}_i$ in terms of $\mathbf{w}_{i-1}$, as well as the time-update (21.34) for $\mathbf{s}_i$ in terms of $\mathbf{s}_{i-1}$, we can derive the following time-update for the minimum cost:

$$\mathcal{E}_{\min}(i) = \lambda \mathcal{E}_{\min}(i-1) + e_p(i) e_a(i) \ , \tag{21.43}$$

where $\mathcal{E}_{\min}(i-1)$ denotes the value of the minimum cost of the same optimization problem (21.27) but with data up to time $(i-1)$.

21.6 RLS Algorithms in Array Forms

As mentioned in the introduction, we intend to stress the array formulations of the RLS solution due to their intrinsic advantages:

- They are easy to implement as a sequence of elementary rotations on arrays of numbers.
- They are modular and parallelizable.
- They have better numerical properties than the classical RLS description.

21.6.1 Motivation

Note from (21.39) that the RLS solution propagates the variable $\mathbf{P}_i$ as the difference of two quantities. This variable should be positive-definite. But due to roundoff errors, however, the update (21.39) may not guarantee the positive-definiteness of $\mathbf{P}_i$ at all times i. This problem can be ameliorated by using the so-called array formulations. These alternative forms propagate square-root factors of either $\mathbf{P}_i$ or $\mathbf{P}_i^{-1}$, namely, $\mathbf{P}_i^{1/2}$ or $\mathbf{P}_i^{-1/2}$, rather than $\mathbf{P}_i$ itself. By squaring $\mathbf{P}_i^{1/2}$, for example, we can always recover a matrix $\mathbf{P}_i$ that is more likely to be positive-definite than the matrix obtained via (21.39),

$$\mathbf{P}_i = \mathbf{P}_i^{1/2}\mathbf{P}_i^{T/2} .$$

21.6.2 A Very Useful Lemma

The derivation of the array variants of the RLS algorithm relies on a very useful matrix result that encounters applications in many other scenarios as well. For this reason, we not only state the result but also provide one simple proof.

LEMMA 21.1 Given two $n \times m$ $(n \leq m)$ matrices $\mathbf{A}$ and $\mathbf{B}$, then $\mathbf{A}\mathbf{A}^{\mathbf{T}} = \mathbf{B}\mathbf{B}^{\mathbf{T}}$ if, and only if, there exists an $m \times m$ orthogonal matrix Θ $(\Theta\Theta^{\mathbf{T}} = \mathbf{I}_m)$ such that $\mathbf{A} = \mathbf{B}\Theta$.

PROOF 21.1 One implication is immediate. If there exists an orthogonal matrix Θ such that $\mathbf{A} = \mathbf{B}\Theta$ then

$$\mathbf{A}\mathbf{A}^T = (\mathbf{B}\Theta)(\mathbf{B}\Theta)^T = \mathbf{B}(\Theta\Theta^T)\mathbf{B}^T = \mathbf{B}\mathbf{B}^T .$$

One proof for the converse implication follows by invoking the singular value decompositions of the matrices $\mathbf{A}$ and $\mathbf{B}$,

$$\begin{aligned} \mathbf{A} &= \mathbf{U}_A \begin{bmatrix} \Sigma_A & \mathbf{0} \end{bmatrix} \mathbf{V}_A^T , \\ \mathbf{B} &= \mathbf{U}_B \begin{bmatrix} \Sigma_B & \mathbf{0} \end{bmatrix} \mathbf{V}_B^T , \end{aligned}$$

where $\mathbf{U}_A$ and $\mathbf{U}_B$ are $n \times n$ orthogonal matrices, $\mathbf{V}_A$ and $\mathbf{V}_B$ are $m \times m$ orthogonal matrices, and Σ_A and Σ_B are $n \times n$ diagonal matrices with nonnegative (ordered) entries.

The squares of the diagonal entries of Σ_A (Σ_B) are the eigenvalues of $\mathbf{A}\mathbf{A}^T$ $(\mathbf{B}\mathbf{B}^T)$. Moreover, $\mathbf{U}_A$ $(\mathbf{U}_B)$ are constructed from an orthonormal basis for the right eigenvectors of $\mathbf{A}\mathbf{A}^{\mathbf{T}}$ $(\mathbf{B}\mathbf{B}^{\mathbf{T}})$.

Hence, it follows from the identity $\mathbf{A}\mathbf{A}^{\mathbf{T}} = \mathbf{B}\mathbf{B}^{\mathbf{T}}$ that we have $\Sigma_A = \Sigma_B$ and we can choose $\mathbf{U}_A = \mathbf{U}_B$. Let $\Theta = \mathbf{V}_B\mathbf{V}_A^T$. We then obtain $\Theta\Theta^{\mathbf{T}} = \mathbf{I}_m$ and $\mathbf{B}\Theta = \mathbf{A}$.

21.6.3 The Inverse QR Algorithm

We now employ the above result to derive an array form of the RLS algorithm that is known as the inverse QR algorithm.

Let $\mathbf{P}_{i-1}^{1/2}$ denote a (preferably lower triangular) square-root factor of $\mathbf{P}_{i-1}$, i.e., any matrix that satisfies

$$\mathbf{P}_{i-1} = \mathbf{P}_{i-1}^{1/2}\,\mathbf{P}_{i-1}^{T/2} .$$

[The triangular square-root factor of a symmetric positive-definite matrix is also known as the Cholesky factor].

Now note that the RLS recursions (21.38) and (21.39) can be expressed in factored form as follows:

$$\begin{bmatrix} 1 & \frac{1}{\sqrt{\lambda}}\mathbf{u}_i^T\mathbf{P}_{i-1}^{1/2} \\ \mathbf{0} & \frac{1}{\sqrt{\lambda}}\mathbf{P}_{i-1}^{1/2} \end{bmatrix} \begin{bmatrix} 1 & \mathbf{0}^T \\ \frac{1}{\sqrt{\lambda}}\mathbf{P}_{i-1}^{T/2}\mathbf{u}_i & \frac{1}{\sqrt{\lambda}}\mathbf{P}_{i-1}^{T/2} \end{bmatrix}$$
$$= \begin{bmatrix} \gamma^{-1/2}(i) & \mathbf{0}^T \\ \mathbf{g}_i\gamma^{-1/2}(i) & \mathbf{P}_i^{1/2} \end{bmatrix} \begin{bmatrix} \gamma^{-1/2}(i) & \mathbf{g}_i^T\gamma^{-1/2}(i) \\ \mathbf{0} & \mathbf{P}_i^{T/2} \end{bmatrix} .$$

To verify that this is indeed the case, we simply multiply the factors and compare terms on both sides of the equality.

The point to note is that the above equality fits nicely into the statement of the previous lemma by taking

$$\mathbf{A} = \begin{bmatrix} 1 & \frac{1}{\sqrt{\lambda}}\mathbf{u}_i^T\mathbf{P}_{i-1}^{1/2} \\ \mathbf{0} & \frac{1}{\sqrt{\lambda}}\mathbf{P}_{i-1}^{1/2} \end{bmatrix} \tag{21.44}$$

and

$$\mathbf{B} = \begin{bmatrix} \gamma^{-1/2}(i) & \mathbf{0}^T \\ \mathbf{g}_i\gamma^{-1/2}(i) & \mathbf{P}_i^{1/2} \end{bmatrix} . \tag{21.45}$$

We therefore conclude that there should exist an orthogonal matrix Θ_i that relates the arrays **A** and **B** in the form

$$\begin{bmatrix} 1 & \frac{1}{\sqrt{\lambda}}\mathbf{u}_i^T\mathbf{P}_{i-1}^{1/2} \\ \mathbf{0} & \frac{1}{\sqrt{\lambda}}\mathbf{P}_{i-1}^{1/2} \end{bmatrix} \Theta_i = \begin{bmatrix} \gamma^{-1/2}(i) & \mathbf{0}^T \\ \mathbf{g}_i\gamma^{-1/2}(i) & \mathbf{P}_i^{1/2} \end{bmatrix} .$$

That is, there should exist an orthogonal Θ_i that transforms the prearray **A** into the postarray **B**.

Note that the prearray contains quantities that are available at step i, namely $\{\mathbf{u}_i, \mathbf{P}_{i-1}^{1/2}\}$, while the postarray provides the (normalized) gain vector $\mathbf{g}_i\gamma^{-1/2}(i)$, which is needed to update the weight vector estimate $\mathbf{w}_{i-1}$ into $\mathbf{w}_i$, as well as the square-root factor of the variable $\mathbf{P}_i$, which is needed to form the prearray for the next iteration.

But how do we determine Θ_i? The answer highlights a remarkable property of array algorithms. We do not really need to know or determine Θ_i explicitly!

To clarify this point, we first remark from the expressions (21.44) and (21.45) for the pre and postarrays that Θ_i is an orthogonal matrix that takes an array of numbers of the form (assuming a vector $\mathbf{u}_i$ of dimension $M = 3$)

$$\left[\begin{array}{cccc} 1 & x & x & x \\ \hline 0 & x & 0 & 0 \\ 0 & x & x & 0 \\ 0 & x & x & x \end{array}\right] \tag{21.46}$$

and transforms it to the form

$$\left[\begin{array}{cccc} x & 0 & 0 & 0 \\ \hline x & x & 0 & 0 \\ x & x & x & 0 \\ x & x & x & x \end{array}\right] . \tag{21.47}$$

That is, Θ_i annihilates all the entries of the top row of the prearray (except for the left-most entry).

Now assume we form the prearray **A** in (21.44) and choose any Θ_i (say as a sequence of elementary rotations) so as to reduce **A** to the triangular form (21.47), that is, in order to annihilate the desired entries in the top row.

Let us denote the resulting entries of the postarray arbitrarily as:

$$\begin{bmatrix} 1 & \frac{1}{\sqrt{\lambda}}\mathbf{u}_i^T\mathbf{P}_{i-1}^{1/2} \\ \mathbf{0} & \frac{1}{\sqrt{\lambda}}\mathbf{P}_{i-1}^{1/2} \end{bmatrix}\Theta_i = \begin{bmatrix} a & \mathbf{0}^T \\ \mathbf{b} & \mathbf{C} \end{bmatrix}, \tag{21.48}$$

where $\{a, \mathbf{b}, \mathbf{C}\}$ are quantities that we wish to identify [a is a scalar, $\mathbf{b}$ is a column vector, and $\mathbf{C}$ is a lower triangular matrix]. The claim is that by constructing Θ_i in this way (i.e., by simply requiring that it achieves the desired zero pattern in the postarray), the resulting quantities $\{a, \mathbf{b}, \mathbf{C}\}$ will be meaningful and can in fact be identified with the quantities in the postarray $\mathbf{B}$.

To verify that the quantities $\{a, \mathbf{b}, \mathbf{C}\}$ can indeed be identified with $\{\gamma^{-1/2}(i), \mathbf{g}_i\gamma^{-1/2}(i), \mathbf{P}_i^{1/2}\}$, we proceed by squaring both sides of (21.48),

$$\begin{bmatrix} 1 & \frac{1}{\sqrt{\lambda}}\mathbf{u}_i^T\mathbf{P}_{i-1}^{1/2} \\ \mathbf{0} & \frac{1}{\sqrt{\lambda}}\mathbf{P}_{i-1}^{1/2} \end{bmatrix} \underbrace{\Theta_i\Theta_i^T}_{\mathbf{I}} \begin{bmatrix} 1 & \mathbf{0} \\ \frac{1}{\sqrt{\lambda}}\mathbf{P}_{i-1}^{T/2}\mathbf{u}_i & \frac{1}{\sqrt{\lambda}}\mathbf{P}_{i-1}^{T/2} \end{bmatrix} = \begin{bmatrix} a & \mathbf{0}^T \\ \mathbf{b} & \mathbf{C} \end{bmatrix}\begin{bmatrix} a & \mathbf{b}^T \\ \mathbf{0} & \mathbf{C}^T \end{bmatrix},$$

and comparing terms on both sides of the equality to get the identities:

$$\begin{aligned} a^2 &= 1 + \lambda^{-1}\mathbf{u}_i^T\mathbf{P}_{i-1}\mathbf{u}_i = \gamma^{-1}(i), \\ \mathbf{b}a &= \lambda^{-1}\mathbf{P}_{i-1}\mathbf{u}_i = \mathbf{g}_i\gamma^{-1}(i), \\ \mathbf{C}\mathbf{C}^T &= \lambda^{-1}\mathbf{P}_{i-1} - \mathbf{b}\mathbf{b}^T = \lambda^{-1}\mathbf{P}_{i-1} - \gamma^{-1}(i)\mathbf{g}_i\mathbf{g}_i^T. \end{aligned}$$

Hence, as desired, we can make the identifications

$$a = \gamma^{-1/2}(i), \quad \mathbf{b} = \mathbf{g}_i\gamma^{-1/2}(i), \quad \mathbf{C} = \mathbf{P}_i^{1/2}.$$

In summary, we have established the validity of an array alternative to the RLS algorithm, known as the inverse QR algorithm (also as square-root RLS). It is listed in Table 21.3. The recursions are known as *inverse* QR since they propagate $\mathbf{P}_i^{1/2}$, which is a square-root factor of the inverse of the coefficient matrix Φ_i.

TABLE 21.3 The Inverse QR Algorithm

Initialization. Start with $\mathbf{w}_{-1} = \bar{\mathbf{w}}$ and $\mathbf{P}_{-1}^{1/2} = \Pi_0^{1/2}$

- Repeat for each time instant $i \geq 0$:

$$\begin{bmatrix} 1 & \frac{1}{\sqrt{\lambda}}\mathbf{u}_i^T\mathbf{P}_{i-1}^{1/2} \\ \mathbf{0} & \frac{1}{\sqrt{\lambda}}\mathbf{P}_{i-1}^{1/2} \end{bmatrix}\Theta_i = \begin{bmatrix} \gamma^{-1/2}(i) & \mathbf{0}^T \\ \mathbf{g}_i\gamma^{-1/2}(i) & \mathbf{P}_i^{1/2} \end{bmatrix}$$

where Θ_i is any orthogonal rotation that produces the zero pattern in the postarray.

The weight-vector estimate is updated via

$$\mathbf{w}_i = \mathbf{w}_{i-1} + \left[\frac{\mathbf{g}_i}{\gamma^{1/2}(i)}\right]\left[\frac{1}{\gamma^{1/2}(i)}\right]^{-1}\left[d(i) - \mathbf{u}_i^T\mathbf{w}_{i-1}\right]$$

where the quantities $\{\gamma^{-1/2}(i), \mathbf{g}_i\gamma^{-1/2}(i)\}$ are read from the entries of the postarray.

The computational cost is $O(M^2)$ per iteration.

21.6.4 The QR Algorithm

The RLS recursion (21.39) and the inverse QR recursion of Table 21.3 propagate the variable $\mathbf{P}_i$ or a square-root factor of it. The starting condition for both algorithms is therefore dependent on the weighting matrix Π_0 or its square-root factor $\Pi_0^{1/2}$.

This situation becomes inconvenient when the initial condition Π_0 assumes relatively large values, say $\Pi_0 = \sigma \mathbf{I}$ with $\sigma \gg 1$. A particular instance arises, for example, when we take $\sigma \to \infty$ in which case the regularized least-squares problem (21.27) reduces to a standard least-squares problem of the form

$$\min_{\mathbf{w}} \left[\mathcal{E}(N) = \sum_{j=0}^{N} \lambda^{N-j} |d(j) - \mathbf{u}_j^T \mathbf{w}|^2 \right] . \tag{21.49}$$

For such problems, it is preferable to propagate the inverse of the variable $\mathbf{P}_i$ rather than $\mathbf{P}_i$ itself. Recall that the inverse of $\mathbf{P}_i$ is Φ_i since we have defined earlier $\mathbf{P}_i = \Phi_i^{-1}$.

The QR algorithm is a recursive procedure that propagates a square-root factor of Φ_i. Its validity can be verified in much the same way as we did for the inverse QR algorithm. We form a prearray of numbers and then choose a sequence of rotations that induces a desired zero pattern in the postarray. Then by squaring and comparing terms on both sides of an equality we can identify the resulting entries of the postarray as meaningful quantities in the RLS context. For this reason, we shall be brief and only highlight the main points.

Let $\Phi_{i-1}^{1/2}$ denote a square-root factor (preferably lower-triangular) of Φ_{i-1}, $\Phi_{i-1} = \Phi_{i-1}^{1/2} \, \Phi_{i-1}^{T/2}$, and define, for notational convenience, the quantity

$$\mathbf{q}_{i-1} = \Phi_{i-1}^{T/2} \, \mathbf{w}_{i-1} \, . \tag{21.50}$$

At time $(i-1)$ we form the prearray of numbers

$$\mathbf{A} = \begin{bmatrix} \sqrt{\lambda}\Phi_{i-1}^{1/2} & \mathbf{u}_i \\ \sqrt{\lambda}\mathbf{q}_{i-1}^T & d(i) \\ \mathbf{0}^T & 1 \end{bmatrix} ,$$

whose entries have the following pattern (shown for $M = 3$):

$$\mathbf{A} = \left[\begin{array}{cccc} x & 0 & 0 & x \\ x & x & 0 & x \\ x & x & x & x \\ \hline x & x & x & x \\ \hline 0 & 0 & 0 & 1 \end{array} \right] .$$

Now implement an orthogonal transformation Θ_i that reduces $\mathbf{A}$ to the form

$$\mathbf{B} = \left[\begin{array}{cccc} x & 0 & 0 & 0 \\ x & x & 0 & 0 \\ x & x & x & 0 \\ \hline x & x & x & x \\ \hline x & x & x & x \end{array} \right] = \begin{bmatrix} \mathbf{C} & \mathbf{0} \\ \mathbf{b}^T & a \\ \mathbf{h}^T & f \end{bmatrix} ,$$

where the quantities $\{\mathbf{C}, \mathbf{b}, \mathbf{h}, a, f\}$ need to be identified. By comparing terms on both sides of the equality

$$\begin{bmatrix} \sqrt{\lambda}\Phi_{i-1}^{1/2} & \mathbf{u}_i \\ \sqrt{\lambda}\mathbf{q}_{i-1}^T & d(i) \\ \mathbf{0}^T & 1 \end{bmatrix} \underbrace{\Theta_i \Theta_i^T}_{\mathbf{I}} \begin{bmatrix} \sqrt{\lambda}\Phi_{i-1}^{1/2} & \mathbf{u}_i \\ \sqrt{\lambda}\mathbf{q}_{i-1}^T & d(i) \\ \mathbf{0}^T & 1 \end{bmatrix}^T = \begin{bmatrix} \mathbf{C} & \mathbf{0} \\ \mathbf{b}^T & a \\ \mathbf{h}^T & f \end{bmatrix} \begin{bmatrix} \mathbf{C} & \mathbf{0} \\ \mathbf{b}^T & a \\ \mathbf{h}^T & f \end{bmatrix}^T ,$$

we can make the identifications:

$$\mathbf{C} = \Phi_i^{1/2},\ \mathbf{b}^T = \mathbf{q}_i^T,\ \mathbf{h}^T = \mathbf{u}_i^T \Phi_i^{-T/2},$$

$$a = e_a(i)\gamma^{1/2}(i), \quad f = \gamma^{1/2}(i)\ ,$$

where $e_a(i) = d(i) - \mathbf{u}_i^T \mathbf{w}_{i-1}$ is the *a priori* estimation error. This derivation establishes the so-called QR algorithm (listed in Table 21.4).

TABLE 21.4 The QR Algorithm

Initialization. Start with $\mathbf{w}_{-1} = \bar{\mathbf{w}},\ \Phi_{-1}^{1/2} = \Pi_0^{-T/2},\ \mathbf{q}_{-1} = \Pi_0^{-1/2}\bar{\mathbf{w}}$.

- Repeat for each time instant $i \geq 0$:

$$\begin{bmatrix} \sqrt{\lambda}\Phi_{i-1}^{1/2} & \mathbf{u}_i^T \\ \sqrt{\lambda}\mathbf{q}_{i-1}^T & d(i) \\ \mathbf{0}^T & 1 \end{bmatrix} \Theta_i = \begin{bmatrix} \Phi_i^{1/2} & \mathbf{0} \\ \mathbf{q}_i^T & e_a(i)\gamma^{1/2}(i) \\ \mathbf{u}_i^T \Phi_i^{-T/2} & \gamma^{1/2}(i) \end{bmatrix}$$

where Θ_i is any orthogonal rotation that produces the zero pattern in the postarray.

The weight-vector estimate can be obtained by solving the triangular linear system of equations

$$\Phi_i^{T/2}\mathbf{w}_i = \mathbf{q}_i$$

where the quantities $\{\Phi_i^{1/2}, \mathbf{q}_i\}$ are available from the entries of the postarray.

The computational complexity is still $O(M^2)$ per iteration.

The QR solution determines the weight-vector estimate $\mathbf{w}_i$ by solving a triangular linear system of equations, e.g., via back-substitution. A major drawback of a back-substitution step is that it involves serial operations and, therefore, does not lend itself to a fully parallelizable implementation.

An alternative procedure for computing the estimate $\mathbf{w}_i$ can be obtained by appending one more block row to the arrays of the QR algorithm, leading to the equations:

$$\left[\begin{array}{cc} \sqrt{\lambda}\Phi_{i-1}^{1/2} & \mathbf{u}_i \\ \sqrt{\lambda}\mathbf{q}_{i-1}^T & d(i) \\ \mathbf{0}^T & 1 \\ \hline \frac{1}{\sqrt{\lambda}}\Phi_{i-1}^{-T/2} & \mathbf{0} \end{array}\right] \Theta_i = \left[\begin{array}{cc} \Phi_i^{1/2} & \mathbf{0} \\ \mathbf{q}_i^T & e_a(i)\gamma^{1/2}(i) \\ \mathbf{u}_i^T \Phi_i^{-T/2} & \gamma^{1/2}(i) \\ \hline \Phi_i^{-T/2} & -\mathbf{g}_i\gamma^{-1/2}(i) \end{array}\right]. \tag{21.51}$$

In this case, the last row of the postarray provides the gain vector $\mathbf{g}_i$ that can be used to update the weight-vector estimate as follows:

$$\mathbf{w}_i = \mathbf{w}_{i-1} + \left[\frac{\mathbf{g}_i}{\gamma^{1/2}(i)}\right]\left[e_a(i)\gamma^{1/2}(i)\right]\ .$$

Note, however, that the pre- and postarrays now propagate both $\Phi_i^{1/2}$ and its inverse, which may lead to numerical difficulties.

21.7 Fast Transversal Algorithms

The earlier recursive least-squares solutions require $O(M^2)$ floating point operations per iteration, where M is the size of the input vector $\mathbf{u}_i$.

$$\mathbf{u}_i^T = \left[\begin{array}{cccc} u_1(i) & u_2(i) & \ldots & u_M(i) \end{array}\right] .$$

It often happens in practice that the entries of $\mathbf{u}_i$ are time-shifted versions of each other. More explicitly, if we denote the value of the first entry of $\mathbf{u}_i$ by $u(i)$ [instead of $u_1(i)$], then $\mathbf{u}_i$ will have the form

$$\mathbf{u}_i^T = \left[\begin{array}{cccc} u(i) & u(i-1) & \ldots & u(i-M+1) \end{array}\right] . \tag{21.52}$$

This has the pictorial representation shown in Fig. 21.3. The term z^{-1} represents a unit-time delay. The structure that takes $u(j)$ as an input and provides the inner product $\sum_{k=1}^{M} u(j+1-k)w(k)$ as an output is known as a transversal or FIR (finite-impulse response) filter.

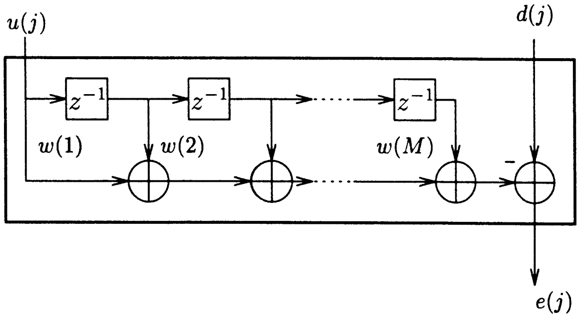

FIGURE 21.3: A linear combiner with shift structure in the input channels.

The shift structure in $\mathbf{u}_i$ can be exploited in order to derive fast variants to the RLS solution that would require $O(M)$ operations per iteration rather than $O(M^2)$. This can be achieved by showing that, in this case, the $M \times M$ variables $\mathbf{P}_i$ that are needed in the RLS recursion (21.39) exhibit certain matrix structure that allows us to replace the RLS recursions by an alternative set of recursions that we now motivate.

21.7.1 The Prewindowed Case

We first assume that no input data is available prior to and including time $i = 0$. That is, $u(i) = 0$ for $i \leq 0$. In this case, the values at time 0 of the variables $\{\mathbf{u}_i, \mathbf{g}_i, \gamma(i), \mathbf{P}_i\}$ become:

$$\mathbf{u}_0 = \mathbf{0},\ \ \mathbf{g}_0 = \mathbf{0},\ \ \gamma(0) = 1,\ \ \mathbf{P}_0 \ = \lambda^{-1}\mathbf{P}_{-1} = \lambda^{-1}\Pi_0 \ .$$

It then follows that the following equality holds:

$$\left[\begin{array}{cc} \mathbf{P}_0 & \mathbf{0} \\ \mathbf{0}^T & 0 \end{array}\right] - \left[\begin{array}{cc} 0 & \mathbf{0}^T \\ \mathbf{0} & \mathbf{P}_{-1} \end{array}\right] = \left[\begin{array}{cc} \lambda^{-1}\Pi_0 & \mathbf{0} \\ \mathbf{0}^T & 0 \end{array}\right] - \left[\begin{array}{cc} 0 & \mathbf{0}^T \\ \mathbf{0} & \Pi_0 \end{array}\right]$$

Note that we have embedded $\mathbf{P}_0$ and $\mathbf{P}_{-1}$ into larger matrices [of size $(M+1) \times (M+1)$ each] by adding one zero row and one zero column. This embedding will allow us to suggest a suitable choice for the initial weighting matrix Π_0 in order to enforce a low-rank difference matrix on the right-hand side of the above expression. In so doing, we guarantee that $(\mathbf{P}_0 \oplus 0)$ can be obtained from $(0 \oplus \mathbf{P}_{-1})$ via a low rank update.

Strikingly enough, the argument will further show that because of the shift structure in the input vectors $\mathbf{u}_i$, if this low-rank property holds for the initial time instant then it also holds for the successive time instants! Consequently, the successive matrices $(\mathbf{P}_i \oplus 0)$ will also be low rank modifications of earlier matrices $(0 \oplus \mathbf{P}_{i-1})$.

In this way, a fast procedure for updating the $\mathbf{P}_i$ can be developed by replacing the propagation of $\mathbf{P}_i$ via (21.39) by a recursion that instead propagates the low rank factors that generate the $\mathbf{P}_i$. We will verify that this procedure also allows us to update the weight vector estimates rapidly (in $O(M)$ operations).

21.7.2 Low-Rank Property

Assume we choose Π_0 in the special diagonal form

$$\Pi_0 = \delta \cdot \text{ diagonal } \{\lambda^2, \lambda^3, \ldots, \lambda^{M+1}\}, \tag{21.53}$$

where δ is a positive quantity (usually much larger than one, $\delta \gg 1$). In this case, we are led to a rank-two difference of the form

$$\begin{bmatrix} \lambda^{-1}\Pi_0 & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix} - \begin{bmatrix} 0 & \mathbf{0}^T \\ \mathbf{0} & \Pi_0 \end{bmatrix} = \delta \cdot \lambda \cdot \begin{bmatrix} 1 & & \\ & \mathbf{0} & \\ & & -\lambda^M \end{bmatrix},$$

which can be factored as

$$\begin{bmatrix} \mathbf{P}_0 & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix} - \begin{bmatrix} 0 & \mathbf{0}^T \\ \mathbf{0} & \mathbf{P}_{-1} \end{bmatrix} = \lambda \cdot \bar{\text{Ł}}_0 \mathbf{S}_0 \bar{\text{Ł}}_0^T , \tag{21.54}$$

where $\bar{\text{Ł}}_0$ is $(M+1) \times 2$ and $\mathbf{S}_0$ is a 2×2 signature matrix that are given by

$$\bar{\text{Ł}}_0 = \sqrt{\delta} \cdot \begin{bmatrix} 1 & 0 \\ \mathbf{0} & \mathbf{0} \\ 0 & \lambda^{\frac{M}{2}} \end{bmatrix}, \quad \mathbf{S}_0 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}.$$

21.7.3 A Fast Array Algorithm

We now argue by induction, and by using the shift property of the input vectors $\mathbf{u}_i$, that if the low-rank property holds at a certain time instant i, say

$$\begin{bmatrix} \mathbf{P}_i & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix} - \begin{bmatrix} 0 & \mathbf{0}^T \\ \mathbf{0} & \mathbf{P}_{i-1} \end{bmatrix} = \lambda \cdot \bar{\text{Ł}}_i \mathbf{S}_i \bar{\text{Ł}}_i^T , \tag{21.55}$$

then three important facts hold:

- The low-rank property also holds at time $i+1$, say

$$\begin{bmatrix} \mathbf{P}_{i+1} & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix} - \begin{bmatrix} 0 & \mathbf{0}^T \\ \mathbf{0} & \mathbf{P}_i \end{bmatrix} = \lambda \cdot \bar{\text{Ł}}_{i+1} \mathbf{S}_{i+1} \bar{\text{Ł}}_{i+1}^T ,$$

- There exists an array algorithm that updates $\bar{\text{Ł}}_i$ to $\bar{\text{Ł}}_{i+1}$. Moreover, the algorithm also provides the gain vector $\mathbf{g}_i$ that is needed to update the weight-vector estimate in the RLS solution.

- The signature matrices $\{\mathbf{S}_i, \mathbf{S}_{i+1}\}$ are equal! That is, all successive low-rank differences have the same signature matrix as the initial difference and, hence,

$$\mathbf{S}_i = \mathbf{S}_0 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad \text{for all } \; i \; .$$

To verify these claims, consider (21.55) and form the prearray

$$\mathbf{A} = \begin{bmatrix} \gamma^{-1/2}(i) & \begin{bmatrix} u(i+1) & \mathbf{u}_i^T \end{bmatrix} \bar{L}_i \\ \begin{bmatrix} 0 \\ \mathbf{g}_i \gamma^{-1/2}(i) \end{bmatrix} & \bar{L}_i \end{bmatrix} .$$

For $M = 3$, the prearray has the following generic form (recall that $\bar{L}_i$ is $(M+1) \times 2$):

$$\mathbf{A} = \left[\begin{array}{ccc} x & x & x \\ \hline 0 & x & x \\ x & x & x \\ x & x & x \\ x & x & x \end{array} \right] .$$

Now let Θ_i be a matrix that satisfies

$$\Theta_i \begin{bmatrix} 1 & & \\ & 1 & \\ & & -1 \end{bmatrix} \Theta_i^T = \begin{bmatrix} 1 & & \\ & 1 & \\ & & -1 \end{bmatrix} = \begin{bmatrix} 1 & \\ & \mathbf{S}_i \end{bmatrix} ,$$

and such that it transforms $\mathbf{A}$ into the form

$$\mathbf{B} = \left[\begin{array}{ccc} x & 0 & 0 \\ \hline x & x & x \\ x & x & x \\ x & x & x \\ x & x & x \end{array} \right] = \begin{bmatrix} a & \mathbf{0}^T \\ \mathbf{b} & \mathbf{C} \end{bmatrix} .$$

That is, Θ_i annihilates two entries in the top row of the prearray. This can be achieved by employing a circular rotation that pivots with the left-most entry of the first row and annihilates its second entry. We then employ a hyperbolic rotation that pivots again with the left-most entry and annihilates the last entry of the top row.

The unknown entries $\{a, \mathbf{b}, \mathbf{C}\}$ can be identified by resorting to the same technique that we employed earlier during the derivation of the QR and inverse QR algorithms. By comparing entries on both sides of the equality

$$\mathbf{A} \begin{bmatrix} 1 & \\ & \mathbf{S}_i \end{bmatrix} \mathbf{A}^T = \begin{bmatrix} a & \mathbf{0}^T \\ \mathbf{b} & \mathbf{C} \end{bmatrix} \begin{bmatrix} 1 & \\ & \mathbf{S}_i \end{bmatrix} \begin{bmatrix} a & \mathbf{0}^T \\ \mathbf{b} & \mathbf{C} \end{bmatrix}^T$$

we obtain several equalities. For example, by equating the $(1, 1)$ entries we obtain the following relation:

$$\gamma^{-1}(i) + \begin{bmatrix} u(i+1) & \mathbf{u}_i^T \end{bmatrix} \bar{L}_i \mathbf{S}_i \bar{L}_i^T \begin{bmatrix} u(i+1) \\ \mathbf{u}_i \end{bmatrix} = a^2 . \tag{21.56}$$

By using (21.55) for $\bar{L}_i \mathbf{S}_i \bar{L}_i$ and by noting that we can rewrite the vector $\begin{bmatrix} u(i+1) & \mathbf{u}_i^T \end{bmatrix}$ in two equivalent forms (due to its shift structure):

$$\begin{bmatrix} u(i+1) & \mathbf{u}_i^T \end{bmatrix} = \begin{bmatrix} \mathbf{u}_{i+1}^T & u(i-M+1) \end{bmatrix} , \tag{21.57}$$

we readily conclude that (21.56) collapses to

$$\gamma^{-1}(i) + \lambda^{-1}\mathbf{u}_{i+1}^T\mathbf{P}_i\mathbf{u}_{i+1} - \lambda^{-1}\mathbf{u}_i^T\mathbf{P}_{i-1}\mathbf{u}_i = a^2\ .$$

But $\gamma^{-1}(i) = 1 + \lambda^{-1}\mathbf{u}_i^T\mathbf{P}_{i-1}\mathbf{u}_i$. Therefore,

$$a^2 = 1 + \lambda^{-1}\mathbf{u}_{i+1}^T\mathbf{P}_i\mathbf{u}_{i+1} = \gamma^{-1}(i+1),$$

which shows that we can identify a as

$$a = \gamma^{-1/2}(i+1).$$

A similar argument allows us to identify **b**. By comparing the (2, 1) entries we obtain

$$a\mathbf{b} = \begin{bmatrix} 0 \\ \mathbf{g}_i\gamma^{-1}(i) \end{bmatrix} + \bar{Ł}_i\mathbf{S}_i\bar{Ł}_i^T \begin{bmatrix} u(i+1) \\ \mathbf{u}_i \end{bmatrix}\ . \tag{21.58}$$

Again, by (21.55) for $\bar{Ł}_i\mathbf{S}_i\bar{Ł}_i^T$, (21.57) for the vector $\begin{bmatrix} u(i+1) & \mathbf{u}_i^T \end{bmatrix}$, and by noting from the definition of $\mathbf{g}_i$ that

$$\begin{bmatrix} 0 \\ \mathbf{g}_i\gamma^{-1}(i) \end{bmatrix} = \begin{bmatrix} 0 \\ \lambda^{-1}\mathbf{P}_{i-1}\mathbf{u}_i \end{bmatrix}$$

we obtain

$$\mathbf{b} = \begin{bmatrix} \mathbf{g}_{i+1}\gamma^{-1/2}(i+1) \\ 0 \end{bmatrix}\ .$$

Finally, for the last term **C** we compare the (2, 2) entries to obtain

$$\mathbf{C}\mathbf{S}_i\mathbf{C}^T = \begin{bmatrix} \mathbf{P}_{i+1} & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix} - \begin{bmatrix} 0 & \mathbf{0}^T \\ \mathbf{0} & \mathbf{P}_i \end{bmatrix}\ .$$

The difference on the right-hand side is by definition $\lambda\bar{Ł}_{i+1}\mathbf{S}_{i+1}\bar{Ł}_{i+1}^T$. This shows that we can make the identifications

$$\mathbf{C} = \sqrt{\lambda}\cdot\bar{Ł}_{i+1}\ ,\quad \mathbf{S}_{i+1} = \mathbf{S}_i\ .$$

In summary, we have established the validity of the array algorithm shown in Table 21.5, which minimizes the cost function (21.27) in the prewindowed case and for the special choice of Π_0 in (21.53).

Note that this fast procedure computes the required gain vectors $\mathbf{g}_i$ without explicitly evaluating the matrices $\mathbf{P}_i$. Instead, the low-rank factors $\bar{Ł}_i$ are propagated, which explains the lower computational requirements.

21.7.4 The Fast Transversal Filter

The fast algorithm of the last section is an array version of fast RLS algorithms known as FTF (Fast Transversal Filter) and FAEST (Fast A posteriori Error Sequential Technique). In contrast to the above array description, where the transformation Θ_i that updates the data from time i to time $(i+1)$ is left implicit, the FTF and FAEST algorithms involve explicit sets of equations.

The derivation of these explicit sets of equations can be motivated as follows. Note that the factorization (21.54) is highly nonunique. What is special about (21.54) [and also (21.55)] is that we have forced $\mathbf{S}_0$ to be a signature matrix, i.e., a matrix with $\pm 1's$ on its diagonal. More generally, we can allow for different factorizations with an $\mathbf{S}_0$ that is not restricted to be a signature matrix. Different choices lead to different sets of equations.

TABLE 21.5 A Fast Array Algorithm

Input. Prewindowed data $\{d(j), u(j)\}$ for $j \geq 1$ and Π_0 as in (21.53) in the cost (21.27).

Initialization. Set

$$\mathbf{w}_{-1} = \bar{\mathbf{w}}, \quad \gamma^{-1/2}(0) = 1$$

$$\bar{\text{Ł}}_0 = \sqrt{\delta} \cdot \begin{bmatrix} 1 & 0 \\ \mathbf{0} & \mathbf{0} \\ 0 & \lambda^{\frac{M}{2}} \end{bmatrix}, \quad \mathbf{S}_0 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

Repeat for each time instant $i \geq 0$:

$$\begin{bmatrix} \gamma^{-1/2}(i) & \begin{bmatrix} u(i+1) & \mathbf{u}_i^T \end{bmatrix} \bar{\text{Ł}}_i \\ \begin{bmatrix} 0 \\ \mathbf{g}_i \gamma^{-1/2}(i) \end{bmatrix} & \bar{\text{Ł}}_i \end{bmatrix} \Theta_i = \begin{bmatrix} \gamma^{-1/2}(i+1) & \mathbf{0}^T \\ \begin{bmatrix} \mathbf{g}_{i+1} \gamma^{-1/2}(i+1) \\ 0 \end{bmatrix} & \sqrt{\lambda}\, \bar{\text{Ł}}_{i+1} \end{bmatrix}$$

where Θ_i is any $(1 \oplus \mathbf{S}_0)-$orthogonal matrix that produces the zero pattern in the postarray, and $\bar{\text{Ł}}_i$ is a two-column matrix.

The weight-vector estimate is updated via:

$$\mathbf{w}_i = \mathbf{w}_{i-1} + \left[\frac{\mathbf{g}_i}{\gamma^{1/2}(i)} \right] \left[\gamma^{-1/2}(i) \right]^{-1} \left[d(i) - \mathbf{u}_i^T \mathbf{w}_{i-1} \right]$$

The computational cost is $O(M)$ per iteration.

More explicitly, assume we factor the difference matrix in (21.55) as

$$\begin{bmatrix} \mathbf{P}_i & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix} - \begin{bmatrix} 0 & \mathbf{0}^T \\ \mathbf{0} & \mathbf{P}_{i-1} \end{bmatrix} = \lambda \cdot \text{Ł}_i \mathbf{M}_i \text{Ł}_i^T , \tag{21.59}$$

where Ł_i is an $(M+1) \times 2$ matrix and $\mathbf{M}_i$ is a 2×2 matrix that is not restricted to be a signature matrix. [We already know from the earlier array-based argument that this difference is always low-rank.]

Given the factorization (21.59), it is easy to verify that two successive gain vectors satisfy the relation:

$$\begin{bmatrix} \mathbf{g}_{i+1} \gamma^{-1}(i+1) \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{g}_i \gamma^{-1}(i) \end{bmatrix} + \text{Ł}_i \mathbf{M}_i \text{Ł}_i^T \begin{bmatrix} u(i+1) \\ \mathbf{u}_i \end{bmatrix} .$$

This is identical to (21.58) except that $\mathbf{S}_i$ is replaced by $\mathbf{M}_i$ and $\bar{\text{Ł}}_i$ is replaced by Ł_i. The fast array algorithm of the previous section provides one possibility for enforcing this relation and, hence, of updating $\mathbf{g}_i$ to $\mathbf{g}_{i+1}$ via updates of $\bar{\text{Ł}}_i$.

The FTF and FAEST algorithms follow by employing one such alternative factorization, where the two columns of the factor Ł_i turn out to be related to the solution of two fundamental problems in adaptive filter theory: the so-called forward and backward prediction problems. Moreover, the $\mathbf{M}_i$ factor turns out to be diagonal with entries equal to the so-called forward and backward minimum prediction energies. An explicit derivation of the FTF equations can be pursued along these lines. We omit the details and continue to focus on the square-root formulation. We now proceed to discuss order-recursive adaptive filters within this framework.

21.8 Order-Recursive Filters

The RLS algorithms that were derived in the previous sections are all fixed-order solutions of (21.27) in the sense that they recursively evaluate successive weight estimates $\mathbf{w}_i$ that correspond to a fixed-order combiner of order M. This form of computing the minimizing solution $\mathbf{w}_N$ is not convenient from an order-recursive point of view. In other words, assume we pose a new optimization problem of the same form as (21.27) but where the vectors $\{\mathbf{w}, \mathbf{u}_j\}$ are now of order $(M+1)$ rather than M. How do the weight estimates of this new higher-dimensional problem relate to the weight estimates of the lower dimensional problem?

Before addressing this issue any further, it is apparent at this stage that we need to introduce a notational modification in order to keep track of the proper sizes of the variables. Indeed, from now on, we shall explicitly indicate the size of a variable by employing an additional subscript. For example, we shall write $\{\mathbf{w}_M, \mathbf{u}_{M,j}\}$ instead of $\{\mathbf{w}, \mathbf{u}_j\}$ to denote vectors of size M.

Returning to the point raised in the previous paragraph, let $\mathbf{w}_{M+1,N}$ denote the optimal solution of the new optimization problem (with $(M+1)-$dimensional vectors $\{\mathbf{w}_{M+1}, \mathbf{u}_{M+1,j}\}$. The adaptive algorithms of the previous sections give an explicit recursive (time-update) relation between $\mathbf{w}_{M,N}$ and $\mathbf{w}_{M,N-1}$. But they do not provide a recursive (order-update) relation between $\mathbf{w}_{M,N}$ and $\mathbf{w}_{M+1,N}$.

There is an alternative to the FIR implementation of Fig. 21.3 that allows us to easily carry over the information from previous computations for the order M filter. This is the so-called lattice filter.

From now on we assume, for simplicity of presentation, that the weighting matrix Π_0 in (21.27) is very large, i.e., $\Pi_0 \to \infty\mathbf{I}$. This assumption reduces (21.27) to a standard least-squares formulation:

$$\min_{\mathbf{w}_M} \left[\sum_{j=0}^{N} \lambda^{N-i} |d(j) - \mathbf{u}_{M,j}^T \mathbf{w}_M|^2 \right] . \tag{21.60}$$

The order-recursive filters of this section deal with this kind of minimization.

Now suppose that our interest in solving (21.60) is not to explicitly determine the weight estimate $\mathbf{w}_{M,N}$, but rather to determine estimates for the reference signals $\{d(\cdot)\}$, say

$$d_M(N) = \mathbf{u}_{M,N}^T \mathbf{w}_{M,N} = \text{ estimate of } \ d(N) \ \text{ of order } \ M .$$

Likewise, for the higher-order problem,

$$d_{M+1}(N) = \mathbf{u}_{M+1,N}^T \mathbf{w}_{M+1,N} = \text{ estimate of } \ d(N) \ \text{ of order } \ M+1 .$$

The resulting estimation errors will be denoted by

$$e_M(N) = d(N) - d_M(N), \quad e_{M+1}(N) = d(N) - d_{M+1}(N) .$$

The lattice solution allows us to update $e_M(N)$ to $e_{M+1}(N)$ without explicitly computing the weight estimates $\mathbf{w}_{M,N}$ and $\mathbf{w}_{M+1,N}$.

The discussion that follows relies heavily on the orthogonality property of least-squares solutions and, therefore, serves as a good illustration of the power and significance of this property. It will further motivate the introduction of the forward and backward prediction problems.

21.8.1 Joint Process Estimation

For the sake of illustration, and without loss of generality, the discussion in this section assumes particular values for M and λ, say $M = 3$ and $\lambda = 1$. These assumptions simplify the exposition without affecting the general conclusions. In particular, a nonunity λ can always be incorporated into the discussion by properly normalizing the vectors involved in the derivation [cf. (21.28) and (21.29)] and we will do so later. We continue to assume prewindowed data (i.e., the data is zero for time instants $i \leq 0$).

To begin with, assume we solve the following problem [as suggested by (21.60)]: minimize over

$\mathbf{w}_3$ the cost function

$$\left\| \underbrace{\begin{bmatrix} 0 \\ d(1) \\ d(2) \\ \vdots \\ d(N) \end{bmatrix}}_{\mathbf{d}_N} - \underbrace{\begin{bmatrix} 0 & 0 & 0 \\ u(1) & 0 & 0 \\ u(2) & u(1) & 0 \\ \vdots & \vdots & \vdots \\ u(N) & u(N-1) & u(N-2) \end{bmatrix}}_{\mathbf{A}_{3,N}} \underbrace{\begin{bmatrix} w_3(1) \\ w_3(2) \\ w_3(3) \end{bmatrix}}_{\mathbf{w}_3} \right\|^2 \tag{21.61}$$

where $\mathbf{d}_N$ denotes the vector of desired signals up to time N, and $\mathbf{A}_{3,N}$ denotes a three-column matrix of input data $\{u(\cdot)\}$, also up to time N.

The optimal solution is denoted by $\mathbf{w}_{3,N}$. The subscript $_N$ indicates that it is an estimate based on the data $u(\cdot)$ up to time N. Determining $\mathbf{w}_{3,N}$ corresponds to determining the entries of a 3-dimensional weight vector so as to approximate the column vector $\mathbf{d}_N$ by the linear combination $\mathbf{A}_{3,N}\mathbf{w}_{3,N}$ in the least-squares sense (21.61). We thus say that expression (21.61) defines a third-order estimator for the reference sequence $\{d(\cdot)\}$. The resulting *a posteriori* estimation error vector is denoted by

$$\mathbf{e}_{3,N} = \mathbf{d}_N - \mathbf{A}_{3,N}\mathbf{w}_{3,N} \ ,$$

where, for example, the last entry of $\mathbf{e}_{3,N}$ is given by

$$e_3(N) = d(N) - \mathbf{u}_{3,N}^T\mathbf{w}_{3,N} \ ,$$

and it denotes the a *posteriori* estimation error in estimating $d(N)$ from a linear combination of the three most recent inputs.

We already know from the orthogonality property of least-squares solutions that the a *posteriori* residual vector $\mathbf{e}_{3,N}$ has to be orthogonal to the data matrix $\mathbf{A}_{3,N}$, viz.,

$$\mathbf{A}_{3,N}^T\mathbf{e}_{3,N} = \mathbf{0}.$$

We also know that the optimal solution $\mathbf{w}_{3,N}$ provides an estimate vector $\mathbf{A}_{3,N}\mathbf{w}_{3,N}$ that is the closest element in the column space of $\mathbf{A}_{3,N}$ to the column vector $\mathbf{d}_N$.

Now assume that we wish to solve the next higher order problem, viz., of order $M = 4$: minimize over $\mathbf{w}_4$ the cost function

$$\left\|\mathbf{d}_N - \mathbf{A}_{4,N}\mathbf{w}_4\right\|^2 \ , \tag{21.62}$$

where

$$\mathbf{A}_{4,N} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ u(1) & 0 & 0 & 0 \\ u(2) & u(1) & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ u(N-1) & u(N-2) & u(N-3) & u(N-4) \\ u(N) & u(N-1) & u(N-2) & u(N-3) \end{bmatrix}, \mathbf{w}_4 = \begin{bmatrix} w_4(1) \\ w_4(2) \\ w_4(3) \\ w_4(4) \end{bmatrix}.$$

This statement is very close to (21.61) except for an extra column in the data matrix $\mathbf{A}_{4,N}$: the first three columns of $\mathbf{A}_{4,N}$ coincide with those of $\mathbf{A}_{3,N}$, while the last column of $\mathbf{A}_{4,N}$ contains the extra new data that are needed for a fourth-order estimator. More specifically, $\mathbf{A}_{3,N}$ and $\mathbf{A}_{4,N}$ are related as follows:

$$\mathbf{A}_{4,N} = \left[\begin{array}{cc} & 0 \\ & 0 \\ & 0 \\ \mathbf{A}_{3,N} & \vdots \\ & u(N-4) \\ & u(N-3) \end{array} \right] . \tag{21.63}$$

The problem in (21.62) requires us to linearly combine the four columns of $\mathbf{A}_{4,N}$ in order to compute the fourth-order estimates of $\{0, d(1), d(2), \ldots, d(N)\}$. In other words, it requires us to determine the closest element in the column space of $\mathbf{A}_{4,N}$ to the same column vector $\mathbf{d}_N$.

We already know what is the closest element to $\mathbf{d}_N$ in the column space of $\mathbf{A}_{3,N}$, which is a submatrix of $\mathbf{A}_{4,N}$. This suggests that we should try to decompose the column space of $\mathbf{A}_{4,N}$ into two orthogonal subspaces, viz.,

$$\text{Range}(\mathbf{A}_{4,N}) = \text{Range}(\mathbf{A}_{3,N}) \oplus \text{Range}(\mathbf{m}) , \tag{21.64}$$

where $\mathbf{m}$ is a column vector that is orthogonal to $\mathbf{A}_{3,N}$, $\mathbf{A}_{3,N}^T \mathbf{m} = \mathbf{0}$. The notation $\text{Range}(\mathbf{A}_{3,N}) \oplus \text{Range}(\mathbf{m})$ also means that every element in the column space of $\mathbf{A}_{4,N}$ can be expressed as a linear combination of the columns of $\mathbf{A}_{3,N}$ and of $\mathbf{m}$.

The desired decomposition motivates the backward prediction problem.

21.8.2 The Backward Prediction Error Vectors

We continue to assume $\lambda = 1$ and $M = 3$, and we note that the required decomposition can be accomplished by projecting the last column of $\mathbf{A}_{4,N}$ onto the column space of its first three columns (i.e., onto the column space of $\mathbf{A}_{3,N}$) and keeping the residual vector as the desired vector $\mathbf{m}$. This is nothing but a Gram-Schmidt orthogonalization step and it is equivalent to the following minimization problem: minimize over $\mathbf{w}_3^b$

$$\left\| \underbrace{\begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ u(N-4) \\ u(N-3) \end{bmatrix}}_{\substack{\text{Last column} \\ \text{of } \mathbf{A}_{4,N}}} - \mathbf{A}_{3,N} \underbrace{\begin{bmatrix} w_3^b(1) \\ w_3^b(2) \\ w_3^b(3) \end{bmatrix}}_{\mathbf{w}_3^b} \right\|^2 . \tag{21.65}$$

This is also a special case of (21.60) where we have replaced the sequence

$$\{0, d(1), \ldots, d(N)\}$$

by the sequence

$$\{0, 0, 0, \ldots, u(N-4), u(N-3)\}.$$

We denote the optimal solution by $\mathbf{w}_{3,N}^b$. The subscript $_N$ indicates that it is an estimate based on the data $u(\cdot)$ up to time N. Determining $\mathbf{w}_{3,N}^b$ corresponds to determining the entries of a 3-dimensional weight vector so as to approximate the last column of $\mathbf{A}_{4,N}$ by a linear combination of the columns of $\mathbf{A}_{3,N}$, viz., $\mathbf{A}_{3,N}\mathbf{w}^b_{3,N}$, in the least-squares sense.

Note that the entries in every row of the data matrix $\mathbf{A}_{3,N}$ are the three "future" values corresponding to the entry in the last column of $\mathbf{A}_{4,N}$. Hence, the last element of the above linear combination serves as a *backward* prediction of $u(N-3)$ in terms of $\{u(N), u(N-1), u(N-2)\}$. A similar remark holds for the other entries. The superscript ^{b} stands for *backward*.

We thus say that expression (21.65) defines a third-order backward prediction problem. The resulting *a posteriori* backward prediction error vector is denoted by

$$\mathbf{b}_{3,N} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ u(N-4) \\ u(N-3) \end{bmatrix} - \mathbf{A}_{3,N}\mathbf{w}^b_{3,N} \,.$$

In particular, the *last* entry of $\mathbf{b}_{3,N}$ is defined as the a *posteriori* backward prediction error in estimating $u(N-3)$ from a linear combination of the future 3 inputs. It is denoted by $b_3(N)$ and is given by

$$b_3(N) = u(N-3) - \mathbf{u}^T_{3,N}\mathbf{w}^b_{3,N} \,. \tag{21.66}$$

We further know, from the orthogonality property of least-squares solutions, that the *a posteriori* backward residual vector $\mathbf{b}_{3,N}$ has to be orthogonal to the data matrix $\mathbf{A}_{3,N}$, $\mathbf{A}^T_{3,N}\mathbf{b}_{3,N} = \mathbf{0}$, which therefore implies that it can be taken as the $\mathbf{m}$ column that we mentioned earlier, viz., we can write

$$\text{Range }(\mathbf{A}_{4,N}) = \text{Range }(\mathbf{A}_{3,N}) \oplus \text{Range }(\mathbf{b}_{3,N}). \tag{21.67}$$

Our original motivation for introducing the *a posteriori* backward residual vector $\mathbf{b}_{3,N}$ was the desire to solve the fourth-order problem (21.62), not afresh, but in a way so as to exploit the solution of lower order, thus leading to an order-recursive algorithm.

Assume now that we have available the estimation error vectors $\mathbf{e}_{3,N}$ and $\mathbf{b}_{3,N}$, which are both orthogonal to $\mathbf{A}_{3,N}$. Knowing that $\mathbf{b}_{3,N}$ leads to an orthogonal decomposition of the column space of $\mathbf{A}_{4,N}$ as in (21.67), then updating $\mathbf{e}_{3,N}$ into a fourth-order *a posteriori* residual vector $\mathbf{e}_{4,N}$, which has to be orthogonal to $\mathbf{A}_{4,N}$, simply corresponds to projecting the vector $\mathbf{e}_{3,N}$ onto the vector $\mathbf{b}_{3,N}$. More explicitly, it corresponds to determining a scalar coefficient k_3 that solves the optimization problem

$$\min_{k_3} \left\| \mathbf{e}_{3,N} - k_3\mathbf{b}_{3,N} \right\|^2 \,. \tag{21.68}$$

This is a standard least-squares problem and its optimal solution is denoted by

$$k_3(N) = \frac{1}{\mathbf{b}^T_{3,N}\mathbf{b}_{3,N}}\, \mathbf{b}^T_{3,N}\mathbf{e}_{3,N} \,. \tag{21.69}$$

We now know how to update $\mathbf{e}_{3,N}$ into $\mathbf{e}_{4,N}$ by projecting $\mathbf{e}_{3,N}$ onto $\mathbf{b}_{3,N}$. In order to be able to proceed with this order update procedure, we still need to know how to order-update the backward residual vector. That is, we need to know how to go from $\mathbf{b}_{3,N}$ to $\mathbf{b}_{4,N}$.

21.8.3 The Forward Prediction Error Vectors

We continue to assume $\lambda = 1$ and $M = 3$. The order-update of the backward residual vector motivates us to introduce the forward prediction problem: minimize over $\mathbf{w}_3^f$ the cost function

$$\left\| \begin{bmatrix} u(1) \\ u(2) \\ u(3) \\ \vdots \\ u(N+1) \end{bmatrix} - \mathbf{A}_{3,N} \underbrace{\begin{bmatrix} w_3^f(1) \\ w_3^f(2) \\ w_3^f(3) \end{bmatrix}}_{\mathbf{w}_3^f} \right\|^2 . \tag{21.70}$$

We denote the optimal solution by $\mathbf{w}_{3,N+1}^f$. The subscript indicates that it is an estimate based on the data $u(\cdot)$ up to time $N+1$. Determining $\mathbf{w}_{3,N+1}^f$ corresponds to determining the entries of a 3-dimensional weight vector so as to approximate the column vector

$$\begin{bmatrix} u(1) \\ u(2) \\ u(3) \\ \vdots \\ u(N+1) \end{bmatrix}$$

by a linear combination of the columns of $\mathbf{A}_{3,N}$, viz., $\mathbf{A}_{3,N}\mathbf{w}_{3,N+1}^f$.

Note that the entries of the successive rows of the data matrix $\mathbf{A}_{3,N}$ are the past three inputs relative to the corresponding entries of the column vector. Hence, the last element of the linear combination $\mathbf{A}_{3,N}\mathbf{w}_{3,N+1}^f$ serves as a *forward* prediction of $u(N+1)$ in terms of $\{u(N), u(N-1), u(N-2)\}$. A similar remark holds for the other entries. The superscript ^{f} stands for *forward*.

We thus say that expression (21.70) defines a third-order forward prediction problem. The resulting *a posteriori* forward prediction error vector is denoted by

$$\mathbf{f}_{3,N+1} = \begin{bmatrix} u(1) \\ u(2) \\ u(3) \\ \vdots \\ u(N+1) \end{bmatrix} - \mathbf{A}_{3,N}\mathbf{w}_{3,N+1}^f .$$

In particular, the *last* entry of $\mathbf{f}_{3,N+1}$ is defined as the *a posteriori* forward prediction error in estimating $u(N+1)$ from a linear combination of the past three inputs. It is denoted by $f_3(N+1)$ and is given by

$$f_3(N+1) = u(N+1) - \mathbf{u}_{3,N}\mathbf{w}_{3,N+1}^f . \tag{21.71}$$

Now assume that we wish to solve the next-higher order problem, viz., of order $M = 4$: minimize over $\mathbf{w}_4^f$ the cost function

$$\left\| \begin{bmatrix} u(1) \\ u(2) \\ u(3) \\ \vdots \\ u(N+1) \end{bmatrix} - \mathbf{A}_{4,N} \begin{bmatrix} w_4^f(1) \\ w_4^f(2) \\ w_4^f(3) \\ w_4^f(4) \end{bmatrix} \right\|^2 . \tag{21.72}$$

We again observe that this statement is very close to (21.70) except for an extra column in the data matrix $\mathbf{A}_{4,N}$, in precisely the same way as happened with $\mathbf{e}_{4,N}$ and $\mathbf{b}_{3,N}$. We can therefore obtain $\mathbf{f}_{4,N+1}$ by projecting $\mathbf{f}_{3,N+1}$ onto $\mathbf{b}_{3,N}$ and taking the residual vector as $\mathbf{f}_{4,N+1}$,

$$\min_{k_3^f} \|\mathbf{f}_{3,N+1} - k_3^f \mathbf{b}_{3,N}\|^2 \ . \tag{21.73}$$

This is also a standard least-squares problem and we denote its optimal solution by $k_3^f(N+1)$,

$$k_3^f(N+1) = \frac{\mathbf{b}_{3,N}^T \mathbf{f}_{3,N+1}}{\mathbf{b}_{3,N}^T \mathbf{b}_{3,N}} \ , \tag{21.74}$$

with

$$\mathbf{f}_{4,N+1} = \mathbf{f}_{3,N+1} - k_3^f(N+1)\mathbf{b}_{3,N} \ . \tag{21.75}$$

Similarly, the backward residual vector $\mathbf{b}_{3,N}$ can be updated to $\mathbf{b}_{4,N+1}$ by projecting $\mathbf{b}_{3,N}$ onto $\mathbf{f}_{3,N+1}$,

$$\min_{k_3^b} \left\| \mathbf{b}_{3,N} - k_3^b \mathbf{f}_{3,N+1} \right\|^2 \ , \tag{21.76}$$

and we get, after denoting the optimal solution by $k_3^b(N+1)$,

$$\mathbf{b}_{4,N+1} = \mathbf{b}_{3,N} - k_3^b(N+1)\mathbf{f}_{3,N+1} \ , \tag{21.77}$$

where

$$k_3^b(N+1) = \frac{\mathbf{f}_{3,N+1}^T \mathbf{b}_{3,N}}{\mathbf{f}_{3,N+1}^T \mathbf{f}_{3,N+1}} \ . \tag{21.78}$$

Note the change in the time index as we move from $\mathbf{b}_{3,N}$ to $\mathbf{b}_{4,N+1}$. This is because $\mathbf{b}_{4,N+1}$ is obtained by projecting $\mathbf{b}_{3,N}$ onto $\mathbf{f}_{3,N+1}$, which corresponds to the following definition for $\mathbf{b}_{4,N+1}$,

$$\mathbf{b}_{4,N+1} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ u(N-4) \\ u(N-3) \end{bmatrix} - \mathbf{A}_{4,N+1} \underbrace{\begin{bmatrix} w_{4,N+1}^b(1) \\ w_{4,N+1}^b(2) \\ w_{4,N+1}^b(3) \\ w_{4,N+1}^b(4) \end{bmatrix}}_{\mathbf{w}_{4,N+1}^b} \ .$$

Finally, in view of (21.69), the joint process estimation problem involves a recursion of the form

$$\mathbf{e}_{4,N} = \mathbf{e}_{3,N} - k_3(N)\mathbf{b}_{3,N} \ , \tag{21.79}$$

where

$$k_3(N) = \frac{\mathbf{b}_{3,N}^T \mathbf{e}_{3,N}}{\mathbf{b}_{3,N}^T \mathbf{b}_{3,N}} \ . \tag{21.80}$$

21.8.4 A Nonunity Forgetting Factor

For a general filter order M and for a nonunity λ, an extension of the above arguments would show that the prediction vectors can be updated as follows:

$$\begin{aligned}
\mathbf{f}_{M+1,N+1} &= \mathbf{f}_{M,N+1} - k_M^f(N+1)\mathbf{b}_{M,N}\ , \\
\mathbf{b}_{M+1,N+1} &= \mathbf{b}_{M,N} - k_M^b(N+1)\mathbf{f}_{M,N+1}\ , \\
\mathbf{e}_{M+1,N} &= \mathbf{e}_{M,N} - k_M(N)\mathbf{b}_{M,N}\ , \\
k_M^f(N+1) &= \frac{\mathbf{b}_{M,N}^T\Lambda_N\mathbf{f}_{M,N+1}}{\mathbf{b}_{M,N}^T\Lambda_N\mathbf{b}_{M,N}}\ , \\
k_M^b(N+1) &= \frac{\mathbf{f}_{M,N+1}^T\Lambda_N\mathbf{b}_{M,N}}{\mathbf{f}_{M,N+1}^T\Lambda_N\mathbf{f}_{M,N+1}}\ , \\
k_M(N) &= \frac{\mathbf{b}_{M,N}^T\Lambda_N\mathbf{e}_{M,N}}{\mathbf{b}_{M,N}^T\Lambda_N\mathbf{b}_{M,N}}\ ,
\end{aligned}$$

where

$$\Lambda_N = \text{diag}\,\{\lambda^N, \lambda^{N-1}, \ldots, \lambda, 1\}\ .$$

For completeness, we also include the defining relations for the *a priori* and *a posteriori* prediction errors:

$$\begin{aligned}
\beta_M(N) &= u(N-M) - \mathbf{u}_{M,N}^T\mathbf{w}_{M,N-1}^b\ , \\
b_M(N) &= u(N-M) - \mathbf{u}_{M,N}^T\mathbf{w}_{M,N}^b\ , \\
\alpha_M(N+1) &= u(N+1) - \mathbf{u}_{M,N}^T\mathbf{w}_{M,N}^f\ , \\
f_M(N+1) &= u(N+1) - \mathbf{u}_{M,N}^T\mathbf{w}_{M,N+1}^f\ .
\end{aligned}$$

Using the definition (21.40) for a conversion factor in a least-squares formulation, it is easy to see that the same factor converts the *a priori* prediction errors to the corresponding *a posteriori* prediction errors. This factor will be denoted by $\gamma_M(N)$.

TABLE 21.6 Useful Relations for the Prediction Problems

Variable	Definition or Relation
A priori forward error	$\alpha_M(N+1) = u(N+1) - \mathbf{u}_{M,N}^T\mathbf{w}_{M,N-1}^f$
A priori backward error	$\beta_M(N) = u(N-M) - \mathbf{u}_{M,N}^T\mathbf{w}_{M,N-1}^b$
A posteriori forward error	$f_M(N+1) = u(N+1) - \mathbf{u}_{M,N}^T\mathbf{w}_{M,N}^f$
A posteriori backward error	$b_M(N) = u(N-M) - \mathbf{u}_{M,N}^T\mathbf{w}_{M,N}^b$
Forward error by conversion	$f_M(N+1) = \alpha_M(N+1)\gamma_M(N)$
Backward error by conversion	$b_M(N) = \beta_M(N)\gamma_M(N)$
Gain vector	$g_{M,N} = \Phi_{M,N}^{-1}\mathbf{u}_{M,N}$
Conversion factor	$\gamma_M(N) = 1 - \mathbf{u}_{M,N}^T\Phi_{M,N}^{-1}\mathbf{u}_{M,N}$
Minimum forward-prediction error energy	$\xi_M^f(N+1) = \lambda\xi_M^f(N) + \lvert\bar{f}_M(N+1)\rvert^2$
Minimum backward-prediction error energy	$\xi_M^b(N+1) = \lambda\xi_M^b(N) + \lvert\bar{b}_M(N+1)\rvert^2$

Table 21.6 summarizes, for ease of reference, the definitions and relations that have been introduced thus far. In particular, the last two lines of the table also provide time-update relations for the minimum costs of the forward and backward prediction problems. These costs are denoted by $\xi_M^f(N+1)$ and $\xi_M^b(N)$ and they are equal to the quantities $\mathbf{f}_{M,N+1}^T\Lambda_N\mathbf{f}_{M,N+1}$ and $\mathbf{b}_{M,N}^T\Lambda_N\mathbf{b}_{M,N}$ that

appear in the denominators of some of the earlier expressions. The last two relations of Table 21.6 use the result (21.43) to express the minimum costs in terms of the so-called *angle-normalized* prediction errors:

$$\begin{aligned} \bar{f}_M(N+1) &= \alpha_M(N+1)\gamma_M^{1/2}(N)\,, && (21.81)\\ \bar{b}_M(N) &= \beta_M(N)\gamma_M^{1/2}(N)\,. && (21.82) \end{aligned}$$

We can derive, in different ways, similar update relations for the inner product terms

$$\begin{aligned} \Delta_M(N+1) &= \mathbf{f}_{M,N+1}^T \Lambda_N \mathbf{b}_{M,N}\,,\\ \rho_M(N) &= \mathbf{b}_{M,N}^T \Lambda_N \mathbf{e}_{M,N}\,. \end{aligned}$$

One possibility is to note, after some algebra and using the orthogonality principle, that the following relation holds:

$$\Delta_M(N+1) = \begin{bmatrix} 1 & -(\mathbf{w}_{M,N}^f)^T & 0 \end{bmatrix} \Phi_{M+2,N+1} \begin{bmatrix} 0 \\ -\mathbf{w}_{M,N}^b \\ 1 \end{bmatrix},$$

where

$$\Phi_{M+2,N+1} = \sum_{j=0}^{N+1} \lambda^{N+1-j} \mathbf{u}_{M+2,j} \mathbf{u}_{M+2,j}^T$$

If we now invoke the time-update expression

$$\Phi_{M+2,N+1} = \lambda \Phi_{M+2,N} + \mathbf{u}_{M+2,N+1}^T \mathbf{u}_{M+2,N+1},$$

we conclude that $\Delta_M(N+1)$ satisfies the time-update formula:

$$\begin{aligned} \Delta_M(N+1) &= \lambda \Delta_M(N) + \alpha_M(N+1) b_M(N)\\ &= \lambda \Delta_M(N) + \frac{f_M(N+1) b_M(N)}{\gamma_M(N)}\,. \end{aligned}$$

A similar argument for $\rho_M(N)$ shows that it satisfies the time-update relation

$$\rho_M(N) = \lambda \rho_M(N-1) + \frac{e_M(N) b_M(N)}{\gamma_M(N)}\,.$$

Finally, the orthogonality principle can again be invoked to derive order-update (rather than time-update) relations for $\xi_M^f(N+1)$ and $\xi_M^b(N)$. Indeed, using $\mathbf{f}_{M+1,N+1}^T \Lambda_N \mathbf{b}_{M,N} = 0$ we obtain

$$\begin{aligned} \xi_{M+1}^f(N+1) &= \mathbf{f}_{M+1,N+1}^T \Lambda_N \mathbf{f}_{M+1,N+1} = \mathbf{f}_{M+1,N+1}^T \Lambda_N \mathbf{f}_{M,N+1}\,,\\ &= \xi_M^f(N+1) - \frac{\|\Delta_M(N+1)\|^2}{\xi_M^b(N)}\,. \end{aligned}$$

Likewise,

$$\xi_{M+1}^b(N+1) = \xi_M^b(N) - \frac{\|\Delta_M(N+1)\|^2}{\xi_M^f(N+1)}\,.$$

Table 21.7 summarizes the order-update relations derived thus far.

TABLE 21.7 Order-Update Relations

$$\Delta_M(N+1) = \lambda\Delta_M(N) + \frac{f_M(N+1)b_M(N)}{\gamma_M(N)}$$

$$\rho_M(N) = \lambda\rho_M(N-1) + \frac{e_M(N)b_M(N)}{\gamma_M(N)}$$

$$\begin{aligned}
&\xi^f_M(N+1) = \lambda\xi^f_M(N) + \frac{|f_M(N+1)|^2}{\gamma_M(N)}\\
&\xi^b_M(N) = \lambda\xi^b_M(N-1) + \frac{|b_M(N)|^2}{\gamma_M(N)}
\end{aligned}$$

$$\begin{aligned}
&k^f_M(N+1) = \Delta_M(N+1)/\xi^b_M(N)\\
&k^b_M(N+1) = \Delta_M(N+1)/\xi^f_M(N+1)\\
&k_M(N) = \rho_M(N)/\xi^b_M(N)
\end{aligned}$$

$$\begin{aligned}
&f_{M+1}(N+1) = f_M(N+1) - k^f_M(N+1)b_M(N)\\
&b_{M+1}(N+1) = b_M(N) - k^b_M(N+1)f_M(N+1)\\
&e_{M+1}(N) = e_M(N) - k_M(N)b_M(N)
\end{aligned}$$

$$\begin{aligned}
&\xi^f_{M+1}(N+1) = \xi^f_M(N+1) - \frac{|\Delta_M(N+1)|^2}{\xi^b_M(N)}\\
&\xi^b_{M+1}(N+1) = \xi^b_M(N) \ - \ \frac{|\Delta_M(N+1)|^2}{\xi^f_M(N+1)}
\end{aligned}$$

21.8.5 The QRD Least-Squares Lattice Filter

There are many variants of adaptive lattice algorithms. In this section we present one such variant in square-root form. Most, if not all, other alternatives can be obtained as special cases. Some alternatives propagate the *a posteriori* prediction errors $\{f_M(N+1), b_M(N)\}$, while others employ the *a priori* prediction errors $\{\alpha_M(N+1), \beta_M(N)\}$. The QRD-LSL algorithm we present here is invariant to the particular choice of *a posteriori* or *a priori* errors because it propagates the *angle normalized* prediction errors that we introduced earlier in (21.81) and (21.82), viz.,

$$\begin{aligned}
\bar{f}_M(i+1) &= \alpha_M(i+1)\gamma_M^{1/2}(i) = [u(i+1) - \mathbf{u}^T_{M,i}\mathbf{w}^f_{M,i}]\gamma_M^{1/2}(i) \, ,\\
\bar{b}_M(i) &= \beta_M(i)\gamma_M^{1/2}(i) = [u(i-M) - \mathbf{u}^T_{M,i}\mathbf{w}^b_{M,i-1}]\gamma_M^{1/2}(i) \, .
\end{aligned}$$

The QRD-LSL algorithm can be motivated as follows. Assume we form the following two vectors of angle normalized prediction errors:

$$\bar{\mathbf{f}}_{M,N+1} = \begin{bmatrix} \bar{f}_M(1) \\ \bar{f}_M(2) \\ \vdots \\ \bar{f}_M(N+1) \end{bmatrix} , \quad \bar{\mathbf{b}}_{M,N} = \begin{bmatrix} \bar{b}_M(0) \\ \bar{b}_M(1) \\ \vdots \\ \bar{b}_M(N) \end{bmatrix} . \tag{21.83}$$

We then conclude from the time-updates in Table 21.6 for $\xi^f_M(N+1)$ and $\xi^b_M(N)$ that $\xi^f_M(N+1)$ and $\xi^b_M(N)$ are the (weighted) squared Euclidean norms of the angle normalized vectors $\bar{\mathbf{f}}_M(N+1)$ and $\bar{\mathbf{b}}_M(N)$, respectively. That is, $\xi^f_M(N+1) = \bar{\mathbf{f}}^T_{M,N+1}\Lambda_N\bar{\mathbf{f}}_{M,N+1}$ and $\xi^b_M(N) = \bar{\mathbf{b}}^T_{M,N}\Lambda_N\bar{\mathbf{b}}_{M,N}$. Likewise, it follows from the time-update for $\Delta_M(N+1)$ that it is equal to the inner product of the angle normalized vectors,

$$\Delta_M(N+1) = \bar{\mathbf{b}}^T_{M,N}\Lambda_N\bar{\mathbf{f}}_{M,N+1} \, . \tag{21.84}$$

Consequently, the coefficients $k^f_M(N+1)$ and $k^b_M(N+1)$ are also equal to the ratios of the inner product of the angle normalized vectors to their energies. But recall that $k^f_M(N+1)$ is the coefficient we need in order to project $\mathbf{f}_{M,N+1}$ onto $\mathbf{b}_{M,N}$. This means that we can alternatively evaluate the same coefficient by posing the problem of projecting $\bar{\mathbf{f}}_{M,N+1}$ onto $\bar{\mathbf{b}}_{M,N}$. In a similar fashion, $k^b_M(N+1)$

can be evaluated alternatively by projecting $\bar{\mathbf{b}}_{M,N}$ onto $\bar{\mathbf{f}}_{M,N+1}$. (The inner products and projections are to be understood here to include the additional weighting by Λ_N.)

We are therefore reduced to two simple projection problems that involve projecting a vector onto another vector (with exponential weighting). But these are special cases of standard least-squares problems. In particular, recall that the QR solution of Table 21.4 solves the problem of projecting a given vector $\mathbf{d}_N$ onto the range space of a data matrix $\mathbf{A}_N$ (whose rows are $\mathbf{u}_j^T$).

In a similar fashion, we can write down the QR solution that would solve the problem of projecting $\bar{\mathbf{f}}_{M,N+1}$ onto $\bar{\mathbf{b}}_{M,N}$. For this purpose, we introduce the scalar variables $q_M^f(N+1)$ and $q_M^b(N+1)$ [recall the earlier notation (21.50)]:

$$q_M^b(N+1) = \frac{\Delta_M(N+1)}{\xi_M^{b/2}(N)}\ , \quad q_M^f(N+1) = \frac{\Delta_M(N+1)}{\xi_M^{f/2}(N+1)}\ . \tag{21.85}$$

The QR array that updates the forward prediction errors can now be obtained as follows. Form the 3 × 2 prearray (this is a special case of the QR array of Table 21.4):

$$\mathbf{A} = \left[\begin{array}{cc} \sqrt{\lambda}\xi_M^{b/2}(N-1) & \bar{b}_M(N) \\ \sqrt{\lambda}q_M^b(N) & \bar{f}_M(N+1) \\ 0 & 1 \end{array}\right]$$

and choose an orthogonal rotation $\Theta_{M,N}^b$ that reduces it to the form

$$\mathbf{A}\Theta_{M,N}^b = \left[\begin{array}{cc} x & 0 \\ a & b \\ y & c \end{array}\right]\ .$$

That is, it annihilates the second entry in the top row of the prearray. The scalar quantities $\{a, b, c, x, y\}$ can be identified, as before, by squaring and comparing entries of the resulting equality. This step allows us to make the following identifications very immediately:

$$\begin{array}{rcl} x & = & \xi_M^{b/2}(N)\ , \\ a & = & q_M^b(N+1)\ , \\ y & = & \bar{b}_M(N)\xi_M^{-b/2}(N)\ , \\ bc & = & \gamma_M^{-1/2}(N)f_{M+1}(N+1)\ , \\ b^2 & = & |\bar{f}_{M+1}(N+1)|^2\ , \end{array}$$

where for the last equality we used the following relation that follows immediately from the last two lines of Table 21.7:

$$\left\|q_M^b(N+1)\right\|^2 + \left\|\bar{f}_{M+1}(N+1)\right\|^2 = \lambda\left\|q_M^b(N)\right\|^2 + \left\|\bar{f}_M(N+1)\right\|^2$$

Therefore, $b^2c^2 = \frac{\gamma_{M+1}(N)}{\gamma_M(N)}|\bar{f}_{M+1}(N+1)|^2$ and we can make the identifications:

$$c = \frac{\gamma_{M+1}^{1/2}(N)}{\gamma_M^{1/2}(N)}\ , \quad b = \bar{f}_{M+1}(N+1)\ .$$

A similar argument leads to an array equation for the update of the backward errors. In summary, we obtain the QRD-LSL algorithm (listed in Table 21.8) for the update of the angle-normalized forward and backward prediction errors with prewindowed data that correspond to the minimization problem:

$$\min_{\mathbf{w}_M} \sum_{j=0}^{N} \lambda^{N-j} |d(j) - \mathbf{u}_{M,j}^T \mathbf{w}_M|^2 .$$

The recursions of the table can be shown to collapse, by squaring and comparing terms on both sides of the resulting equality, to several lattice forms that are available in the literature. We forgo the details here.

TABLE 21.8 The QRD Least-Squares Lattice Algorithm

Input. Prewindowed data $\{d(j), u(j)\}$ for $j \geq 1$.

Initialization. For each $M = 0, 1, 2, \ldots, M_{max}$ set

$$\xi_M^{f/2}(0) = 0, \quad \xi_M^{b/2}(-1) = 0, \quad q_M^b(0) = 0 = q_M^f(0)$$

- For each time instant $N \geq 0$ do:

$$\gamma_0(N) = 1, \quad \bar{f}_0(N) = u(N), \quad \bar{b}_0(N) = u(N)$$

- For each $M = 0, 1, 2, \ldots, M_{max} - 1$ do:

$$\begin{bmatrix} \sqrt{\lambda}\xi_M^{b/2}(N-1) & \bar{b}_M(N) \\ \sqrt{\lambda} q_M^b(N) & \bar{f}_M(N+1) \\ 0 & \gamma_M^{1/2}(N) \end{bmatrix} \Theta_{M,N}^b = \begin{bmatrix} \xi_M^{b/2}(N) & 0 \\ q_M^b(N+1) & \bar{f}_{M+1}(N+1) \\ b_M(N)\xi_M^{-b/2}(N) & \gamma_{M+1}^{1/2}(N) \end{bmatrix}$$

$$\begin{bmatrix} \sqrt{\lambda}\xi_M^{f/2}(N) & \bar{f}_M(N+1) \\ \sqrt{\lambda} q_M^f(N) & \bar{b}_M(N) \end{bmatrix} \Theta_{M,N+1}^f = \begin{bmatrix} \xi_M^{f/2}(N+1) & 0 \\ q_M^f(N+1) & \bar{b}_{M+1}(N+1) \end{bmatrix}$$

The orthogonal matrices $\Theta_{M,N}^b$ and $\Theta_{M,N+1}^f$ are chosen so as to annihilate the (1, 2) entries in the corresponding postarrays.

- end

◇ end

21.8.6 The Filtering or Joint Process Array

We now return to the estimation of the sequence $\{d(\cdot)\}$. We argued earlier that if we are given the backward residual vector $\mathbf{b}_{M,N}$ and the estimation residual vector $\mathbf{e}_{M,N}$, then the higher-order estimation residual vector $\mathbf{e}_{M+1,N}$ can be obtained by projecting $\mathbf{e}_{M,N}$ onto $\mathbf{b}_{M,N}$ and using the corresponding residual vector as $\mathbf{e}_{M+1,N}$.

Arguments similar to what we have done in the previous section will readily show that the array for the joint process estimation problem is the following: define the angle-normalized residual

$$\bar{e}_M(i) = e_M(i)\gamma_M^{-1/2}(i) = [d(i) - \mathbf{u}_{M,i}^T \mathbf{w}_{M,i}]\gamma_M^{-1/2}(i) ,$$

as well as the scalar quantity

$$q_M^d(N) = \frac{\rho_M(N)}{\xi_M^{b/2}(N)} .$$

Then the array for the filtering process is what is shown in Table 21.9. Note that it uses precisely the same rotation as the first array in the QRD-LSL algorithm. Hence, the second line in the above array can be included as one more line in the first array of QRD-LSL, thus completing the algorithm to also include the joint-process estimation part.

TABLE 21.9 Array for Joint Process Estimation

```
*Input.* Prewindowed data $\{d(j), u(j)\}$ for $j \geq 1$.

*Initialization.* For each $M = 0, 1, 2, \ldots, M_{max}$ set
        $\xi_M^{b/2}(-1) = 0, \quad q_M^d(-1) = 0, \quad q_M^b(0) = 0$

• For each time instant $N \geq 0$ do:

        $\gamma_0(N) = 1, \quad \bar{e}_0(N) = d(N), \quad \bar{b}_0(N) = u(N)$

  • For each $M = 0, 1, 2, \ldots, M_{max} - 1$ do:

    $\begin{bmatrix} \sqrt{\lambda}\xi_M^{b/2}(N-1) & \bar{b}_M(N) \\ \sqrt{\lambda}q_M^d(N-1) & \bar{e}_M(N) \end{bmatrix} \Theta_{M,N}^b = \begin{bmatrix} \xi_M^{b/2}(N) & 0 \\ q_M^d(N) & \bar{e}_{M+1}(N) \end{bmatrix}$

    where the orthogonal matrix $\Theta_{M,N}^b$ is the same as in the
    QRD-LSL algorithm.
  • end
◇ end
```

21.9 Concluding Remarks

The intent of this chapter was to provide an overview of the fundamentals of recursive least-squares estimation, with emphasis on array formulations of the varied algorithms (slow or fast) that are available for this purpose. More details and related discussion can be found in several of the references indicated in this section. The references are not intended to be complete but rather indicative of the work in the different areas. More complete lists can be found in several of the textbooks mentioned herein.

References

Detailed discussions on the different forms of RLS adaptive algorithms and their potential applications can be found in:

[1] Haykin, S., *Adaptive Filter Theory*, 3rd ed., Prentice-Hall, Englewood Cliffs, NJ, 1996.

[2] Proakis, J.G., Rader, C.M., Ling, F., and Nikias, C.L., *Advanced Digital Signal Processing*, Macmillan, New York, 1992.

[3] Honig, M.L. and Messerschmitt, D.G., *Adaptive Filters — Structures, Algorithms and Applications*, Kluwer Academic Publishers, 1984.

[4] Orfanidis, S.J., *Optimum Signal Processing*, 2nd ed., McGraw-Hill, New York, 1988.

[5] Kalouptsidis, N. and Theodoridis, S., *Adaptive System Identification and Signal Processing Algorithms*, Prentice-Hall, Englewood Cliffs, NJ, 1993.

The array formulation that we emphasized in this chapter is motivated by the state-space approach developed in

[6] Sayed, A.H. and Kailath, T., A state-space approach to adaptive RLS filtering, *IEEE Signal Processing Magazine*, 11(3), 18–60, July 1994.

This reference also clarifies the connections between adaptive RLS filtering and Kalman filter theory and treats other forms of lattice filters.

A detailed discussion of the square-root formulation in the context of Kalman filtering can be found in

[7] Morf, M. and Kailath, T. Square root algorithms for least squares estimation, *IEEE Trans. Automatic Control*, AC-20(4), 487–497, Aug. 1975.

Further motivation, and earlier discussion, on lattice algorithms can be found in several places in the literature:

[8] Lee, D.T.L., Morf, M., and Friedlander, B., Recursive least-squares ladder estimation algorithms, *IEEE Trans. Circuits and Systems*, CAS-28(6), 467–481, June 1981.

[9] Friedlander, B., Lattice filters for adaptive processing, *Proc. IEEE*, 70(8), 829–867, Aug. 1982.

[10] Lev-Ari, H., Kailath, T., and Cioffi, J., Least squares adaptive lattice and transversal filters: a unified geometrical theory, *IEEE Trans. Information Theory*, IT-30(2), 222–236, March, 1984.

The fast fixed-order recursive least-squares algorithms (FTF and FAEST) were independently derived in

[11] Carayannis, G., Manolakis, D., and Kalouptsidis, N., A fast sequential algorithm for least squares filtering and prediction, *IEEE Trans. Acoustics, Speech, and Signal Processing*, ASSP-31(6), 1394–1402, Dec. 1983.

[12] Cioffi, J., and Kailath, T., Fast recursive-least-squares transversal filters for adaptive filtering, *IEEE Trans. Acoustics, Speech and Signal Processing*, ASSP-32, 304–337, April 1984.

These algorithms, however, suffer from numerical instability problems. Some variables that are supposed to remain positive or bounded by one may lose this property due to roundoff errors. A treatment of these issues appears in

[13] Slock, D.T.M. and Kailath, T., Numerically stable fast transversal filters for recursive least squares adaptive filtering, *IEEE Trans. Signal Processing*, SP-39(1), 92–114, Jan. 1991.

More discussion on the QRD least-squares lattice filter, including alternative derivations that are based on the QR decomposition of certain data matrices, can be found in the references:

[14] Cioffi, J., The fast adaptive rotor's RLS algorithm, *IEEE Trans. Acoustics, Speech and Signal Processing*, ASSP-38, 631–653, 1990.

[15] Proudler, I.K., McWhirter, J.G., and Shepherd, T.J., Computationally efficient QR decomposition approach to least squares adaptive filtering, *IEE Proc.*, 138(4), 341–353, Aug. 1991.

[16] Regalia, P.A. and Bellanger, M.G., On the duality between fast QR methods and lattice methods in least squares adaptive filtering, *IEEE Trans. Signal Processing*, 39(4), 879–891, April 1991.

[17] Yang, B. and Böhme, J.F., Rotation-based RLS algorithms: unified derivations, numerical properties, and parallel implementations, *IEEE Trans. Signal Processing*, SP-40(5), 1151–1167, May 1992.

More discussion and examples of elementary and square-root free rotations and Householder transformations can be found in:

[18] Golub, G.B. and Van Loan, C.F., *Matrix Computations*, 2nd ed., The Johns Hopkins University Press, Baltimore, MD, 1989.

[19] Rader, C.M. and Steinhardt, A.O., Hyperbolic householder transformations, *IEEE Trans. Acoustics, Speech and Signal Processing*, ASSP-34(6), 1589–1602, Dec. 1986.

[20] Bojanczyk, A.W. and Steinhardt, A.O., Stabilized hyperbolic householder transformations, *IEEE Trans. Acoustics, Speech, and Signal Processing*, ASSP-37(8), 1286–1288, Aug. 1989.

[21] Hsieh, S.F., Liu, K.J.R., and Yao, K., A unified square-root-free approach for QRD-based recursive least-squares estimation, *IEEE Trans. Signal Processing*, SP-41(3), 1405–1409, March 1993.

Fast fixed-order adaptive algorithms that consider different choices of the initial weighting matrix Π_0, and also the case of data that is not necessarily prewindowed, can be found in:

[22] Houacine, A., Regularized fast recursive least squares algorithms for adaptive filtering, *IEEE Trans. Signal Processing*, SP-39(4), 860–870, April 1991.

Gauss' original exposition of the least-squares criterion can be found in:

[23] Gauss, C.F., *Theory of the Motion of Heavenly Bodies*, Dover, New York, 1963 (English translation of *Theoria Motus Corporum Coelestium, 1809*).