Chapter 3
Conventional and Inverse QRD-RLS Algorithms

José A. Apolinário Jr. and Maria D. Miranda

Abstract This chapter deals with the basic concepts used in the recursive least-squares (RLS) algorithms employing conventional and inverse QR decomposition. The methods of triangularizing the input data matrix and the meaning of the internal variables of these algorithms are emphasized in order to provide details of their most important relations. The notation and variables used herein will be exactly the same used in the previous introductory chapter. For clarity, all derivations will be carried out using real variables and the final presentation of the algorithms (tables and pseudo-codes) will correspond to their complex-valued versions.

3.1 The Least-Squares Problem and the QR Decomposition

We start by introducing the weighted least-squares (WLS) filtering problem for the identification of a linear system [1]. To this end, we consider two sets of variables, $d(\ell)$ and $x(\ell)$, and the errors $\bar{e}(\ell)$, for $0 \leq \ell \leq k$. The set $d(\ell)$ is the response of an unknown system at time-instant ℓ when the input is the set of variables $x(\ell)$, with $x(\ell) = 0$ for $\ell < 0$. The error at time-instant ℓ is defined as $\bar{e}(\ell) = d(\ell) - \mathbf{w}^{\mathrm{T}}(k)\mathbf{x}(\ell)$, $\mathbf{w}(k)$ being the filter coefficient vector. These errors and the variables $d(\ell)$ and $x(\ell)$ are attenuated by the factor $\lambda^{(k-\ell)/2}$, $0 \ll \lambda < 1$.

José A. Apolinário Jr.
Military Institute of Engineering (IME), Rio de Janeiro – Brazil
e-mail: apolin@ieee.org

Maria D. Miranda
University of São Paulo (USP), São Paulo – Brazil
e-mail: maria@lcs.poli.usp.br

J.A. Apolinário Jr. (ed.), *QRD-RLS Adaptive Filtering*,
DOI 10.1007/978-0-387-09734-3_3,

Now, to facilitate the problem formulation, we collect the attenuated estimation errors into a $(k+1)\times 1$ vector, written as

$$\mathbf{e}(k) = \begin{bmatrix} \bar{e}(k) & \lambda^{1/2}\bar{e}(k-1) & \cdots & \lambda^{k/2}\bar{e}(0) \end{bmatrix}^{\mathrm{T}}. \tag{3.1}$$

This vector can take the form

$$\mathbf{e}(k) = \mathbf{d}(k) - \widehat{\mathbf{d}}(k), \tag{3.2}$$

where the desired response vector is given by

$$\mathbf{d}(k) = \begin{bmatrix} d(k) & \lambda^{1/2}d(k-1) & \cdots & \lambda^{k/2}d(0) \end{bmatrix}^{\mathrm{T}}, \tag{3.3}$$

and its estimate, obtained from linear weighted averages of observation sequences, is given as

$$\widehat{\mathbf{d}}(k) = \underbrace{\begin{bmatrix} x(k) & x(k-1) & \cdots & x(k-N) \\ \lambda^{1/2}x(k-1) & \lambda^{1/2}x(k-2) & & \vdots \\ & & & \lambda^{(k-N)/2}x(0) \\ \vdots & \vdots & & 0 \\ \lambda^{(k-1)/2}x(1) & \lambda^{(k-1)/2}x(0) & & \vdots \\ \lambda^{k/2}x(0) & 0 & \cdots & 0 \end{bmatrix}}_{\mathbf{X}(k)} \underbrace{\begin{bmatrix} w_0(k) \\ w_1(k) \\ \vdots \\ w_N(k) \end{bmatrix}}_{\mathbf{w}(k)}. \tag{3.4}$$

The input data matrix, denoted here as $\mathbf{X}(k)$, has dimension $(k+1)\times(N+1)$ and can be represented in terms of its $N+1$ columns or its $k+1$ rows, that is,

$$\mathbf{X}(k) = \begin{bmatrix} \mathbf{x}^{(0)}(k)\, \mathbf{x}^{(1)}(k) \cdots \mathbf{x}^{(N)}(k) \end{bmatrix} = \begin{bmatrix} \mathbf{x}^{\mathrm{T}}(k) \\ \lambda^{1/2}\mathbf{x}^{\mathrm{T}}(k-1) \\ \vdots \\ \lambda^{k/2}\mathbf{x}^{\mathrm{T}}(0) \end{bmatrix}. \tag{3.5}$$

Note that $\mathbf{x}(\ell) = [x(\ell)\, x(\ell-1)\, \cdots\, x(\ell-N)]^{\mathrm{T}}$ represents the input regression vector at instant ℓ, $0 \le \ell \le k$, with $N+1$ elements, and $\mathbf{x}^{(i)}(k)$ for $(i = 0, \cdots, N)$ represents the $(i+1)$th column of $\mathbf{X}(k)$.

The Euclidean norm of the weighted estimation error vector corresponds to the deterministic cost function $\xi_D(k)$, the same one used for the recursive least-squares (RLS) algorithm, which is given by

$$\xi_D(k) = \|\mathbf{e}(k)\|^2 = \mathbf{e}^{\mathrm{T}}(k)\mathbf{e}(k) = \sum_{\ell=0}^{k} \lambda^{k-\ell}\bar{e}^2(\ell). \tag{3.6}$$

The WLS filtering problem consists in determining, at instant k, the coefficient vector $\mathbf{w}(k)$ that minimizes $\xi_D(k)$. The optimal solution is obtained when the coefficient vector satisfies [1, 2]

$$\mathbf{w}(k) = \mathbf{R}^{-1}(k)\mathbf{p}(k), \tag{3.7}$$

where

$$\mathbf{R}(k) = \mathbf{X}^{\mathrm{T}}(k)\mathbf{X}(k) \tag{3.8}$$

is the $(N+1) \times (N+1)$ input-data deterministic autocorrelation matrix, and

$$\mathbf{p}(k) = \mathbf{X}^{\mathrm{T}}(k)\mathbf{d}(k) \tag{3.9}$$

is the $(N+1) \times 1$ deterministic cross-correlation vector.

The deterministic autocorrelation matrix $\mathbf{R}(k)$ is considered non-singular; nevertheless, the inverse of $\mathbf{R}(k)$, used in (3.7), can become ill-conditioned, e.g., due to loss of persistence of excitation of the input signal or due to quantization effects [1, 2].

The WLS problem may be solved by means of a QR decomposition which is numerically well-conditioned [3, 4]. This approach is based on the following triangularization of the input data matrix:

$$\mathbf{Q}(k)\mathbf{X}(k) = \begin{bmatrix} \mathbf{0}_{(k-N)\times(N+1)} \\ \\ \mathbf{U}(k) \end{bmatrix}. \tag{3.10}$$

Matrix $\mathbf{U}(k)$ has dimension $(N+1) \times (N+1)$ and a triangular structure as, for example, shown in Figure 3.1. $\mathbf{Q}(k)$ is an unitary matrix with dimension $(k+1) \times (k+1)$ which represents the overall triangularization process. The unicity of the decomposition is yielded through the condition that all elements of the main anti-diagonal of $\mathbf{U}(k)$ are non-negative [4].

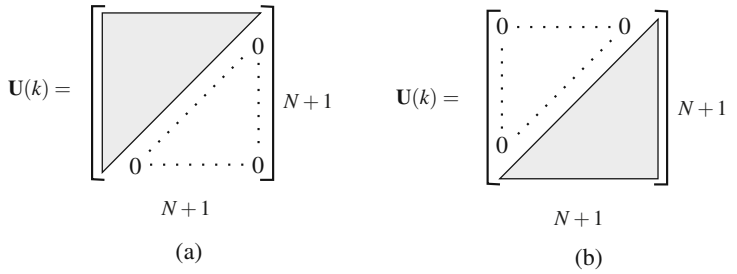

Fig. 3.1 The different triangularizations of $\mathbf{U}(k)$: (a) UPPER and (b) LOWER.

As $\mathbf{X}(k)$ is a $(k+1)\times(N+1)$ matrix, it is interesting to consider the following partition of matrix $\mathbf{Q}(k)$:

$$\mathbf{Q}(k) = \begin{bmatrix} \mathbf{Q}_1(k) \\ \\ \mathbf{Q}_2(k) \end{bmatrix}, \tag{3.11}$$

with $\mathbf{Q}_1(k)$ and $\mathbf{Q}_2(k)$ having dimensions $(k-N)\times(k+1)$ and $(N+1)\times(k+1)$, respectively. Since matrix $\mathbf{Q}(k)$ is unitary, i.e.,

$$\mathbf{Q}^{\mathrm{T}}(k)\mathbf{Q}(k) = \mathbf{Q}(k)\mathbf{Q}^{\mathrm{T}}(k) = \mathbf{I}_{k+1}, \tag{3.12}$$

it follows that

$$\begin{aligned} &\mathbf{Q}_1^{\mathrm{T}}(k)\mathbf{Q}_1(k) + \mathbf{Q}_2^{\mathrm{T}}(k)\mathbf{Q}_2(k) = \mathbf{I}_{k+1}, \\ &\mathbf{Q}_1(k)\mathbf{Q}_1^{\mathrm{T}}(k) = \mathbf{I}_{k-N}, \text{ and} \\ &\mathbf{Q}_2(k)\mathbf{Q}_2^{\mathrm{T}}(k) = \mathbf{I}_{N+1}. \end{aligned} \tag{3.13}$$

The pre-multiplication of (3.10) by $\mathbf{Q}^{\mathrm{T}}(k)$, with (3.12) and (3.11) yields

$$\mathbf{X}(k) = \mathbf{Q}_2^{\mathrm{T}}(k)\mathbf{U}(k). \tag{3.14}$$

Therefore, the deterministic autocorrelation matrix satisfies

$$\mathbf{R}(k) = \mathbf{X}^{\mathrm{T}}(k)\mathbf{X}(k) = \mathbf{U}^{\mathrm{T}}(k)\mathbf{U}(k), \tag{3.15}$$

and matrix $\mathbf{U}(k)$ is referred to as the Cholesky factor of $\mathbf{R}(k)$ [1, 3].

The pre-multiplication of (3.2) by $\mathbf{Q}(k)$, i.e.,

$$\mathbf{e}_q(k) = \mathbf{Q}(k)\mathbf{e}(k) = \begin{bmatrix} \mathbf{e}_{q_1}(k) \\ \mathbf{e}_{q_2}(k) \end{bmatrix} = \begin{bmatrix} \mathbf{d}_{q_1}(k) \\ \mathbf{d}_{q_2}(k) \end{bmatrix} - \begin{bmatrix} \mathbf{O} \\ \mathbf{U}(k) \end{bmatrix} \mathbf{w}(k), \tag{3.16}$$

triangularizes $\mathbf{X}(k)$ and, being $\mathbf{Q}(k)$ a unitary matrix, it will not affect the cost function, i.e., $\|\mathbf{e}_q(k)\|^2 = \|\mathbf{e}(k)\|^2$. The subscripts 1 and 2 indicate the first $k-N$ and the last $N+1$ components of the vector, respectively.

The weighted-square error (or cost function) can be minimized when, by a proper choice of $\mathbf{w}(k)$, the term $\mathbf{d}_{q_2}(k) - \mathbf{U}(k)\mathbf{w}(k)$ becomes zero. The tap-weight coefficients can then be computed as

$$\mathbf{w}(k) = \mathbf{U}^{-1}(k)\mathbf{d}_{q2}(k). \tag{3.17}$$

Note that (3.7) and (3.17) represent different ways to find the same WLS solution. Next, with geometric and linear algebraic arguments, we show how these two representations are related.

When the coefficient vector $\mathbf{w}(k)$ satisfies (3.7), it is easy to figure out that the estimated desired response $\widehat{\mathbf{d}}(k) = \mathbf{X}(k)\mathbf{w}(k)$ and the estimation error vector

$\mathbf{e}(k) = \mathbf{d}(k) - \widehat{\mathbf{d}}(k)$ can be expressed as

$$\widehat{\mathbf{d}}(k) = \mathbf{X}(k)\underbrace{\mathbf{R}^{-1}(k)\mathbf{p}(k)}_{\mathbf{w}(k)} = \mathscr{P}(k)\mathbf{d}(k) \tag{3.18}$$

and

$$\mathbf{e}(k) = [\mathbf{I}_{k+1} - \mathscr{P}(k)]\,\mathbf{d}(k), \tag{3.19}$$

where $\mathscr{P}(k) = \mathbf{X}(k)\mathbf{R}^{-1}(k)\mathbf{X}^{\mathrm{T}}(k)$.

Matrix $\mathscr{P}(k)$ is a projection operator, responsible for projecting the desired response vector in the space spanned by the data matrix columns of $\mathbf{X}(k)$. This projection is orthogonal to the estimation error (orthogonality principle). In this condition, $\mathbf{e}(k)$ is known as the optimum estimation error.

When the norm of $\mathbf{e}(k)$ is minimum, the coefficient vector $\mathbf{w}(k)$ also satisfies (3.17). Then, vector $\widehat{\mathbf{d}}(k) = \mathbf{X}(k)\mathbf{w}(k)$ can also be rewritten using (3.17) and (3.14) as

$$\widehat{\mathbf{d}}(k) = \mathbf{Q}_2^{\mathrm{T}}(k)\mathbf{d}_{q_2}(k) = \mathbf{Q}_2^{\mathrm{T}}(k)\mathbf{Q}_2(k)\mathbf{d}(k), \tag{3.20}$$

and the optimum estimation error vector is given as

$$\mathbf{e}(k) = \left[\mathbf{I}_{k+1} - \mathbf{Q}_2^{\mathrm{T}}(k)\mathbf{Q}_2(k)\right]\mathbf{d}(k) = \mathbf{Q}_1^{\mathrm{T}}(k)\mathbf{Q}_1(k)\mathbf{d}(k). \tag{3.21}$$

Therefore, the projection operator can be defined by matrix $\mathbf{Q}_2(k)$ as

$$\mathscr{P}(k) = \mathbf{Q}_2^{\mathrm{T}}(k)\mathbf{Q}_2(k), \tag{3.22}$$

and the complementary projection operator by matrix $\mathbf{Q}_1(k)$ as

$$\mathscr{P}^{\perp}(k) = \mathbf{I}_{k+1} - \mathscr{P}(k) = \mathbf{Q}_1^{\mathrm{T}}(k)\mathbf{Q}_1(k). \tag{3.23}$$

From the mathematical relationships presented so far, we can observe that matrix $\mathbf{Q}(k)$ rotates the subspace spanned by the columns of matrix $\mathbf{X}(k)$ and by its optimum estimation error $\mathbf{e}(k)$, which is orthogonal to the columns of $\mathbf{X}(k)$. In this case, matrix $\mathbf{Q}(k)$ forces the subspace spanned by the columns of matrix $\mathbf{X}(k)$, of dimension $(N+1)$, to coincide with the subspace spanned by the last $(N+1)$ vectors of the Euclidean space basis, of dimension $(k+1)\times(k+1)$, used in the representation. Hence, the subspace of dimension $(k-N)$, where the optimum estimation error $\mathbf{e}(k)$ lies and which is orthogonal to the space spanned by the columns of $\mathbf{X}(k)$, coincides with the subspace spanned by the other $(k-N)$ vectors from the Euclidean basis. Figure 3.2 illustrates these interpretations. Such transformation affects the input signal, the desired signal, and the projection operator, without modifying the autocorrelation matrix, the cross-correlation vector, and the optimum coefficient vector.

Table 3.1 presents the relationship between the conventional LS (or WLS) method and its QR decomposition counterpart. In the second column, the representations of the main results and the operators in the rotated domain are presented. The

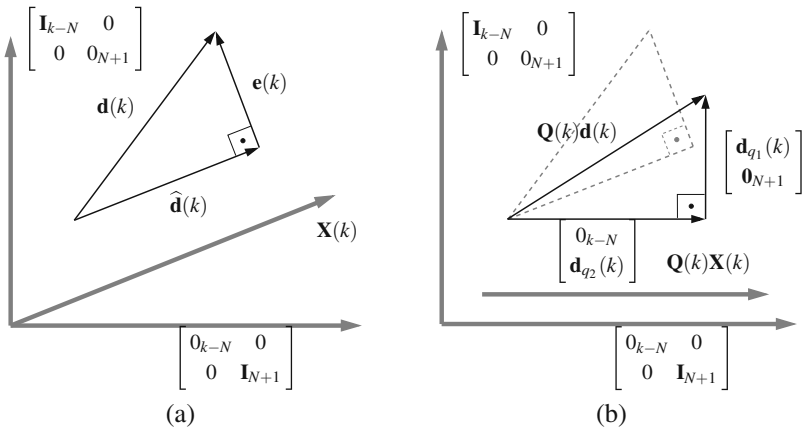

Fig. 3.2 Spacial visualization of the rotated vectors. (a) Estimation of $\mathbf{d}(k)$ projected onto the subspace spanned by the columns of $\mathbf{X}(k)$. (b) Estimation of $\mathbf{d}_q(k)$ projected onto the Euclidean basis space.

last three rows indicate how to calculate the results, which values will not change due to the transformation, using whether the original or the rotated variables. It is worth mentioning that, in the domain of the signals rotated by the matrix $\mathbf{Q}(k)$, the projection operator and its complement assume very simple forms.

Table 3.1 Relationships between methods LS and QRD-LS.

Method	LS	QRD-LS
Data matrix	$\mathbf{X}(k)$	$\begin{bmatrix} \mathbf{O} \\ \mathbf{U}(k) \end{bmatrix} = \mathbf{Q}(k)\mathbf{X}(k)$
Desired response vector	$\mathbf{d}(k)$	$\begin{bmatrix} \mathbf{d}_{q1}(k) \\ \mathbf{d}_{q2}(k) \end{bmatrix} = \mathbf{Q}(k)\mathbf{d}(k)$
Projection operator	$\mathbf{X}(k)\mathbf{R}^{-1}(k)\mathbf{X}^{\mathrm{T}}(k)$	$\begin{bmatrix} \mathbf{O}_{k-N} & \mathbf{O} \\ \mathbf{O} & \mathbf{I}_{N+1} \end{bmatrix}$
Estimated response	$\mathscr{P}(k)\mathbf{d}(k)$	$\begin{bmatrix} 0 \\ \mathbf{d}_{q2}(k) \end{bmatrix} = \begin{bmatrix} \mathbf{O} \\ \mathbf{Q}_2(k) \end{bmatrix} \mathbf{d}(k)$
Complementary projection operator	$\mathscr{P}^{\perp}(k)$	$\begin{bmatrix} \mathbf{I}_{k-N} & \mathbf{O} \\ \mathbf{O} & \mathbf{O}_{N+1} \end{bmatrix}$
Estimation error	$\mathscr{P}^{\perp}(k)\mathbf{d}(k)$	$\begin{bmatrix} \mathbf{d}_{q1}(k) \\ 0 \end{bmatrix} = \begin{bmatrix} \mathbf{Q}_1(k) \\ 0 \end{bmatrix} \mathbf{d}(k)$
Autocorrelation matrix	$\mathbf{R}(k) = \mathbf{X}^{\mathrm{T}}(k)\mathbf{X}(k) = \mathbf{U}^{\mathrm{T}}(k)\mathbf{U}(k)$	
Cross-correlation vector	$\mathbf{p}(k) = \mathbf{X}^{\mathrm{T}}(k)\mathbf{d}(k) = \mathbf{U}^{\mathrm{T}}(k)\mathbf{d}_{q2}(k)$	
Optimum coefficients vector	$\mathbf{w}(k) = \mathbf{R}^{-1}(k)\mathbf{p}(k) = \mathbf{U}^{-1}(k)\mathbf{d}_{q2}(k)$	

The solution for the least-squares problem using the QR decomposition consists basically in the data matrix decomposition (3.10) and, depending on the particular method employed, in executing some of the calculation indicated in (3.20), (3.17), and (3.21). Obviously, the procedure is numerically complex due to the data matrix QR decomposition. But, in practical adaptive filtering applications, the data matrix can be performed recursively. Also, if $\mathbf{U}(k)$ and $\mathbf{d}_{q2}(k)$ are available, the optimum vector $\mathbf{w}(k)$ can be computed with a reduced computational complexity. This is due to the triangular nature of matrix $\mathbf{U}(k)$ and the possibility to use the so-called back-substitution procedure (if assuming a lower triangular matrix). In Appendix 1, we present this procedure as well as the forward substitution procedure, its counterpart for the case of an upper triangular matrix $\mathbf{U}(k)$. It is important to note that, apart from the reduction in the computational burden, if the main diagonal elements are non-zero, the existence of the inverse of $\mathbf{U}(k)$ is ensured [3, 4].

3.2 The Givens Rotation Method

The orthogonal triangularization process may be carried out using various techniques such as Gram–Schmidt orthogonalization, Householder transformation, or Givens rotations. Particularly, Givens rotations leads to an efficient algorithm whereby the triangularization process is updated recursively. As we are interested in the iterative least-squares solution, we consider a triangularization procedure carried out through Givens rotations.

In order to introduce the Givens rotation method, we consider vectors $\mathbf{v}$ and $\mathbf{v}'$ as shown in Figure 3.3. If we represent vector $\mathbf{v}$ as

$$\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} r\cos\theta \\ r\sin\theta \end{bmatrix}, \tag{3.24}$$

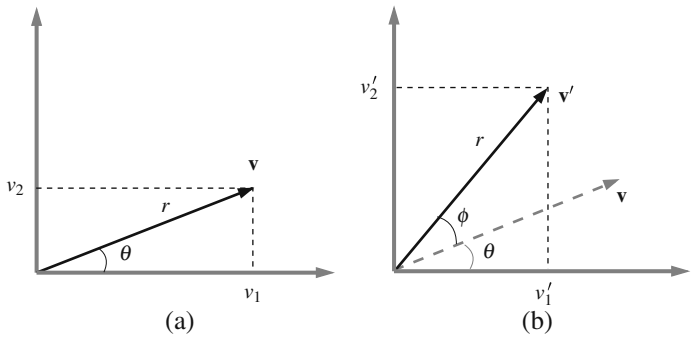

Fig. 3.3 (a) Vector $\mathbf{v}$ projected onto the Euclidean space; (b) Vector $\mathbf{v}$ rotated by ϕ and projected onto the Euclidean space.

vector $\mathbf{v}'$, obtained from a plane rotation of ϕ degrees counterclockwise, can be written as

$$\mathbf{v}' = \begin{bmatrix} v_1' \\ v_2' \end{bmatrix} = \begin{bmatrix} r\cos(\theta+\phi) \\ r\sin(\theta+\phi) \end{bmatrix} = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}. \tag{3.25}$$

For convenience, we rewrite $\mathbf{v}' = G\mathbf{v}$, where

$$G = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix} \tag{3.26}$$

is the elementary Givens rotation matrix.

When the plane rotation is clockwise, $\mathbf{v}'$ is given as

$$\mathbf{v}' = \begin{bmatrix} r\cos(\theta-\phi) \\ r\sin(\theta-\phi) \end{bmatrix} = \begin{bmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}. \tag{3.27}$$

In this case, the elementary Givens rotation matrix takes the form

$$G = \begin{bmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{bmatrix}. \tag{3.28}$$

In both cases (counterclockwise and clockwise), the rotation matrix is unitary, that is, $G^T G = GG^T = I$, and therefore it preserves the norm r of vector $\mathbf{v}$. We consider the counterclockwise rotation throughout this chapter.

If $\phi + \theta = \pi/2$, then $v_1' = 0$, $v_2' = r$ and

$$\begin{bmatrix} 0 \\ r \end{bmatrix} = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}. \tag{3.29}$$

The above development suggests that it is possible to annihilate an element in a two-dimensional vector by using plane rotation. From the relation in (3.29) and noting that $\sin\phi$ and $\cos\phi$ are always constrained by the trigonometric relation

$$\sin^2\phi + \cos^2\phi = 1, \tag{3.30}$$

we have: $\cos\phi = v_2/r$, $\sin\phi = v_1/r$, and $r = \sqrt{v_1^2 + v_2^2}$.

The annihilation using the Givens rotations, as seen above, can be extended to annihilate a specific element in a vector of $N+1$ elements, or a sequence of elements in a vector or in a matrix [5]. Various choices of rotation orders can be used to solve the problem.

In order to illustrate, for a 4×3 matrix formed by a first row and a lower triangular matrix, the rotation angles to annihilate all elements of the first row, we consider

$$\mathbf{A} = \begin{bmatrix} x_{1,1} & x_{1,2} & x_{1,3} \\ 0 & 0 & v_{1,3} \\ 0 & v_{2,2} & v_{2,3} \\ v_{3,1} & v_{3,2} & v_{3,3} \end{bmatrix} \quad \text{and} \quad \mathbf{B} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & u_{1,3} \\ 0 & u_{2,2} & u_{2,3} \\ u_{3,1} & u_{3,2} & u_{3,3} \end{bmatrix}. \tag{3.31}$$

In this example, we have to find a transformation matrix $\mathbf{Q}_\theta(k)$ such that $\mathbf{B} = \mathbf{Q}_\theta(k)\mathbf{A}$. Matrix $\mathbf{Q}_\theta(k)$ must annihilate all elements of the first row of $\mathbf{A}$ from left to right. Three Givens rotations are shown in the following steps.

$$\textbf{Step 1:} \begin{bmatrix} \cos\theta_0 & 0 & 0 & -\sin\theta_0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \sin\theta_0 & 0 & 0 & \cos\theta_0 \end{bmatrix} \begin{bmatrix} x_{1,1} & x_{1,2} & x_{1,3} \\ 0 & 0 & v_{1,3} \\ 0 & v_{2,2} & v_{2,3} \\ v_{3,1} & v_{3,2} & v_{3,3} \end{bmatrix} = \begin{bmatrix} 0 & \bar{x}_{1,2} & \bar{x}_{1,3} \\ 0 & 0 & v_{1,3} \\ 0 & v_{2,2} & v_{2,3} \\ u_{3,1} & u_{3,2} & u_{3,3} \end{bmatrix} \tag{3.32}$$

$$\textbf{Step 2:} \begin{bmatrix} \cos\theta_1 & 0 & -\sin\theta_1 & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta_1 & 0 & \cos\theta_1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & \bar{x}_{1,2} & \bar{x}_{1,3} \\ 0 & 0 & v_{1,3} \\ 0 & v_{2,2} & v_{2,3} \\ u_{3,1} & u_{3,2} & u_{3,3} \end{bmatrix} = \begin{bmatrix} 0 & 0 & \bar{\bar{x}}_{1,3} \\ 0 & 0 & v_{1,3} \\ 0 & u_{2,2} & u_{2,3} \\ u_{3,1} & u_{3,2} & u_{3,3} \end{bmatrix} \tag{3.33}$$

$$\textbf{Step 3:} \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 & 0 \\ \sin\theta_2 & \cos\theta_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & \bar{\bar{x}}_{1,3} \\ 0 & 0 & v_{1,3} \\ 0 & u_{2,2} & u_{2,3} \\ u_{3,1} & u_{3,2} & u_{3,3} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & u_{1,3} \\ 0 & u_{2,2} & u_{2,3} \\ u_{3,1} & u_{3,2} & u_{3,3} \end{bmatrix}. \tag{3.34}$$

This completes the lower triangularization of matrix $\mathbf{A}$.

At this point, it is important to note that the transformation matrix $\mathbf{Q}_\theta(k)$ can be written as a product of Givens rotation matrices given by

$$\mathbf{Q}_\theta(k) = \mathbf{Q}_{\theta_N}(k) \cdots \mathbf{Q}_{\theta_1}(k)\,\mathbf{Q}_{\theta_0}(k) = \prod_{i=0}^{N} \mathbf{Q}_{\theta_i}(k). \tag{3.35}$$

If matrix $\mathbf{U}(k)$ is triangularized as depicted in Figure 3.1, the corresponding $\mathbf{Q}_{\theta_i}(k)$, for $0 \leq I \leq N$, are given as follows:

$$UPPER: \quad \mathbf{Q}_{\theta_i}(k) = \begin{bmatrix} \cos\theta_i(k) & 0^{\mathrm{T}} & -\sin\theta_i(k) & 0^{\mathrm{T}} \\ 0 & \mathbf{I}_i & 0 & 0\cdots 0 \\ \sin\theta_i(k) & 0^{\mathrm{T}} & \cos\theta_i(k) & 0^{\mathrm{T}} \\ 0 & 0\cdots 0 & 0 & \mathbf{I}_{N-i} \end{bmatrix} \tag{3.36}$$

$$LOWER: \quad \mathbf{Q}_{\theta_i}(k) = \begin{bmatrix} \cos\theta_i(k) & 0^{\mathrm{T}} & -\sin\theta_i(k) & 0^{\mathrm{T}} \\ 0 & \mathbf{I}_{N-i} & 0 & 0\cdots 0 \\ \sin\theta_i(k) & 0^{\mathrm{T}} & \cos\theta_i(k) & 0^{\mathrm{T}} \\ 0 & 0\cdots 0 & 0 & \mathbf{I}_i \end{bmatrix}. \tag{3.37}$$

The row and column of each $\cos\theta_i$ and $\sin\theta_i$ are related with the element that we want to annihilate. Only the elements of matrix $\mathbf{A}$ at a position related to the row and column of $\cos\theta_i$ and $\sin\theta_i$, respectively, are affected by the transformation imposed by $\mathbf{Q}_{\theta_i}$. The angles $\theta_i(k)$ in (3.36) and (3.37) are not the same although written, for the sake of simplicity, using the same notation.

3.3 The Conventional QRD-RLS Algorithm

One of the first works that used QR decomposition to solve the RLS problem was proposed by Gentleman [6]. He used a triangular array in order to avoid matrix inversion and proposed a pipelined sequence of Givens rotations to perform the back-substitution process required to solve the associated system of equations. From [6], the conventional QRD-RLS algorithm, as we know it today, was conceived by McWhirter [7]. He was the first to describe a systolic array, using a pipelined sequence of Givens rotations, performing an orthogonal triangularization of the input data matrix and generating the estimated error without having to resort to back-substitution. This section presents the most basic equations of the QRD-RLS algorithms.

Let us start by using the fact that the data matrix in (3.5) can be rewritten as

$$\mathbf{X}(k) = \begin{bmatrix} \mathbf{x}^{\mathrm{T}}(k) \\ \lambda^{1/2}\mathbf{X}(k-1) \end{bmatrix} \tag{3.38}$$

and also that $\mathbf{Q}(k-1)$ is unitary. Thus, the product $\mathbf{Q}(k)\mathbf{X}(k)$ can be written as

$$\mathbf{Q}(k) \underbrace{\begin{bmatrix} 1 & 0^{\mathrm{T}} \\ 0 & \mathbf{Q}^{\mathrm{T}}(k-1) \end{bmatrix} \begin{bmatrix} 1 & 0^{\mathrm{T}} \\ 0 & \mathbf{Q}(k-1) \end{bmatrix}}_{\mathbf{I}} \underbrace{\begin{bmatrix} \mathbf{x}^{\mathrm{T}}(k) \\ \lambda^{1/2}\mathbf{X}(k-1) \end{bmatrix}}_{\mathbf{X}(k)} = \begin{bmatrix} \mathbf{O} \\ \mathbf{U}(k) \end{bmatrix}. \tag{3.39}$$

In the above equation, if we denote the product of the first two matrices on the left side by $\tilde{\mathbf{Q}}(k)$ and compute the multiplication of the remaining two matrices, we obtain

$$\tilde{\mathbf{Q}}(k) \begin{bmatrix} \mathbf{x}^{\mathrm{T}}(k) \\ \mathbf{0}_{(k-N-1)\times(N+1)} \\ \lambda^{1/2}\mathbf{U}(k-1) \end{bmatrix} = \begin{bmatrix} \mathbf{0}_{(k-N)\times(N+1)} \\ \mathbf{U}(k) \end{bmatrix}. \tag{3.40}$$

From (3.40), we see that $\tilde{\mathbf{Q}}(k)$ is a product of Givens rotations matrices that annihilates the current input vector. Moreover, the recursive nature of $\mathbf{Q}(k)$ may be expressed by

$$\mathbf{Q}(k) = \tilde{\mathbf{Q}}(k) \begin{bmatrix} 1 & 0^{\mathrm{T}} \\ 0 & \mathbf{Q}(k-1) \end{bmatrix}. \tag{3.41}$$

Once $\tilde{\mathbf{Q}}(k)$ is responsible for zeroing $\mathbf{x}^{\mathrm{T}}(k)$, as shown in (3.40), its structure includes a sub-matrix $\mathbf{I}_{k-N-1}$, such that it can be represented as

$$\tilde{\mathbf{Q}}(k) = \begin{bmatrix} * & 0 & \cdots & 0 & * & \cdots & * \\ 0 & & & & 0 & & \\ \vdots & & \mathbf{I}_{k-N-1} & & \vdots & & \\ 0 & & & & 0 & & \\ * & 0 & \cdots & 0 & * & \cdots & * \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ * & 0 & \cdots & 0 & * & \cdots & * \end{bmatrix}, \tag{3.42}$$

with the asterisks ($*$) corresponding to the non-zero elements. Although $\mathbf{Q}(k)$ is $(k+1)\times(k+1)$, we can avoid the ever-increasing order characteristic rewriting (3.40) as

$$\begin{bmatrix} 0^{\mathrm{T}} \\ \mathbf{U}(k) \end{bmatrix} = \mathbf{Q}_{\theta}(k) \begin{bmatrix} \mathbf{x}^{\mathrm{T}}(k) \\ \lambda^{1/2}\mathbf{U}(k-1) \end{bmatrix}, \tag{3.43}$$

where $\mathbf{Q}_{\theta}(k)$ is $(N+2)\times(N+2)$ and corresponds to $\tilde{\mathbf{Q}}(k)$ after removing the $\mathbf{I}_{k-N-1}$ section along with the corresponding rows and columns.

Recalling (3.16), we see that $\mathbf{e}_{q_1}(k) = \mathbf{d}_{q_1}(k)$; moreover, (3.41) shows that the product $\mathbf{Q}(k)\mathbf{d}(k)$ can be written as

$$\begin{aligned} \mathbf{Q}(k)\mathbf{d}(k) = \begin{bmatrix} \mathbf{e}_{q_1}(k) \\ \mathbf{d}_{q_2}(k) \end{bmatrix} &= \underbrace{\tilde{\mathbf{Q}}(k) \begin{bmatrix} 1 & 0^{\mathrm{T}} \\ 0 & \mathbf{Q}(k-1) \end{bmatrix}}_{\mathbf{Q}(k)} \underbrace{\begin{bmatrix} d(k) \\ \lambda^{1/2}\mathbf{d}(k-1) \end{bmatrix}}_{\mathbf{d}(k)} \\ &= \tilde{\mathbf{Q}}(k) \begin{bmatrix} d(k) \\ \lambda^{1/2}\mathbf{e}_{q_1}(k-1) \\ \lambda^{1/2}\mathbf{d}_{q_2}(k-1) \end{bmatrix} \\ &= \begin{bmatrix} e_{q_1}(k) \\ \lambda^{1/2}\mathbf{e}_{q_1}(k-1) \\ \mathbf{d}_{q_2}(k) \end{bmatrix}, \end{aligned} \tag{3.44}$$

where the last multiplication can be easily understood if we note in (3.40) and in (3.42) that $\tilde{\mathbf{Q}}(k)$ alter only the first and the last $N+1$ components of a $(k+1)\times 1$ vector.

From (3.44), it is also possible to remove the increasing section of $\tilde{\mathbf{Q}}(k)$, resulting in the following expression:

$$\begin{bmatrix} e_{q_1}(k) \\ \mathbf{d}_{q_2}(k) \end{bmatrix} = \mathbf{Q}_{\theta}(k) \begin{bmatrix} d(k) \\ \lambda^{1/2}\mathbf{d}_{q_2}(k-1) \end{bmatrix}, \tag{3.45}$$

where $e_{q_1}(k)$ is the first element of the *rotated* error vector $\mathbf{e}_q(k)$ and $\mathbf{d}_{q_2}(k)$ is a vector with the last $N+1$ elements of the *rotated* desired signal vector. The

rotated error $e_{q_1}(k)$, with the help of a conversion factor, allows the computation of the prediction error—the difference between $d(k)$ and its prediction from $\mathbf{x}(k)$—without computing the estimate of the desired response. This process is usually termed *joint-process estimation*.

It is important to note that $\mathbf{Q}_\theta(k)$, being responsible for the update of $\mathbf{U}(k)$ as in (3.43), is formed by a product of Givens rotation matrices as shown in (3.35) and its structure will depend on the type of triangularization of $\mathbf{U}(k)$, i.e., whether it is an **upper** or a **lower** triangular matrix. This choice determines two classes of *fast* (reduced complexity) algorithms, as will be seen in Chapter 4. From the example of the previous section, it is possible to obtain, using (3.35), the structure of $\mathbf{Q}_\theta(k)$ for upper and lower triangularization of $\mathbf{U}(k)$; the results, for $N = 2$, are given by the following two matrices:

$$UPPER: \ \mathbf{Q}_\theta(k) = \begin{bmatrix} c\theta_2 c\theta_1 c\theta_0 & -c\theta_2 c\theta_1 s\theta_0 & -c\theta_2 s\theta_1 & -s\theta_2 \\ s\theta_0 & c\theta_0 & 0 & 0 \\ s\theta_1 c\theta_0 & -s\theta_1 s\theta_0 & c\theta_1 & 0 \\ s\theta_2 c\theta_1 c\theta_0 & -s\theta_2 c\theta_1 s\theta_0 & -s\theta_2 s\theta_1 & c\theta_2 \end{bmatrix}, \tag{3.46}$$

and

$$LOWER: \ \mathbf{Q}_\theta(k) = \begin{bmatrix} c\theta_2 c\theta_1 c\theta_0 & -s\theta_2 & -c\theta_2 s\theta_1 & -c\theta_2 c\theta_1 s\theta_0 \\ s\theta_2 c\theta_1 c\theta_0 & c\theta_2 & -s\theta_2 s\theta_1 & -s\theta_2 c\theta_1 s\theta_0 \\ s\theta_1 c\theta_0 & 0 & c\theta_1 & -s\theta_1 s\theta_0 \\ s\theta_0 & 0 & 0 & c\theta_0 \end{bmatrix}, \tag{3.47}$$

where $c\theta_i = \cos\theta_i(k)$ and $s\theta_i = \sin\theta_i(k)$.

The structure of matrix $\mathbf{Q}_\theta(k)$ suggests, in both cases, that it can be partitioned as

$$\mathbf{Q}_\theta(k) = \begin{bmatrix} \gamma(k) & \mathbf{g}^{\mathrm{T}}(k) \\ \mathbf{f}(k) & \mathbf{E}(k) \end{bmatrix}, \tag{3.48}$$

where the structures of vectors $\mathbf{f}(k)$, $\mathbf{g}(k)$, and matrix $\mathbf{E}(k)$ depend on the type of triangularization of the data matrix. On the other hand, for both types of triangularization, the scalar $\gamma(k)$ denotes the product of successive cosine terms, i.e.,

$$\gamma(k) = \prod_{i=0}^{N} \cos\theta_i(k). \tag{3.49}$$

It was shown in [1], and will be detailed later in Section 3.5, that $\gamma(k)$ represents the squared root of the conversion factor between the *a priori* and *a posteriori* output errors for the degree $N+1$ filtering problem.

In order to have all equations of the traditional QR algorithm, let us postmultiply the transposed vector $\mathbf{e}_q^{\mathrm{T}}(k)\mathbf{Q}(k)$ by the pinning vector $[1\ 0\ \cdots\ 0]^{\mathrm{T}}$, then

$$\mathbf{e}_q^{\mathrm{T}}(k)\mathbf{Q}(k)\begin{bmatrix}1\\0\\\vdots\\0\end{bmatrix}=\mathbf{e}^{\mathrm{T}}(k)\mathbf{Q}^{\mathrm{T}}(k)\mathbf{Q}(k)\begin{bmatrix}1\\0\\\vdots\\0\end{bmatrix}=\bar{e}(k), \tag{3.50}$$

where $\bar{e}(k)$ is the first element of the error vector defined in (3.1) and represents the *a posteriori* estimation error $\varepsilon(k)$, defined in Chapter 2 as

$$\varepsilon(k)=d(k)-\mathbf{x}^{\mathrm{T}}(k)\mathbf{w}(k)=\bar{e}(k). \tag{3.51}$$

From Equations (3.41) and (3.48) and the fact that $\mathbf{Q}_\theta(k)$ is $\tilde{\mathbf{Q}}(k)$ after removing the $k-N-1$ increasing columns and rows, we can conclude that $\mathbf{Q}(k)[1\ 0\ \cdots\ 0]^{\mathrm{T}}=[\gamma(k)\ 0\ \cdots\ 0\ \mathbf{f}^{\mathrm{T}}(k)]^{\mathrm{T}}$. Once $\mathbf{e}_{q_2}(k)$ is a null vector (keep in mind that $\mathbf{w}(k)$ in (3.16) was chosen in order to make it zero), it is possible to see that

$$\varepsilon(k)=e_{q_1}(k)\gamma(k). \tag{3.52}$$

This equation was first noted by McWhirter in [7] and is particularly useful in applications where the coefficients of the adaptive filter are not explicitly necessary since we can obtain $\varepsilon(k)$, the *a posteriori* output error, without computing $\mathbf{w}(k)$. If, however, the coefficient vector $\mathbf{w}(k)$ is needed, it can be computed by solving $\mathbf{U}(k)\mathbf{w}(k)=\mathbf{d}_{q2}(k)$. In this case, due to the fact that the Cholesky matrix $\mathbf{U}(k)$ is triangular, a matrix inversion can be avoided, e.g., with a *forward* or a *back-substitution* procedure (see Appendix 1 for further details).

The equations of the conventional QR algorithm (complex version) are presented in Table 3.2 where the type of triangularization used has no influence on the performance of the conventional QRD-RLS algorithm. A pseudo-code of a lower triangularization implementation is available in Appendix 3 (Table 3.5).

In the next section, we provide hints about the initialization of the triangularization procedure. Appendix 2 presents ways of avoiding the square-root operation in

Table 3.2 The conventional QRD-RLS equations.

QRD-RLS
for each k { Obtaining $\mathbf{Q}_\theta(k)$ and updating $\mathbf{U}(k)$: $\begin{bmatrix}0^{\mathrm{T}}\\\mathbf{U}(k)\end{bmatrix}=\mathbf{Q}_\theta(k)\begin{bmatrix}\mathbf{x}^{\mathrm{H}}(k)\\\lambda^{1/2}\mathbf{U}(k-1)\end{bmatrix}$ Obtaining $\gamma(k)$: $\gamma(k)=\prod_{i=0}^{N}\cos\theta_i(k)$ Obtaining $e_{q_1}(k)$ and updating $\mathbf{d}_{q_2}(k)$: $\begin{bmatrix}e_{q_1}(k)\\\mathbf{d}_{q_2}(k)\end{bmatrix}=\mathbf{Q}_\theta(k)\begin{bmatrix}d^*(k)\\\lambda^{1/2}\mathbf{d}_{q_2}(k-1)\end{bmatrix}$ Obtaining $\varepsilon(k)$: $\varepsilon(k)=e_{q_1}^*(k)\gamma(k)$ % *a posteriori* error: $d(k)-\mathbf{w}^{\mathrm{H}}(k)\mathbf{x}(k)$ }

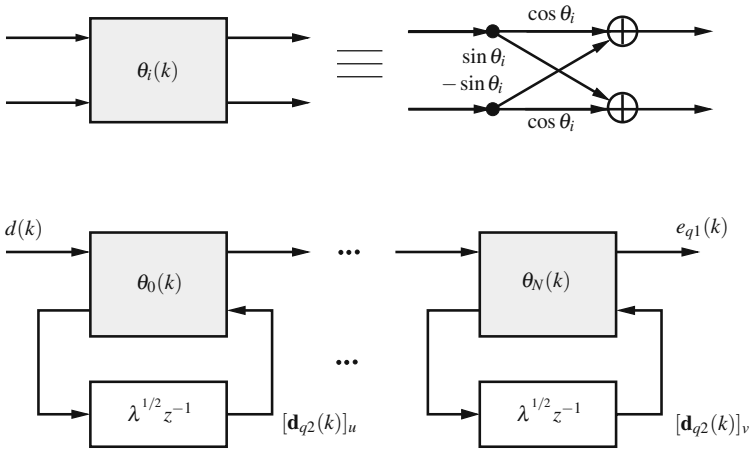

Fig. 3.4 McWhirter structure: Equation (3.45) implemented as a cascade of first-order orthogonal filters.

QRD-RLS algorithms. Now, to close this section, it is worth noting that, since $\mathbf{Q}_\theta(k)$ is a product of Givens rotation matrices, the equation system in (3.45) becomes the cascade of first-order orthogonal filters. Figure 3.4 depicts the operation carried out in (3.45). Due to the work of McWhirter [7], each orthogonal filter in [8] was named a McWhirter structure. In this figure, when we use an upper triangular Cholesky factor, we make $u = 0$ and $v = N$; this means that we update the elements of vector $\mathbf{d}_{q2}(k)$ from the first to the last one. Otherwise, when a lower triangular matrix is used, the updating is from the last element ($u = N$) to the first one ($v = 0$).

3.4 Initialization of the Triangularization Procedure

In order to run the QRD-RLS algorithm at time-instant $k = 0$, we need vector $\mathbf{d}_{q_2}(-1)$ and matrix $\mathbf{U}(-1)$. Assuming pre-windowing, a natural choice would be $\mathbf{d}_{q_2}(-1) = \mathbf{0}_{N+1}$ and $\mathbf{U}(-1) = \mathbf{0}_{(N+1)\times(N+1)}$. In that case, the choice $\mathbf{U}(-1) = \mathbf{0}_{(N+1)\times(N+1)}$ would lead to a non-singular matrix. In order to solve this problem, two strategies are possible: the *exact initialization* and the *soft-start*, as in the RLS algorithm [1, 9].

The **exact initialization** procedure comprises a period of $N+1$ samples, from $k = 0$ to N, during which the estimation error is zero. At $k = N+1$, the initialization period is completed and the estimation error may assume a value different than zero by executing the steps of the algorithm in Table 3.2.

If we are interested in an upper triangularization procedure, since $\mathbf{X}(N)$ is already upper triangular, nothing needs to be done. The exact initialization in this case is carried out as detailed in [2, Chapter 9] which corresponds to the exact initialization

of the RLS algorithm, using back-substitution to obtain the coefficient vector. At $k = N+1$, when $x(N+1)$ is available, the matrix is no longer triangular and $N+1$ Givens rotations are necessary to annihilate all elements of the first row and the QRD-RLS equations start to be used with $\mathbf{U}(N) = \mathbf{X}(N)$.

If we are working with a lower triangular Cholesky factor, then the data matrix needs to be transformed before $k = N+1$. The complete transformation can be carried out with $N+1$ Givens rotations at $k = N$. In this case, we begin by annihilating the elements of column one until column N. For each column i, only the elements of row $j = 1, \ldots (N-i+1)$ are annihilated. The exact initialization procedure of a 3×3 lower triangular Cholesky matrix, that is, order $N = 2$, for $k = 2$, is described below.

$$\begin{bmatrix} c_0 & 0 & -s_0 \\ 0 & 1 & 0 \\ s_0 & 0 & c_0 \end{bmatrix} \begin{bmatrix} x(2) & x(1) & x(0) \\ \lambda^{1/2}x(1) & \lambda^{1/2}x(0) & 0 \\ \lambda\, x(0) & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & \bar{x}(1) & \bar{x}(0) \\ \lambda^{1/2}x(1) & \lambda^{1/2}x(0) & 0 \\ \bar{u}(3,1) & \bar{u}(3,2) & \bar{u}(3,3) \end{bmatrix} \tag{3.53}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & c_1 & -s_1 \\ 0 & s_1 & c_1 \end{bmatrix} \begin{bmatrix} 0 & \bar{x}(1) & \bar{x}(0) \\ \lambda^{1/2}x(1) & \lambda^{1/2}x(0) & 0 \\ \bar{u}(3,1) & \bar{u}(3,2) & \bar{u}(3,3) \end{bmatrix} = \begin{bmatrix} 0 & \bar{x}(1) & \bar{x}(0) \\ 0 & \bar{u}(2,2) & \bar{u}(2,3) \\ u(3,1) & u(3,2) & u(3,3) \end{bmatrix} \tag{3.54}$$

$$\begin{bmatrix} c_2 & -s_2 & 0 \\ s_2 & c_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & \bar{x}(1) & \bar{x}(0) \\ 0 & \bar{u}(2,2) & \bar{u}(2,3) \\ u(3,1) & u(3,2) & u(3,3) \end{bmatrix} = \begin{bmatrix} 0 & 0 & u(1,3) \\ 0 & u(2,2) & u(2,3) \\ u(3,1) & u(3,2) & u(3,3) \end{bmatrix} \tag{3.55}$$

As vector $\mathbf{e}(N) = \mathbf{d}(N) - \mathbf{X}(N)\mathbf{w}(N)$ is $(N+1) \times 1$, then $\mathbf{d}_{q_2}(N) = \mathbf{Q}(N)\mathbf{d}(N)$ and

$$\mathbf{dq}(2) = \underbrace{\begin{bmatrix} c_2 & -s_2 & 0 \\ s_2 & c_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_1 & -s_1 \\ 0 & s_1 & c_1 \end{bmatrix} \begin{bmatrix} c_0 & 0 & -s_0 \\ 0 & 1 & 0 \\ s_0 & 0 & c_0 \end{bmatrix}}_{\mathbf{Q}(N)} \underbrace{\begin{bmatrix} d(2) \\ \lambda^{1/2}d(1) \\ \lambda\, d(0) \end{bmatrix}}_{\mathbf{d}(N)}. \tag{3.56}$$

The initialization of the lower triangular Cholesky factor can also be carried out in a recursive way [1]. For this case, it is assumed that $x(k) = 0$ for $k < 0$ and, for $k = 0, \ldots, N$, and the same update procedures for the lower triangular $\mathbf{U}(k)$ and vector $\mathbf{dq}(k)$ described in Table 3.2 are used. The iterative exact initialization procedure of a 3×3 lower triangular matrix ($N = 2$), for $k = 2$ is described as follows:
At $k = 0$:

$$\underbrace{\mathbf{Q}_{\theta_0}(0)}_{\mathbf{Q}_\theta(0)} \begin{bmatrix} x(0) & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ x(0) & 0 \end{bmatrix} \tag{3.57}$$

At $k = 1$:

$$\underbrace{\mathbf{Q}_{\theta_1}(1)\,\mathbf{Q}_{\theta_0}(1)}_{\mathbf{Q}_\theta(1)} \begin{bmatrix} x(1) & x(0) \\ 0 & 0 \\ \lambda^{1/2}x(0) & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & \bar{\bar{x}} \\ \bar{\bar{x}} & \bar{\bar{x}} \end{bmatrix} \tag{3.58}$$

At $k=2$:

$$\underbrace{\mathbf{Q}_{\theta_2}(2)\,\mathbf{Q}_{\theta_1}(2)\,\mathbf{Q}_{\theta_0}(2)}_{\mathbf{Q}_\theta(2)}\begin{bmatrix} x(2) & x(1) & x(0) \\ 0 & 0 & 0 \\ 0 & \lambda^{1/2}\bar{\bar{x}} & 0 \\ \lambda^{1/2}\bar{\bar{x}} & \lambda^{1/2}\bar{\bar{x}} & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & u(1,3) \\ 0 & u(2,2) & u(2,3) \\ u(3,1) & u(3,2) & u(3,3) \end{bmatrix}. \quad (3.59)$$

It is worth mentioning that, in an infinite-precision environment, this exact initialization of the QRD-RLS algorithm is equivalent to the exact initialization of the RLS algorithm.

For the case of the **soft-start** procedure, one should choose $\mathbf{U}(-1)=\delta\mathbf{J}$, where $\mathbf{J}$ is the reversal matrix and δ is a regularization parameter with a value proportional to the standard deviation of the input signal. Note that $\mathbf{Jv}$ reverses vector $\mathbf{v}$: Its first element becomes the last one and vice-versa. The *soft-start* strategy is simpler than the exact initialization and its effect becomes negligible when k increases.

Regarding the soft-constrained initialization, it is relevant to note that:

- As in the conventional RLS algorithm [1], the soft initialization causes a bias and, if $\lambda<1$, this bias tends to zero as k increases.
- In [10], the full quantitative analysis of the dynamic range of all internal quantities of the QRD-RLS algorithm was presented; it was also shown in this reference, for the case of fixed-point arithmetics, that when the value of the regularization parameter is approximately $\delta=\sigma_x/\sqrt{1-\lambda}$, σ_x^2 being the variance of the input signal, overflow in the internal variable with soft initialization can be avoided.

3.5 On the $\mathbf{Q}_\theta(k)$ Matrix

Different versions of QRD-RLS algorithms can be obtained from adequate interpretation of matrix $\mathbf{Q}_\theta(k)$. Below, based on (3.48), we present the key relations to allow such interpretation.

- Observing the fact that $\mathbf{Q}_\theta(k)$ is unitary, that is,

$$\begin{aligned}\mathbf{I}_{N+2} &= \mathbf{Q}_\theta(k)\mathbf{Q}_\theta^{\mathrm{T}}(k) = \begin{bmatrix} \gamma(k) & \mathbf{g}^{\mathrm{T}}(k) \\ \mathbf{f}(k) & \mathbf{E}(k) \end{bmatrix}\begin{bmatrix} \gamma(k) & \mathbf{f}^{\mathrm{T}}(k) \\ \mathbf{g}(k) & \mathbf{E}^{\mathrm{T}}(k) \end{bmatrix} \\ &= \mathbf{Q}_\theta^{\mathrm{T}}(k)\mathbf{Q}_\theta(k) = \begin{bmatrix} \gamma(k) & \mathbf{f}^{\mathrm{T}}(k) \\ \mathbf{g}(k) & \mathbf{E}^{\mathrm{T}}(k) \end{bmatrix}\begin{bmatrix} \gamma(k) & \mathbf{g}(k)^{\mathrm{T}} \\ \mathbf{f}(k) & \mathbf{E}(k) \end{bmatrix}, \end{aligned} \quad (3.60)$$

then

$$\mathbf{f}(k) = -\gamma^{-1}(k)\mathbf{E}(k)\mathbf{g}(k), \text{ and} \quad (3.61)$$

$$\mathbf{g}(k) = -\gamma^{-1}(k)\mathbf{E}^{\mathrm{T}}(k)\mathbf{f}(k). \quad (3.62)$$

- Replacing $\mathbf{Q}_\theta(k)$, as in (3.48), in (3.43), we have:

$$\begin{bmatrix} \gamma(k) & \mathbf{g}^{\mathrm{T}}(k) \\ \mathbf{f}(k) & \mathbf{E}(k) \end{bmatrix} \begin{bmatrix} \mathbf{x}^{\mathrm{T}}(k) \\ \lambda^{1/2}\mathbf{U}(k-1) \end{bmatrix} = \begin{bmatrix} 0^{\mathrm{T}} \\ \mathbf{U}(k) \end{bmatrix}; \tag{3.63}$$

  then, the next two relations follow:

$$\mathbf{f}(k)\mathbf{x}^{\mathrm{T}}(k) + \lambda^{1/2}\mathbf{E}(k)\mathbf{U}(k-1) = \mathbf{U}(k), \text{ and} \tag{3.64}$$

$$\mathbf{g}(k) = -\gamma(k)\lambda^{-1/2}\mathbf{U}^{-\mathrm{T}}(k-1)\mathbf{x}(k). \tag{3.65}$$

- Observing the fact that $\mathbf{Q}_\theta(k)$ is unitary, we see that the norm of the expression in (3.63) obeys

$$\mathbf{U}^{\mathrm{T}}(k)\mathbf{U}(k) = \mathbf{x}(k)\mathbf{x}^{\mathrm{T}}(k) + \lambda\mathbf{U}^{\mathrm{T}}(k-1)\mathbf{U}(k-1). \tag{3.66}$$

  If we pre-multiply (3.66) by $\mathbf{U}^{-\mathrm{T}}(k)$, and confront the result

$$\mathbf{U}(k) = \mathbf{U}^{-\mathrm{T}}(k)\mathbf{x}(k)\mathbf{x}^{\mathrm{T}}(k) + \lambda\mathbf{U}^{-\mathrm{T}}(k)\mathbf{U}^{\mathrm{T}}(k-1)\mathbf{U}(k-1) \tag{3.67}$$

  with (3.64), we see that $\mathbf{f}(k)$ and $\mathbf{E}(k)$ can be given by

$$\mathbf{f}(k) = \mathbf{U}^{-\mathrm{T}}(k)\mathbf{x}(k), \text{ and} \tag{3.68}$$

$$\mathbf{E}(k) = \lambda^{1/2}\mathbf{U}^{-\mathrm{T}}(k)\mathbf{U}^{\mathrm{T}}(k-1). \tag{3.69}$$

The above relations are common to both triangularization methods. Especially (3.65), (3.68), and (3.69) are key relations for the comprehension of other algorithms of the QR family.

With the above, we can relate some expressions of the RLS algorithm, from the previous chapter, with their corresponding QRD-RLS counterparts. These relations, linking both algorithms, are shown in Table 3.3 where the expressions of the first column rise naturally from the RLS algorithm while the equations in the second column were basically obtained from (3.60).

Table 3.3 Relating equivalent expressions from the RLS and the QRD-RLS algorithms.

RLS	QRD-RLS
$\gamma^2(k) = 1 - \mathbf{x}^{\mathrm{T}}(k)\mathbf{R}^{-1}(k)\mathbf{x}(k)$	$\gamma^2(k) = 1 - \mathbf{f}^{\mathrm{T}}(k)\mathbf{f}(k) = 1 - \|\|\mathbf{f}(k)\|\|^2$
$\gamma^{-2}(k) = 1 + \lambda^{-1}\mathbf{x}^{\mathrm{T}}(k)\mathbf{R}^{-1}(k-1)\mathbf{x}(k)$	$\gamma^{-2}(k) = 1 + \|\|\gamma^{-1}(k)\mathbf{g}(k)\|\|^2$
$\lambda^{-1}\gamma^2(k)\mathbf{R}^{-1}(k-1)\mathbf{x}(k) = \mathbf{R}^{-1}(k)\mathbf{x}(k)$	$\mathbf{g}(k) = -\gamma^{-1}(k)\mathbf{E}^{\mathrm{T}}(k)\mathbf{f}(k)$
$\mathbf{x}(k)\mathbf{x}^{\mathrm{T}}(k) = \mathbf{R}(k) - \lambda\mathbf{R}(k-1)$	$\mathbf{f}(k)\mathbf{f}^{\mathrm{T}}(k) = \mathbf{I} - \mathbf{E}(k)\mathbf{E}^{\mathrm{T}}(k)$
$\mathbf{R}^{-1}(k)\mathbf{x}(k)\mathbf{x}^{\mathrm{T}}(k) = \mathbf{I} - \lambda\mathbf{R}^{-1}(k)\mathbf{R}(k-1)$	$\mathbf{g}(k)\mathbf{g}^{\mathrm{T}}(k) = \mathbf{I} - \mathbf{E}^{\mathrm{T}}(k)\mathbf{E}(k)$

We next interpret the elements of $\mathbf{Q}_\theta(k)$ beginning with $\gamma(k)$. If $\mathbf{Q}_\theta(k)$ is regarded as in (3.48), the first element of the equation system in (3.45) can be written as

$$e_{q1}(k) = \gamma(k)d(k) + \mathbf{g}^{\mathrm{T}}(k)\lambda^{1/2}\mathbf{d}_{q_2}(k-1). \tag{3.70}$$

From (3.65) and using (3.17), the rotated error can be expressed as

$$e_{q1}(k) = \gamma(k)e(k), \tag{3.71}$$

where $e(k) = d(k) - \mathbf{x}^{\mathrm{T}}(k)\mathbf{w}(k-1)$ is the *a priori* estimation error as defined in Chapter 2. With (3.52) and (3.71), it follows that

$$\varepsilon(k) = \gamma^2(k)e(k), \tag{3.72}$$

and $\gamma^2(k)$ represents the conversion factor between the *a priori* and the *a posteriori* output errors for the degree $N+1$ filtering problem. Note that, this same conversion factor is also used in the context of the RLS algorithm.

In order to interpret the other elements of $\mathbf{Q}_\theta(k)$, it is necessary to apply the QR decomposition to the problems of forward and backward predictions. In Sections 3.5.1 and 3.5.2, we consider the scheme shown in Figure 3.5 to present the prediction of a past sample $x(k-N-1)$ from vector $\mathbf{x}(k)$ (backward prediction) and the prediction of the current sample $x(k)$ from vector $\mathbf{x}(k-1)$ (forward prediction).

Throughout the next subsections, the $(k+1)\times i$ input data matrix is denoted as $\mathbf{X}^{(i)}(k)$ and all variables of backward and forward predictors, related to order i predictors ($i+1$ prediction coefficients), are indicated with the superscript $^{(i+1)}$,

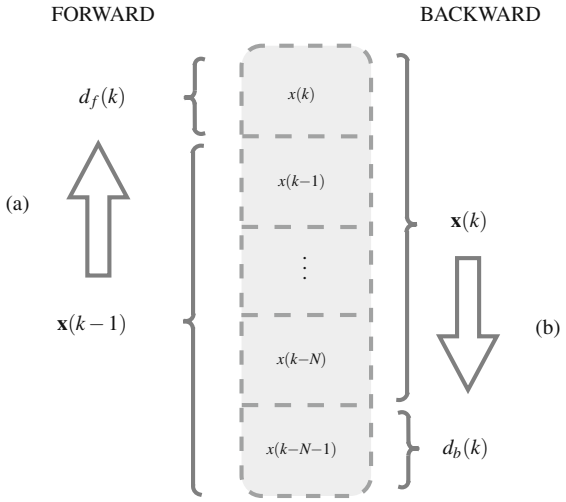

Fig. 3.5 Signal prediction at instant k and order N. (a) Forward prediction: sample $x(k)$ is predicted from vector $\mathbf{x}(k-1)$. (b) Backward prediction: sample $x(k-N-1)$ is predicted from vector $\mathbf{x}(k)$.

e.g., $\mathbf{e}_b^{(i+1)}(k)$ and $\mathbf{e}_f^{(i+1)}(k)$. However, for convenience of notation, order N predictors are indicated without the superscript $^{(N+1)}$, i.e., $\mathbf{e}_b(k) = \mathbf{e}_b^{(N+1)}(k)$ and $\mathbf{e}_f(k) = \mathbf{e}_f^{(N+1)}(k)$.

3.5.1 The backward prediction problem

In the backward prediction problem, we try to obtain an estimate of a past sample of a given input sequence using the more recent available information of this sequence. In the problem of order N at instant k, the prediction of the desired backward sample $x(k-N-1)$ is based on vector $\mathbf{x}(k)$. The weighted backward prediction error vector is defined as

$$\mathbf{e}_b(k) = \underbrace{\begin{bmatrix} x(k-N-1) \\ \lambda^{1/2}x(k-N-2) \\ \vdots \\ \lambda^{(k-N-1)/2}x(0) \\ 0_{N+1} \end{bmatrix}}_{\mathbf{d}_b(k)} - \underbrace{\begin{bmatrix} x(k) & \cdots & x(k-N) \\ \lambda^{1/2}x(k-1) & \cdots & \lambda^{1/2}x(k-N-1) \\ \vdots & \ddots & \vdots \\ \lambda^{(k-1)/2}x(1) & \cdots & 0 \\ \lambda^{k/2}x(0) & \cdots & 0 \end{bmatrix}}_{\mathbf{X}(k)} \mathbf{w}_b(k), \quad (3.73)$$

where $\mathbf{w}_b(k)$ is the backward prediction coefficient vector and $\mathbf{d}_b(k)$ the weighted backward desired signal vector.

In the backward prediction problem of order $i-1$ the weighted backward prediction error vector is

$$\mathbf{e}_b^{(i)}(k) = \mathbf{d}_b^{(i)}(k) - \mathbf{X}^{(i)}(k)\mathbf{w}_b^{(i)}(k), \quad (3.74)$$

where the weighted desired signal vector

$$\mathbf{d}_b^{(i)}(k) = \left[x(k-i) \;\; \lambda^{1/2}x(k-i-1) \;\; \cdots \;\; \lambda^{(k-i)/2}x(0) \;\; 0_i^{\mathrm{T}}\right]^{\mathrm{T}} \quad (3.75)$$

represents the $(i+1)$th column of the data matrix in (3.5) and is denoted as $\mathbf{x}^{(i)}(k)$.

By differentiating ${\mathbf{e}_b^{(i)}}^{\mathrm{T}}(k)\mathbf{e}_b^{(i)}(k)$ with respect to $\mathbf{w}_b^{(i)}(k)$, the optimum backward prediction coefficient vector is given by

$$\mathbf{w}_b^{(i)}(k) = \left[{\mathbf{X}^{(i)}}^{\mathrm{T}}(k)\mathbf{X}^{(i)}(k)\right]^{-1} {\mathbf{X}^{(i)}}^{\mathrm{T}}(k)\mathbf{d}_b^{(i)}(k). \quad (3.76)$$

From the relations used in backward prediction, it is worth noting that:

- For $i=0$, $\mathbf{e}_b^{(0)}(k) = \mathbf{d}_b^{(0)}(k) = \mathbf{x}^{(0)}(k)$. For $\{i=1,\ldots,N\}$, the data matrix can be denoted as $\mathbf{X}^{(i+1)}(k) = \left[\mathbf{X}^{(i)}(k) \;\; \mathbf{d}_b^{(i)}(k)\right]$ and (3.74) can be rewritten as

$$\mathbf{e}_b^{(i)}(k) = \mathbf{X}^{(i+1)}(k)\begin{bmatrix} -\mathbf{w}_b^{(i)}(k) \\ 1 \end{bmatrix}. \quad (3.77)$$

- The error vectors in (3.77) for $\{i = 0, \ldots, N\}$ can be collected in matrix

$$\mathbf{G}(k) = \mathbf{X}(k)\mathbf{K}^{-1}(k), \tag{3.78}$$

where

$$\mathbf{G}(k) = \left[\mathbf{e}_b^{(N)}(k)\ \mathbf{e}_b^{(N-1)}(k) \cdots \mathbf{e}_b^{(0)}(k) \right], \tag{3.79}$$

and

$$\mathbf{K}^{-1}(k) = \begin{bmatrix} -w_{b,0}^{(N)}(k) & -w_{b,0}^{(N-1)}(k) & \cdots & -w_{b,0}^{(1)}(k) & 1 \\ -w_{b,1}^{(N)}(k) & -w_{b,1}^{(N-1)}(k) & \cdots & 1 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ -w_{b,N-1}^{(N)}(k) & 1 & \cdots & 0 & 0 \\ 1 & 0 & \cdots & 0 & 0 \end{bmatrix}. \tag{3.80}$$

- The $(i+1)$th column of $\mathbf{K}^{-1}(k)$ for $i = 0, \ldots, N$, represents the coefficients of the backward prediction errors filters of order $N - i$;
- The first row of matrix $\mathbf{G}(k)$ in (3.78) corresponds to the *a posteriori* backward prediction error (with different orders and at instant k) vector transposed, i.e.,

$$\mathbf{K}^{-\mathrm{T}}(k)\mathbf{x}(k) = \begin{bmatrix} \varepsilon_b^{(N)}(k) \\ \varepsilon_b^{(N-1)}(k) \\ \vdots \\ \varepsilon_b^{(1)}(k) \\ \varepsilon_b^{(0)}(k) \end{bmatrix}. \tag{3.81}$$

- When $\mathbf{w}_b^{(i)}(k)$ fulfills (3.76) for $\{i = N, N-1, \ldots, 1, 0\}$, the product $\mathbf{D}^2(k) = \mathbf{G}^{\mathrm{T}}(k)\mathbf{G}(k)$ is a diagonal matrix whose elements $||\mathbf{e}_b^{(i)}(k)||^2$ represent the minimum backward prediction errors energy.
- If we replace first (3.76) in (3.74) and then $\mathbf{X}^{(i)}(k)$ by $\mathbf{G}^{(i)}(k)\mathbf{K}^{(i)}(k)$ (of order $i-1$), we obtain the expression

$$\mathbf{e}_b^{(i)}(k) = \mathbf{x}^{(i)}(k) - \mathbf{G}(k)\mathbf{D}^{-2}(k)\mathbf{G}^{\mathrm{T}}(k)\mathbf{x}^{(i)}(k). \tag{3.82}$$

For convenience, we rewrite the last equation as

$$\mathbf{e}_b^{(i)}(k) = \mathbf{x}^{(i)}(k) - \sum_{j=0}^{i-1} \mathbf{e}_j\ \kappa_{ji}(k), \tag{3.83}$$

with $\mathbf{e}_j = \mathbf{e}_b^{(N-j)}(k)$, $\kappa_{ji} = \mathbf{e}_j^{\mathrm{T}}\mathbf{x}^{(i)}(k) / \|\mathbf{e}_j\|^2$ and $j < i$.
- The set of error vectors for $\{i = 0, \ldots, N\}$ can be rewritten as

$$\mathbf{X}(k) = [\mathbf{e}_0\, \mathbf{e}_1 \cdots \mathbf{e}_N] \begin{bmatrix} 0 & \cdots & 0 & 1 \\ \vdots & & & \kappa_{1N} \\ 0 & & & \vdots \\ 1 & \kappa_{N1} & \cdots & \kappa_{NN} \end{bmatrix} = \mathbf{G}(k)\mathbf{K}(k). \tag{3.84}$$

- Equation (3.84) represents the Gram–Schmidt orthogonalization procedure [3] of the columns of matrix $\mathbf{X}(k)$ of special interest for the case of a lower triangular Cholesky factor.

The rotated weighted backward prediction error vector is defined below and it will be used, in the next chapter, in the derivation of the fast QR algorithms.

$$\mathbf{e}_{b_q}(k) = \mathbf{Q}(k)\mathbf{e}_b(k) = \begin{bmatrix} \mathbf{O} & \mathbf{e}_{b_{q_1}}(k) \\ \mathbf{U}(k) & \mathbf{d}_{b_{q_2}}(k) \end{bmatrix} \begin{bmatrix} -\mathbf{w}_b(k) \\ 1 \end{bmatrix} \tag{3.85}$$

By following similar procedure as the one used in the WLS estimation problem, it is possible to show that

$$\varepsilon_b(k) = \gamma(k) e_{b_{q_1}}(k) = \gamma^2(k) e_b(k). \tag{3.86}$$

3.5.2 The forward prediction problem

In the forward prediction problem, we obtain an estimate of a sample of a given input sequence using the past available information of this sequence. In the problem of order N at instant k, the prediction of the desired signal $x(k)$ is based on vector $\mathbf{x}(k-1)$ and the weighted forward prediction error vector is defined as

$$\mathbf{e}_f(k) = \underbrace{\begin{bmatrix} x(k) \\ \lambda^{1/2} x(k-1) \\ \vdots \\ \lambda^{(k-1)/2} x(1) \\ \lambda^{k/2} x(0) \end{bmatrix}}_{\mathbf{d}_f(k)} - \underbrace{\begin{bmatrix} x(k-1) & \cdots & x(k-N-1) \\ \lambda^{1/2} x(k-2) & \cdots & \lambda^{1/2} x(k-N-2) \\ \vdots & \ddots & \vdots \\ \lambda^{(k-1)/2} x(0) & \cdots & 0 \\ 0 & \cdots & 0 \end{bmatrix}}_{\begin{bmatrix} \mathbf{X}(k-1) \\ 0^{\mathrm{T}} \end{bmatrix},} \mathbf{w}_f(k) \tag{3.87}$$

where $\mathbf{w}_f(k)$ is the forward prediction coefficient vector. Note that the last row of the data matrix, which contains zeros, appears due to the fact that we assume $x(k) = 0$ for $k < 0$.

The weighted forward prediction error vector, of order $i-1$ and instant k, is given as

$$\mathbf{e}_f^{(i)}(k) = \mathbf{d}_f^{(i)}(k) - \begin{bmatrix} \mathbf{X}^{(i)}(k-1) \\ 0^{\mathrm{T}} \end{bmatrix} \mathbf{w}_f^{(i)}(k), \tag{3.88}$$

where $\mathbf{d}_f^{(i)}(k)$ is the weighted desired signal and represents the first column of the data matrix denoted in (3.5) as $\mathbf{x}^{(0)}(k)$, that is,

$$\mathbf{d}_f^{(i)}(k) = \left[x(k) \;\; \lambda^{1/2} x(k-1) \;\; \cdots \;\; \lambda^{k/2} x(0) \right]^{\mathrm{T}}. \tag{3.89}$$

By differentiating $[\mathbf{e}_f^{(i)}(k)]^{\mathrm{T}} \mathbf{e}_f^{(i)}(k)$ with respect to $\mathbf{w}_f^{(i)}(k)$ and equating the result to zero, we find the optimum forward prediction coefficient vector of order $(i-1)$, i.e.,

$$\mathbf{w}_f^{(i)}(k) = \left\{ \left[\mathbf{X}^{(i)}(k-1) \right]^{\mathrm{T}} \mathbf{X}^{(i)}(k-1) \right\}^{-1} \begin{bmatrix} \mathbf{X}^{(i)}(k-1) \\ 0^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}} \mathbf{d}_f^{(i)}(k). \tag{3.90}$$

From the relations used in forward prediction, it is relevant to note that:

- At instant $\ell = k-N$ and for $i = 0$, the weighted forward prediction error vector is $\mathbf{e}_f^{(0)}(k-N) = \mathbf{d}_f^{(0)}(k-N) = \mathbf{x}^{(0)}(k-N)$. At instant $\ell+i$ with $\{i = 1, \ldots, N\}$ the data matrix with dimension $(\ell+i+1) \times (i+1)$ can be represented as $\mathbf{X}^{(i+1)}(\ell+i) = \begin{bmatrix} \mathbf{d}_f^{(i)}(\ell+i) & \mathbf{X}^{(i)}(\ell+i-1) \\ & 0^{\mathrm{T}} \end{bmatrix}$, and (3.88) can be rewritten as

$$\mathbf{e}_f^{(i)}(\ell+i) = \mathbf{X}^{(i+1)}(\ell+i) \begin{bmatrix} 1 \\ -\mathbf{w}_f^{(i)}(\ell+i) \end{bmatrix}. \tag{3.91}$$

- The error vectors in (3.91) for $\{i = 0, \ldots, N\}$ can be collected into matrix

$$\mathbf{G}(k) = \mathbf{X}(k)\mathbf{K}^{-1}(k), \tag{3.92}$$

where

$$\mathbf{G}(k) = \left[\mathbf{e}_0 \; \mathbf{e}_1 \; \cdots \; \mathbf{e}_N \right], \tag{3.93}$$

with

$$\mathbf{e}_i = \begin{bmatrix} \mathbf{e}_f^{(i)}(k-N+i) \\ 0_{N-i} \end{bmatrix}, \tag{3.94}$$

and

$$\mathbf{K}^{-1}(k) = \begin{bmatrix} 0 & 0 & \cdots & 0 & 1 \\ 0 & 0 & \cdots & 1 & -w_{f,0}^{(N)}(k) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 1 & \cdots & -w_{f,N-3}^{(N-1)}(k-1) & -w_{f,N-2}^{(N)}(k) \\ 1 & -w_{f,0}^{(1)}(k-N+1) & \cdots & -w_{f,N-2}^{(N-1)}(k-1) & -w_{f,N-1}^{(N)}(k) \end{bmatrix}. \tag{3.95}$$

- The $(i+1)$th column of $\mathbf{K}^{-1}(k)$ for $i=0,\ldots,N$, represents the coefficients of the forward prediction error filters of order i at instant $k-N+i$.
- The first row of the system of equations in (3.92) is the *a posteriori* forward prediction error vector, with different orders, and at distinct time instants, i.e.,

$$\mathbf{K}^{-\mathrm{T}}(k)\mathbf{x}(k)=\begin{bmatrix}\varepsilon_f^{(0)}(k-N)\\ \varepsilon_f^{(1)}(k-N+1)\\ \vdots\\ \varepsilon_f^{(N)}(k)\end{bmatrix}. \tag{3.96}$$

- When $\mathbf{w}_f^{(i)}(k-N+i)$ obeys (3.90) for $\{i=N,\ldots,0\}$, the product $\mathbf{D}^2(k)=\mathbf{G}^{\mathrm{T}}(k)\mathbf{G}(k)$ is a diagonal matrix whose elements are the minimum forward prediction error energy $||\mathbf{e}_f^{(i)}(k-N+i)||^2$.
- If we replace (3.90) in (3.88) and rewrite the data matrix $\mathbf{X}^{(i)}(k-1)$ as $\mathbf{G}^{(i)}(k-1)\mathbf{K}^{(i)}(k-1)$, we obtain, at instant $\ell+i$ with $\ell=k-N$,

$$\mathbf{e}_f^{(i)}(\ell+i)=\mathbf{d}_f^{(i)}(\ell+i)-\begin{bmatrix}\sum_{j=0}^{i-1}\mathbf{e}_f^{(j)}(\ell+j-1)\ \kappa_{ij}\\ 0\end{bmatrix}, \tag{3.97}$$

where $\kappa_{ij}=\begin{bmatrix}\mathbf{e}_f^{(j)}(\ell+j-1)\\ 0\end{bmatrix}^{\mathrm{T}}\mathbf{d}_f^{(i)}(\ell+i)/||\mathbf{e}_f^{j}(\ell+j)||^2$ and $j<i$.

- The set of weighted forward prediction error vectors for $\{i=0,\ldots,N\}$ can be rewritten as

$$\mathbf{X}(k)=[\mathbf{e}_0\,\mathbf{e}_1\cdots\mathbf{e}_N]\begin{bmatrix}\kappa_{00} & \cdots & \kappa_{0\,N-1} & 1\\ \vdots & & & 0\\ & & & \\ \kappa_{N-1\,0} & & & \vdots\\ 1 & 0 & \cdots & \end{bmatrix}=\mathbf{G}(k)\mathbf{K}(k). \tag{3.98}$$

- Equation (3.98) represents the Gram–Schmidt orthogonalization procedure [3] of the columns of $\mathbf{X}(k)$ matrix of special interest for the case of an upper triangular Cholesky factor.

The rotated weighted forward prediction error vector, obtained from (3.88) with $i=N+1$, is then defined as

$$\mathbf{e}_{f_q}(k)=\begin{bmatrix}\mathbf{Q}(k-1) & 0\\ 0^{\mathrm{T}} & 1\end{bmatrix}\mathbf{e}_f(k)=\begin{bmatrix}\mathbf{e}_{f_{q_1}}(k) & \mathbf{O}\\ \mathbf{d}_{f_{q_2}}(k) & \mathbf{U}(k-1)\\ \lambda^{k/2}x(0) & 0^{\mathrm{T}}\end{bmatrix}\begin{bmatrix}1\\ -\mathbf{w}_f(k)\end{bmatrix}. \tag{3.99}$$

By following similar procedures as the one used in the WLS estimation problem, it is possible to show that

$$\varepsilon_f(k) = \gamma(k-1)e_{f_{q_1}}(k) = \gamma^2(k-1)e_f(k). \tag{3.100}$$

3.5.3 *Interpreting the elements of* $\mathbf{Q}_\theta(k)$ *for a lower triangular Cholesky factor*

This subsection provides insightful information about the variables used by the QR algorithms. To start, we observe that both representations of the input data matrix, in (3.84) and in (3.98), can be used to rewrite the autocorrelation matrix as

$$\begin{aligned}\mathbf{R}(k) = \mathbf{U}^{\mathrm{T}}(k)\mathbf{U}(k) &= \mathbf{X}^{\mathrm{T}}(k)\mathbf{X}(k)\\ &= \mathbf{K}^{\mathrm{T}}(k)\mathbf{G}^{\mathrm{T}}(k)\mathbf{G}(k)\mathbf{K}(k)\\ &= [\mathbf{D}(k)\mathbf{K}(k)]^{\mathrm{T}}[\mathbf{D}(k)\mathbf{K}(k)],\end{aligned} \tag{3.101}$$

and the projection operator as

$$\mathscr{P}(k) = \mathbf{X}(k)\mathbf{R}^{-1}(k)\mathbf{X}^{\mathrm{T}}(k) = \mathbf{Q}_2^{\mathrm{T}}(k)\mathbf{Q}_2(k) = \mathbf{G}(k)\mathbf{D}^{-2}(k)\mathbf{G}^{\mathrm{T}}(k). \tag{3.102}$$

The term $\mathbf{D}(k)\mathbf{K}(k)$ may represent the Cholesky factor of the deterministic autocorrelation matrix, defined in (3.15) as $\mathbf{U}(k)$, and the columns of matrix $\mathbf{Q}_2^{\mathrm{T}}(k) = \mathbf{G}(k)\mathbf{D}^{-1}(k)$ represent the prediction error vectors normalized by the prediction error energy.

Once $\mathbf{U}(k) = \mathbf{D}(k)\mathbf{K}(k)$, using (3.48), (3.68), (3.69) and (3.65), yields the following expression for $\mathbf{Q}_\theta(k)$:

$$\mathbf{Q}_\theta(k) = \begin{bmatrix} \gamma(k) & -\gamma(k)\lambda^{-1/2}\left[\mathbf{D}^{-1}(k-1)\boldsymbol{K}^{-\mathrm{T}}(k-1)\mathbf{x}(k)\right]^{\mathrm{T}} \\ \mathbf{D}^{-1}(k)\mathbf{K}^{-\mathrm{T}}(k)\mathbf{x}(k) & \lambda^{1/2}\mathbf{D}^{-1}(k)\mathbf{K}^{-\mathrm{T}}(k)\mathbf{K}^{\mathrm{T}}(k-1)\mathbf{D}^{\mathrm{T}}(k-1) \end{bmatrix}. \tag{3.103}$$

Equations (3.101, 3.102, 3.103) are valid for both types of triangularization. Nevertheless, from the backward prediction problem, where the product $\mathbf{D}(k)\mathbf{K}(k)$ corresponds to a lower triangular matrix, if we recall the physical interpretation of $\mathbf{K}^{-\mathrm{T}}(k)\mathbf{x}(k)$ as in (3.81) and that the elements of the diagonal matrix $\mathbf{D}(k)$ are given by $\| \mathbf{e}_b^{(N-i)}(k) \|$, it follows that

$$\mathbf{f}(k) = \mathbf{D}^{-1}(k)\mathbf{K}^{-\mathrm{T}}(k)\mathbf{x}(k) = \begin{bmatrix} \varepsilon_b^{(N)}(k)/\| \mathbf{e}_b^{(N)}(k) \| \\ \varepsilon_b^{(N-1)}(k)/\| \mathbf{e}_b^{(N-1)}(k) \| \\ \vdots \\ \varepsilon_b^{(0)}(k)/\| \mathbf{e}_b^{(0)}(k) \| \end{bmatrix}. \tag{3.104}$$

Thus, $\mathbf{f}(k)$, for the case of lower triangularization of the Cholesky factor, is the *a posteriori* backward prediction error vector at instant k normalized by backward error energies at the same instant.

Moreover, from the same interpretation of $\mathbf{D}(k-1)$ and $\mathbf{K}^{-\mathrm{T}}(k-1)$, vector $\mathbf{g}(k)$, from (3.48) and (3.65), may be given as

$$\mathbf{g}(k) = -\gamma(k)\lambda^{-1/2} \begin{bmatrix} e_b^{(N)}(k)/\parallel \mathbf{e}_b^{(N)}(k-1) \parallel \\ e_b^{(N-1)}(k)/\parallel \mathbf{e}_b^{(N-1)}(k-1) \parallel \\ \vdots \\ e_b^{(0)}(k)/\parallel \mathbf{e}_b^{(0)}(k-1) \parallel \end{bmatrix}. \tag{3.105}$$

Thus, $\mathbf{g}(k)$ is the *a priori* backward prediction error vector at instant k weighted by $-\gamma(k)\lambda^{-1/2}$ and normalized by backward error energies at instant $k-1$.

3.5.4 Interpreting the elements of $\mathbf{Q}_\theta(k)$ for an upper triangular Cholesky factor

For the case of an upper triangular Cholesky factor, the product $\mathbf{D}(k)\mathbf{K}(k)$ comes from the forward prediction orthogonalization procedure. It is also worth mentioning that (3.95) brings an interpretation of the non-zero elements of the rows of $\mathbf{K}^{-\mathrm{T}}(k)$ as the coefficients of forward prediction filters of different orders at distinct time instants. Recalling the physical interpretation of $\mathbf{K}^{-\mathrm{T}}(k)\mathbf{x}(k)$ as (3.96) and that the elements of the diagonal matrix $\mathbf{D}(k)$ are also given by $\parallel \mathbf{e}_f^{(i)}(k-N+i) \parallel$, it follows that

$$\mathbf{f}(k) = \begin{bmatrix} \varepsilon_f^{(0)}(k-N)/\parallel \mathbf{e}_f^{(0)}(k-N) \parallel \\ \varepsilon_f^{(1)}(k-N+1)/\parallel \mathbf{e}_f^{(1)}(k-N+1) \parallel \\ \vdots \\ \varepsilon_f^{(N)}(k)/\parallel \mathbf{e}_f^{(N)}(k) \parallel \end{bmatrix}. \tag{3.106}$$

In this case, $\mathbf{f}(k)$ is the *a posteriori* forward prediction error vector normalized by forward error energies at different time instants.

By using the same interpretation of $\mathbf{D}(k-1)$ and $\mathbf{K}^{-\mathrm{T}}(k-1)$, vector $\mathbf{g}(k)$ corresponds, in the upper triangularization case, to

$$\mathbf{g}(k) = -\gamma(k)\lambda^{-1/2} \begin{bmatrix} e_f^{(0)}(k-N)/\parallel \mathbf{e}_f^{(0)}(k-N-1) \parallel \\ e_f^{(1)}(k-N+1)/\parallel \mathbf{e}_f^{(1)}(k-N) \parallel \\ \vdots \\ e_f^{(N)}(k)/\parallel \mathbf{e}_f^{(N)}(k-1) \parallel \end{bmatrix}. \tag{3.107}$$

Vector $\mathbf{g}(k)$ is then an *a priori* forward prediction error vector normalized by the *a posteriori* forward error energies and weighted by $-\gamma(k)\lambda^{-1/2}$.

We note that, in the case of upper triangularization, the normalized errors present in $\mathbf{Q}_\theta(k)$ are of different orders at distinct instants of time (order and time updating); this fact seems to be the cause of the extra computational effort of the fast—or $\mathcal{O}[N]$ (order of N)—algorithms derived from this type of triangularization.

With the physical interpretation of vectors $\mathbf{g}(k)$ and $\mathbf{f}(k)$ presented above and taking into account Equations (3.61) and (3.62), matrix $\mathbf{E}(k)$ can be interpreted as a *conversion factor matrix* between the *a priori* and the *a posteriori* prediction error vectors.

3.6 The Inverse QRD-RLS Algorithm

An alternative approach to the conventional QRD-RLS algorithm based on the update of the inverse Cholesky factor was presented in [11]. This algorithm, known as the Inverse QR decomposition (IQRD-RLS) algorithm, allows the calculation of the weight vector without back-substitution. In the following, based on the structure of $\mathbf{Q}_\theta(k)$ and on the relations (3.62), (3.65), (3.68), and (3.69), we present the IQRD-RLS algorithm.

Starting from the RLS solution $\mathbf{w}(k) = \mathbf{R}^{-1}(k)\mathbf{p}(k)$ with $\mathbf{R}(k)$ and $\mathbf{p}(k)$ defined as in (3.9) and in (3.8), respectively, and using the expression in (3.38) instead of $\mathbf{X}(k)$, after some manipulations, we can show that

$$\mathbf{w}(k) = \mathbf{w}(k-1) + e(k)\mathbf{U}^{-1}(k)\mathbf{U}^{-\mathrm{T}}(k)\mathbf{x}(k), \tag{3.108}$$

where $e(k) = d(k) - \mathbf{x}^{\mathrm{T}}(k)\mathbf{w}(k-1)$ is the *a priori* error and the term multiplying this variable is known as the *Kalman Gain*. Also note, knowing that $\mathbf{R}(k) = \mathbf{U}^{\mathrm{T}}(k)\mathbf{U}(k)$, that (3.108) corresponds to (2.55), from the previous chapter.

Since we know that $\mathbf{Q}_\theta(k)$ is unitary, if we post-multiply this matrix by its first row transposed, it follows that

$$\mathbf{Q}_\theta(k)\begin{bmatrix}\gamma(k)\\ \mathbf{g}(k)\end{bmatrix} = \begin{bmatrix}1\\ 0\end{bmatrix}. \tag{3.109}$$

From (3.65), we have $\mathbf{g}(k) = -\gamma(k)\lambda^{-1/2}\mathbf{U}^{-\mathrm{T}}(k-1)\mathbf{x}(k)$. For convenience, we define

$$\mathbf{a}(k) = -\gamma^{-1}(k)\mathbf{g}(k) = \lambda^{-1/2}\mathbf{U}^{-\mathrm{T}}(k-1)\mathbf{x}(k) \tag{3.110}$$

and rewrite (3.109) as

$$\mathbf{Q}_\theta(k)\begin{bmatrix}1\\ -\mathbf{a}(k)\end{bmatrix} = \begin{bmatrix}\gamma^{-1}(k)\\ 0\end{bmatrix}. \tag{3.111}$$

This expression, if we know $\mathbf{a}(k)$, provides $\mathbf{Q}_\theta(k)$. At this point, it is relevant to observe that (3.69), rewritten as $\lambda^{-1/2}\mathbf{E}(k)\mathbf{U}^{-\mathrm{T}}(k-1) = U^{-\mathrm{T}}(k)$, suggests that $\mathbf{U}^{-\mathrm{T}}(k-1)$ can be updated with the same matrix that updates $\mathbf{U}(k-1)$. In fact, if we rotate $\begin{bmatrix} 0 & \lambda^{-1/2}\mathbf{U}^{-1}(k-1) \end{bmatrix}^{\mathrm{T}}$ with $\mathbf{Q}_\theta(k)$, we obtain

$$\begin{bmatrix} \gamma(k) & \mathbf{g}^{\mathrm{T}}(k) \\ \mathbf{f}(k) & \mathbf{E}(k) \end{bmatrix} \begin{bmatrix} 0^{\mathrm{T}} \\ \lambda^{-1/2}\mathbf{U}^{-\mathrm{T}}(k-1) \end{bmatrix} = \begin{bmatrix} \lambda^{-1/2}\mathbf{g}^{\mathrm{T}}(k)\mathbf{U}^{-\mathrm{T}}(k-1) \\ \mathbf{U}^{-\mathrm{T}}(k) \end{bmatrix}. \tag{3.112}$$

For convenience, we define $\mathbf{u}(k) = \lambda^{-1/2}\mathbf{U}^{-1}(k-1)\mathbf{g}(k)$. Using vector $\mathbf{a}(k)$ from (3.110), this vector can be expressed as

$$\mathbf{u}(k) = -\lambda^{-1/2}\gamma(k)\mathbf{U}^{-1}(k-1)\mathbf{a}(k), \tag{3.113}$$

or, using (3.62), (3.68), and (3.69), as

$$\mathbf{u}(k) = -\gamma^{-1}(k)\mathbf{U}^{-1}(k)\mathbf{U}^{-\mathrm{T}}(k)\mathbf{x}(k). \tag{3.114}$$

Finally, with this last equation, (3.108) can be rewritten as

$$\mathbf{w}(k) = \mathbf{w}(k-1) - e(k)\gamma(k)\mathbf{u}(k), \tag{3.115}$$

where the Kalman vector is now expressed as $-\gamma(k)\mathbf{u}(k)$.

By combining (3.112) and (3.111) in one single equation, we have

$$\begin{bmatrix} 1/\gamma(k) & \mathbf{u}^{\mathrm{T}}(k) \\ 0 & \mathbf{U}^{-\mathrm{T}}(k) \end{bmatrix} = \mathbf{Q}_\theta(k) \begin{bmatrix} 1 & 0^{\mathrm{T}} \\ -\mathbf{a}(k) & \lambda^{-1/2}\mathbf{U}^{-\mathrm{T}}(k-1) \end{bmatrix}. \tag{3.116}$$

Equation (3.116) is a key relation to the inverse QRD-RLS algorithm, which equations are presented in Table 3.4. A pseudo-code of the (lower-triangularization version) inverse QRD-RLS algorithm is available in Appendix 3 (Table 3.6).

To close this section, we note that Equation (3.45) can be added to (3.116), i.e,

$$\begin{bmatrix} \gamma^{-1}(k) & \mathbf{u}^{\mathrm{T}}(k) & e_{q_1}(k) \\ 0 & \mathbf{U}^{-\mathrm{T}}(k) & \mathbf{d}_{q_2}(k) \end{bmatrix} = \mathbf{Q}_\theta(k) \begin{bmatrix} 1 & 0^{\mathrm{T}} & d(k) \\ -\mathbf{a}(k) & \lambda^{-1/2}\mathbf{U}^{-\mathrm{T}}(k-1) & \lambda^{1/2}\mathbf{d}_{q_2}(k-1) \end{bmatrix} \tag{3.117}$$

The resulting set of equations is known as the extended QRD-RLS algorithm. It was presented in [12], before the inverse QRD-RLS algorithm [11].

3.7 Conclusion

This chapter presented concepts and derivations of the basic algorithms belonging to the QRD-RLS family: the conventional and the inverse QRD-RLS algorithms. We started by noting that the QR decomposition, when applied to solve the LS

Table 3.4 The inverse QRD-RLS equations.

IQRD-RLS
for each k
{ Obtaining $\mathbf{a}(k)$:
$\mathbf{a}(k) = \mathbf{U}^{-\mathrm{H}}(k-1)\mathbf{x}(k)/\sqrt{\lambda}$
Obtaining $\mathbf{Q}_\theta(k)$ and $\gamma(k)$:
$\begin{bmatrix} 1/\gamma(k) \\ 0 \end{bmatrix} = \mathbf{Q}_\theta(k) \begin{bmatrix} 1 \\ -\mathbf{a}(k) \end{bmatrix}$
Obtaining $\mathbf{u}(k)$ and updating $\mathbf{U}^{-\mathrm{H}}(k)$:
$\begin{bmatrix} \mathbf{u}^{\mathrm{H}}(k) \\ \mathbf{U}^{-\mathrm{H}}(k) \end{bmatrix} = \mathbf{Q}_\theta(k) \begin{bmatrix} 0^{\mathrm{T}} \\ \lambda^{-1/2}\mathbf{U}^{-\mathrm{H}}(k-1) \end{bmatrix}$
Obtaining the *a priori* error $e(k)$:
$e(k) = d(k) - \mathbf{w}^{\mathrm{H}}(k-1)\mathbf{x}(k)$
Updating the coefficient vector:
$\mathbf{w}(k) = \mathbf{w}(k-1) - \gamma(k)e^*(k)\mathbf{u}(k)$
}

problem, comprises the rotation of the space spanned by the columns of the input data matrix. The relationship between the conventional LS and the QRD-LS methods was established and the orthogonality principle was shown in the rotate signal domain.

Using the recursive structure of the data matrix, the rotated RLS solution of (3.45) follows. It is worth mentioning that, if we apply the transformation

$$\begin{bmatrix} \gamma(k) & 0^{\mathrm{T}} \\ 0 & \mathbf{U}(k) \end{bmatrix} \tag{3.118}$$

in the filtering and adaptation operations of the RLS algorithm, i.e.,

$$\begin{bmatrix} e(k) \\ \mathbf{w}(k) \end{bmatrix} = \begin{bmatrix} 1 & -\mathbf{x}^{\mathrm{T}}(k) \\ \mathbf{R}^{-1}(k)\mathbf{x}(k) & \mathbf{I} - \mathbf{R}^{-1}(k)\mathbf{x}(k)\mathbf{x}^{\mathrm{T}}(k) \end{bmatrix} \begin{bmatrix} d(k) \\ \mathbf{w}(k-1) \end{bmatrix}, \tag{3.119}$$

after some algebra, the same conventional QRD-RLS algorithm of (3.45) follows, that is,

$$\begin{bmatrix} e_{q_1}(k) \\ \mathbf{d}_{q_2}(k) \end{bmatrix} = \mathbf{Q}_\theta(k) \begin{bmatrix} d(k) \\ \lambda^{1/2}\mathbf{d}_{q_2}(k-1) \end{bmatrix}. \tag{3.120}$$

In fact, with the change of coordinates, the system described in (3.119) can be transformed into another system, with different internal descriptions, but with the same input–output behavior for solving the LS problem. Although theoretically equivalent, the resulting system may have different numerical behavior when implemented in finite precision arithmetics [13]. Concerning specially the transformed system in (3.120), it is possible to find a numerically robust algorithm, e.g., those in [8, 14].

In Section 3.5, the structure of matrix $\mathbf{Q}_\theta(k)$ and the physical interpretation of its internal variables were presented. From the known structure of matrix $\mathbf{Q}_\theta(k)$, the IQRD-RLS algorithm was easily obtained. This algorithm, besides the inherited numerical robustness of its family, provides the coefficient vector at every iteration, without having to resort to the computationally onerous backward or forward substitution procedures. With the interpretation of the internal variables of $\mathbf{Q}_\theta(k)$, expressed in terms of backward and forward prediction errors, we have also provided all necessary tools to help readers understand the forthcoming chapters of this book.

It is worth mentioning that the QRD-RLS algorithms addressed in this chapter present a computational complexity of $\mathcal{O}[N^2]$ and preserve the desirable fast convergence property of the conventional RLS algorithm. Hence, assuming an infinite-precision environment, the outcomes of both algorithms (the RLS and the QRD-RLS, conventional or inverse), once initialized in an equivalent form, are identical in terms of speed of convergence and quality of estimation. Since the solution obtained by the QRD-RLS algorithm corresponds to the solution of the RLS algorithm from a transformed domain, the good numerical behavior can be presumed as a direct consequence of this transformation.

Appendix 1 - Forward and Back-Substitution Procedures

In order to show two possible procedures used to obtaining the coefficient vector with one order of magnitude less computational complexity than matrix inversion [15], consider the following system of equations:

$$\mathbf{U}\mathbf{x} = \mathbf{y}, \tag{3.121}$$

where $\mathbf{U}$ is a $(N+1)\times(N+1)$ triangular matrix, as illustrated in Figure 3.1, and $\mathbf{x}$ is the $(N+1)\times 1$ vector we need to obtain.

When $\mathbf{U}$ is upper triangular, we use the ***forward substitution procedure***:

$$\begin{aligned} x_1 &= \frac{y_{N+1}}{U_{N+1,1}} \\ x_i &= \frac{1}{U_{\ell,i}}\left(y_\ell - \sum_{j=1}^{i-1} U_{\ell,j}x_j\right) \end{aligned} \tag{3.122}$$

for $i = 2,3,\ldots,N+1$ and $\ell = N-i+2$.

When $\mathbf{U}$ is lower triangular, we use the ***back-substitution procedure***:

$$\begin{aligned} x_{N+1} &= \frac{y_1}{U_{1,N+1}} \\ x_i &= \frac{1}{U_{\ell,i}}\left(y_\ell - \sum_{j=i+1}^{N+1} U_{\ell,j}x_j\right) \end{aligned} \tag{3.123}$$

for $i = N,N-1,\ldots,1$ and $\ell = N-i+2$.

Appendix 2 - Square-Root-Free QRD-RLS Algorithms

The two-dimensional vector rotations, necessary to execute the QRD-RLS algorithm, can be efficiently implemented using a CORDIC (COrdinate Rotation DIgital Computer) processor [16, 17]. In systems based on general-purpose programmable DSPs, vector rotations requires a SQuare-RooT (SQRT) operation, which may eventually represent a bottleneck due to its significant computational burden. To circumvent this problem, many authors have proposed square-root-free methods for performing Givens rotations. Gentleman, in his pioneer work [18], shows how to perform the plane rotations without SQRT. After that, different versions of Givens rotations without SQRT were introduced (see, e.g. [19–23]).

Among different implementations of rotations without SQRT, for the sake of simplicity, we address the version introduced in [19]. We thus start with the following rotation.

$$\begin{bmatrix} \cos\theta_1 & -\sin\theta_1 \\ \sin\theta_1 & \cos\theta_1 \end{bmatrix} \begin{bmatrix} x_1 & x_2 \\ u_1 & u_2 \end{bmatrix} = \begin{bmatrix} 0 & x_2' \\ u_1' & u_2' \end{bmatrix} \tag{3.124}$$

In order to have the first element of the first column nulled, the rotation angle is such that $\cos\theta_1 = u_1/u_1'$, $\sin\theta_1 = x_1/u_1'$ and $u_1' = \sqrt{x_1^2 + u_1^2}$, which requires a SQRT operation.

The main "trick" behind this class of algorithm is to scale the rows as follows.

$$\begin{aligned} x_i &= \delta_1^{1/2}\, \bar{x}_i \\ u_i &= d_1^{1/2}\, \bar{u}_i \\ u_i' &= d_2^{1/2}\, \bar{u}_i' \end{aligned} \tag{3.125}$$

for $i = 1, 2$ and, also, $x_2' = \delta_2^{1/2}\, \bar{x}_2'$. In terms of the new quantities, the Givens rotation in (3.124) can be rewritten as

$$\begin{bmatrix} \cos\theta_1 & -\sin\theta_1 \\ \sin\theta_1 & \cos\theta_1 \end{bmatrix} \begin{bmatrix} \delta_1^{1/2} & 0 \\ 0 & d_1^{1/2} \end{bmatrix} \begin{bmatrix} \bar{x}_1 & \bar{x}_2 \\ \bar{u}_1 & \bar{u}_2 \end{bmatrix} = \begin{bmatrix} \delta_2^{1/2} & 0 \\ 0 & d_2^{1/2} \end{bmatrix} \begin{bmatrix} 0 & \bar{x}_2' \\ \bar{u}_1' & \bar{u}_2' \end{bmatrix}. \tag{3.126}$$

By rearranging the transformations, we obtain

$$\bar{G} \begin{bmatrix} \bar{x}_1 & \bar{x}_2 \\ \bar{u}_1 & \bar{u}_2 \end{bmatrix} = \begin{bmatrix} 0 & \bar{x}_2' \\ \bar{u}_1' & \bar{u}_2' \end{bmatrix}, \tag{3.127}$$

where

$$\bar{G} = \begin{bmatrix} \cos\theta_1 \sqrt{\delta_1/\delta_2} & -\sin\theta_1 \sqrt{d_1/\delta_2} \\ \sin\theta_1 \sqrt{\delta_1/d_2} & \cos\theta_1 \sqrt{d_1/d_2} \end{bmatrix}. \tag{3.128}$$

We now rewrite $\cos\theta_1$ in terms of the new quantities:

$$\cos\theta_1 = \frac{u_1}{u_1'} = \frac{d_1^{1/2}\, \bar{u}_1}{d_2^{1/2}\, \bar{u}_1'}. \tag{3.129}$$

To find an adequate scale factor, we set

$$\cos\theta_1 = \sqrt{d_2/d_1} = \sqrt{\delta_2/\delta_1}. \tag{3.130}$$

From (3.129) and (3.130), it follows that $d_2\bar{u}'_1 = d_1\bar{u}_1$. Therefore

$$\sin\theta_1 = \frac{x_1}{u'_1} = \frac{\delta_1^{1/2}\bar{x}_1}{d_2^{1/2}\bar{u}'_1} = \frac{d_2^{1/2}\delta_1^{1/2}\bar{x}_1}{d_1\bar{u}_1}, \tag{3.131}$$

$$\sin\theta_1\sqrt{\delta_1/d_2} = \frac{d_2^{1/2}\delta_1^{1/2}\bar{x}_1}{d_1\bar{u}_1}\frac{\delta_1^{1/2}}{d_2^{1/2}} = \frac{\delta_1\bar{x}_1}{d_1\bar{u}_1}, \text{ and} \tag{3.132}$$

$$\sin\theta_1\sqrt{d_1/\delta_2} = \frac{d_2^{1/2}\delta_1^{1/2}\bar{x}_1}{d_1\bar{u}_1}\frac{d_1{}^{1/2}}{\delta_2^{1/2}} = \frac{\bar{x}_1}{\bar{u}_1}\frac{d_2^{1/2}}{d_1^{1/2}}\frac{\delta_1^{1/2}}{\delta_2^{1/2}} = \frac{\bar{x}_1}{\bar{u}_1}. \tag{3.133}$$

Thus, we can write (3.128) as

$$\bar{G} = \begin{bmatrix} 1 & -\frac{\bar{x}_1}{\bar{u}_1} \\ \frac{\delta_2\bar{x}_1}{d_2\bar{u}_1} & 1 \end{bmatrix}. \tag{3.134}$$

From (3.130), we have

$$d_2 = d_1\cos^2\theta_1, \text{ and} \tag{3.135}$$

$$\delta_2 = \delta_1\cos^2\theta_1. \tag{3.136}$$

From (3.130) and (3.131), it appears that

$$\frac{\sin^2\theta_1}{\cos^2\theta_1} = \frac{\delta_1\bar{x}_1^2}{d_1\bar{u}_1^2}, \tag{3.137}$$

such that $\cos^2\theta_1$ can be obtained with

$$\cos^2\theta_1 = (1+\sin^2\theta_1/\cos^2\theta_1)^{-1} = (1+\frac{\delta_1\bar{x}_1^2}{d_1\bar{u}_1^2})^{-1}. \tag{3.138}$$

As a result, the update formula in (3.127) with $\bar{G}$ as in (3.134) and their elements computed with (3.138, 3.135, 3.136) shall avoid the use of SQRT.

Appendix 3 - Pseudo-Codes

In the following, we present the pseudo-codes for the conventional and the inverse QRD-RLS algorithms. In both cases, we present their complex versions employing the Cholesky vector with lower triangular matrices.

Table 3.5 Pseudo-code for the conventional QRD-RLS algorithm.

QRD-RLS

% Initialization:
N (filter order), δ (small constant), λ (forgetting factor)
$\mathbf{U}(k-1) = \delta \mathbf{J}_{N+1}$; ($\mathbf{J}_{N+1}$ being the reversal matrix)
$\mathbf{w}(k) = 0_{(N+1)\times 1}$; (if necessary) $\mathbf{d}_{q2}(k-1) = 0_{(N+1)\times 1}$;
for $k = 1, 2, \ldots$
{ xaux $= \mathbf{x}^{\mathrm{H}}(k)$; % xaux$(n) = x^*(k-n-1)$ for $n = 1 : N+1$
Uaux $= \lambda^{1/2}\mathbf{U}(k-1)$; % Uaux(n,m) = $\lambda^{1/2}[\mathbf{U}(k-1)]_{n,m}$ for $n, m = 1 : N+1$
daux $= d^*(k)$;
dq2aux $= \lambda^{1/2}\mathbf{d}_{q2}(k-1)$;
gamma $= 1$;
for $n = 1 : N+1$
{ % Obtaining $\mathbf{Q}_\theta(k)$ and updating $\mathbf{U}(k)$:
$cos\theta_{n-1}(k) = \dfrac{|\mathrm{Uaux}(N+2-n,n)|}{\sqrt{|\mathrm{xaux}(n)|^2 + |\mathrm{Uaux}(N+2-n,n)|^2}}$;
$sin\theta_{n-1}(k) = \left(\dfrac{\mathrm{xaux}(n)}{\mathrm{Uaux}(N+2-n,n)}\right)^* cos\theta_{n-1}(k)$;
xaux$(n) = 0$;
Uaux$(N+2-n,n) = sin\theta_{n-1}(k)$xaux$(n) + cos\theta_{n-1}(k)$Uaux$(N+2-n,n)$;
for $m = n+1 : N+1$
{ oldxaux = xaux(m);
xaux$(m) = cos\theta_{n-1}(k)$oldxaux $- sin^*\theta_{n-1}(k)$Uaux$(N+2-n,m)$;
Uaux$(N+2-n,m) = sin\theta_{n-1}(k)$oldxaux $+ cos\theta_{n-1}(k)$Uaux$(N+2-n,m)$;
}
% Obtaining $\gamma(k)$:
gamma $=$ gamma $cos\theta_{n-1}(k)$;
% Obtaining $eq_1(k)$ and updating $\mathbf{d}_{q2}(k)$:
olddaux = daux;
daux $= cos\theta_{n-1}(k)$.olddaux $- sin^*\theta_{n-1}(k)$dq2aux$(N+2-n)$;
dq2aux$(N+2-n) = sin\theta_{n-1}(k)$olddaux $+ cos\theta_{n-1}(k)$dq2aux$(N+2-n)$;
% Back-substitution, if necessary:
summa$= 0$;
for $m = 1 : (n-1)$
{ summa $=$ summa+Uaux$(n, N+2-m)[\mathbf{w}(k)]_{N+2-n}$;
}
$[\mathbf{w}(k)]_{N+2-n}$ = (dq2aux(n)-summa)/Uaux$(n, N+2-n)$;
}
$\mathbf{U}(k) =$ Uaux;
$\gamma(k) =$ gamma;
$\mathbf{d}_{q2}(k) =$ dq2aux;
$e_{q1}(k) =$ daux;
% Obtaining the estimation errors:
$\varepsilon(k) = e_{q1}^*(k)\gamma(k)$; % *a posteriori* error, i.e, $d(k) - \mathbf{w}^{\mathrm{H}}(k)\mathbf{x}(k)$
$e(k) = e_{q1}^*(k)/\gamma(k)$; % *a priori* error, i.e, $d(k) - \mathbf{w}^{\mathrm{H}}(k-1)\mathbf{x}(k)$
}

Table 3.6 Pseudo-code for the inverse QRD-RLS algorithm.

IQRD-RLS

% Initialization:
  N (filter order), δ (small constant), λ (forgetting factor)
  $\mathbf{U}^{\mathrm{H}}(k-1) = \frac{1}{\delta}\mathbf{J}_{N+1}$; ($\mathbf{J}_{N+1}$ being the reversal matrix)
  $\mathbf{w}(k) = 0_{(N+1)\times 1}$; $\mathbf{d}_{q2}(k-1) = 0_{(N+1)\times 1}$;
for $k = 1, 2, \ldots$
{ % Obtaining $\mathbf{a}(k)$:
  akaux $= 0_{(N+1)\times 1}$;
  xaux $= \lambda^{-1/2}\mathbf{x}(k)$;
  for $n = 1 : N+1$
    for $m = 1 : (N+2-n)$
    { akaux(n) = akaux(n)+ $[\mathbf{U}^{-\mathrm{H}}(k-1)]_{n,m}$xaux($m$);
    }
  }
  $\mathbf{a}(k) =$ akaux;
  % Obtaining $\mathbf{Q}_{\theta}(k)$ and $\gamma(k)$:
  igamma $= 1$;
  for $n = 1 : N+1$
    aux1 $= \sqrt{|\text{igamma}|^2 + |\text{akaux}(N+2-n)|^2}$;
    $cos\theta_{n-1}(k) = \dfrac{|\text{igamma}|}{\text{aux1}}$;
    $sin\theta_{n-1}(k) = \dfrac{\text{akaux}(N+2-n)}{\text{igamma}}\, cos\theta_{n-1}(k)$;
    igamma = aux1; % or $cos\theta_{n-1}(k)$igamma $+ sin^{*}\theta_{n-1}(k)$akaux$(N+2-m)$;
  }
  $\gamma(k) = 1/$igamma;
  % Obtaining $\mathbf{u}(k)$ and updating $\mathbf{U}^{-\mathrm{H}}(k)$:
  uHaux $= 0_{(N+1)\times 1}$;
  UmHaux $= \lambda^{-1/2}\mathbf{U}^{-\mathrm{H}}(k-1)$;
  for $n = 1 : N+1$
    for $m = 1 : n$
    { aux2 = uHaux(m);
      uHaux(m) $= cos\theta_{n-1}(k)$aux2 $- sin^{*}\theta_{n-1}(k)$UmHaux$(N+2-n, m)$;
      UmHaux$(N+2-n, m) = sin\theta_{n-1}(k)$aux2 $+ \cdots$
                                $\cdots + cos\theta_{n-1}(k)$UmHaux$(N+2-n, m)$;
    }
  }
  $\mathbf{u}(k) =$ uHaux*;
  $\mathbf{U}^{-\mathrm{H}}(k) =$ UmHaux;
  % Obtaining $e(k)$:
  $e(k) = d(k) - \mathbf{w}^{\mathrm{H}}(k-1)\mathbf{x}(k)$; % *a priori* error
  % Updating the coefficient vector:
  $\mathbf{w}(k) = \mathbf{w}(k-1) - \gamma(k)e^{*}(k)\mathbf{u}(k)$;
}

In order to provide a better understanding of a Givens rotation matrix, for the case of a complex vector, we give a simple example: consider vector $\mathbf{z} = [a\ b]^{\mathrm{T}}$ where a and b are complex numbers and we want to rotate this vector by pre-multiplying matrix $\mathbf{Q}_\theta$ such that the resulting vector has the same norm but one element was annihilated. This can be carried out as follows.

1. $\mathbf{Q}_\theta \mathbf{z} = \begin{bmatrix} \cos\theta & -\sin^*\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \alpha \\ 0 \end{bmatrix}$

   For this case, the values of the cosine and the sine of θ are given by

   $$\begin{cases} \cos\theta = \frac{|a|}{\sqrt{|a|^2+|b|^2}}, \text{ and} \\ \sin\theta = \left(-\frac{b}{a}\right)^* cos\theta. \end{cases}$$

   This definition was used in the conventional QRD-RLS algorithm in order to obtain $\mathbf{Q}_\theta$.

2. $\mathbf{Q}_\theta \mathbf{z} = \begin{bmatrix} \cos\theta & -\sin^*\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} 0 \\ \alpha \end{bmatrix}$

   For this second case, the values of the cosine and the sine of θ are given by

   $$\begin{cases} \cos\theta = \frac{|b|}{\sqrt{|a|^2+|b|^2}}, \text{ and} \\ \sin\theta = \left(-\frac{a}{b}\right)^* cos\theta. \end{cases}$$

   These expressions were used in inverse QRD-RLS algorithm to obtain $\mathbf{Q}_\theta$.

In both cases, α has the same norm of $\mathbf{z}$ and may be expressed as $\pm e^{j\phi}\sqrt{|a|^2+|b|^2}$ where ϕ is the phase of a (first case) or b (second case). $\mathbf{Q}_\theta$ can also be chosen slightly different in order to compensate this phase and produce a real number, the norm of z, instead of $\pm e^{j\phi}||\mathbf{z}||$.

A pseudo-code for the conventional QRD-RLS algorithm is presented in Table 3.5 while a pseudo-code for the inverse QRD-RLS algorithm is presented in Table 3.6, both employing lower triangular Cholesky factors.

References

1. S. Haykin, Adaptive Filter Theory. 4th edition Prentice-Hall, Englewood Cliffs, NJ, USA (2002)
2. P. S. R. Diniz, Adaptive Filtering: Algorithms and Practical Implementation. 3rd edition Springer, New York, NY, USA (2008)
3. G. H. Golub and C. F. Van Loan, Matrix Computations. 2nd edition John Hopkins University Press, Baltimore, MD, USA (1989)
4. R. A. Horn and C. R. Johnson, Matrix Analysis. Cambridge University Press, New York, NY, USA (1986)

5. W. Givens, Computation of plane unitary rotations transforming a general matrix to triangular form. Journal of the Society for Industrial and Applied Mathematics, vol. 6, no. 1, pp. 26–50 (March 1958)
6. W. H. Gentleman and H. T. Kung, Matrix triangularization by systolic arrays. SPIE Real-Time Signal Processing IV, vol. 298, pp. 19–26 (January 1981)
7. J. G. McWhirter, Recursive least-squares minimization using a systolic array. SPIE Real-Time Signal Processing VI, vol. 431, pp. 105–112 (January 1983)
8. P. A. Regalia and M. G. Bellanger, On the duality between fast QR methods and lattice methods in least squares adaptive filtering. IEEE Transactions on Signal Processing, vol. 39, no. 4, pp. 879–891 (April 1991)
9. N. E. Hubing and S. T. Alexander, Statistical analysis of initialization methods for RLS adaptive filters. IEEE Transactions on Signal Processing, vol. 39, no. 8, pp. 1793–1804 (August 1991)
10. P. S. R. Diniz and M. G. Siqueira, Fixed-point error analysis of the QR-recursive least square algorithm. IEEE Transactions on Circuits and Systems–II: Analog and Digital Signal Processing, vol. 42, no. 5, pp. 334–348 (May 1995)
11. S. T. Alexander and A. L. Ghirnikar, A method for recursive least squares filtering based upon an inverse QR decomposition. IEEE Transactions on Signal Processing, vol. 41, no. 1, pp. 20–30 (January 1993)
12. J. E. Hudson and T. J. Shepherd, Parallel weight extraction by a systolic least squares algorithm. SPIE Advanced Algorithms and Architectures for Signal Processing IV, vol. 1152, pp. 68–77 (August 1989)
13. S. Ljung and L. Ljung, Error propagation properties of recursive least-squares adaptation algorithms. Automatica, vol. 21, no. 2, pp. 157–167 (March 1985)
14. M. D. Miranda and M. Gerken, Hybrid least squares QR-lattice algorithm using a priori errors. IEEE Transactions on Signal Processing, vol. 45, no. 12, pp. 2900–2911 (December 1997)
15. G. Strang, Computational Science and Engineering. Wellesly-Cambridge Press, Wellesley, MA, USA (2007)
16. B. Haller, J. Götze and J. R. Cavallaro, Efficient implementation of rotation operations for high performance QRD-RLS filtering. IEEE International Conference on Application-Specific Systems, Architectures and Processors, ASAP'97, Zurich, Switzerland, pp. 162–174 (July 1997)
17. J. E. Volder, The CORDIC Trigonometric Computing Technique. IRE Transactions on Electronic Computers, vol. EC-8, no. 3, pp. 330–334 (September 1959)
18. W. M. Gentleman, Least squares computations by Givens transformations without square roots. IMA Journal of Applied Mathematics, vol. 12, no. 3, pp. 329–336 (December 1973)
19. S. Hammarling, A note on modifications to the Givens plane rotation. IMA Journal of Applied Mathematics vol. 13, no. 2, pp. 215–218 (April 1974)
20. J. L. Barlow and I. C. F. Ipsen, Scaled Givens rotations for the solution of linear least squares problems on systolic arrays. SIAM Journal on Scientific and Statistical Computing, vol. 8, no. 5, pp. 716–733 (September 1987)
21. J. Götze and U. Schwiegelshohn, A square root and division free Givens rotation for solving least squares problems on systolic arrays. SIAM Journal on Scientific and Statistical Computing, vol. 12, no. 4, pp. 800–807 (July 1991)
22. E. N. Frantzeskakis and K. J. R. Liu, A class of square root and division free algorithms and architectures for QRD-based adaptive signal processing. IEEE Transactions on Signal Processing, vol. 42, no. 9, pp. 2455–2469 (September 1994)
23. S. F. Hsieh, K. J. R. Liu, and K. Yao, A unified square-root-free approach for QRD-based recursive-least-squares estimation. IEEE Transactions on Signal Processing, vol. 41, no. 3, pp. 1405–1409 (March 1993)