Chapter 4
Fast QRD-RLS Algorithms

José A. Apolinário Jr. and Paulo S. R. Diniz

Abstract Although numerically robust, the QR-decomposition recursive least-squares (QRD-RLS) algorithms studied in the previous chapter are computationally intensive, requiring a number of mathematical operations in the order of N^2, or $\mathcal{O}[N^2]$, N being the order of the adaptive filter. This chapter describes the so-called *fast* QRD-RLS algorithms, i.e., those computationally efficient algorithms that, besides keeping the attractive numerical robustness of the family, benefit from the fact that the input signal is a delay line, reducing the complexity to $\mathcal{O}[N]$. The fast versions of the QRD-RLS algorithms using real variables are classified and derived. For each algorithm, we present the final set of equations as well as their pseudo-codes in tables. For the main algorithms, their descriptions are given utilizing complex variables.

4.1 Introduction

Usually the choice of a given adaptive filtering algorithm for an application relies on a number of properties such as speed of convergence, steady-state behavior in stationary environments, and tracking capability in non-stationary environments. However, very often the algorithm choice is highly correlated to its computational complexity. In the case of the recursive least-squares(RLS) family of algorithms, their distinctive features are behavior in finite wordlength implementations and computational burden.

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

Paulo S. R. Diniz
Federal University of Rio de Janeiro (UFRJ), Rio de Janeiro – Brazil
e-mail: diniz@lps.ufrj.br

J.A. Apolinário Jr. (ed.), *QRD-RLS Adaptive Filtering*,
DOI 10.1007/978-0-387-09734-3_4,

In case the application at hand has some structure related to the input signal data vector, such as being composed by a tapped delay line, it is possible to derive a number of algorithms with lower computational complexity collectively known as *fast* algorithms.

From the conventional ($\mathcal{O}[N^2]$) QR decomposition method [1, 2], a number of *fast* algorithms ($\mathcal{O}[N]$) has been derived [3, 4]. These algorithms can be classified in terms of the type of triangularization applied to the input data matrix (upper or lower triangular) and type of error vector (*a posteriori* or *a priori*) involved in the updating process. As it can be verified from the Gram–Schmidt orthogonalization procedure, an *upper triangularization* of the data correlation matrix Cholesky factor (in the notation adopted in this work) involves the updating of *forward prediction* errors while a *lower triangularization* involves the updating of *backward prediction* errors.

This chapter presents a classification of a family of fast QRD-RLS algorithms along with their corresponding equations. Specifically, Table 4.1 shows how the algorithms discussed here will be designated hereafter as well as the classification of the type of prediction problem the algorithms rely.

Table 4.1 Classification of the fast QRD-RLS algorithms.

Error type	Prediction	
	Forward	Backward
A posteriori	FQR_POS_F [3]	FQR_POS_B [5, 6]
A priori	FQR_PRI_F [7]	FQR_PRI_B [4, 8]

It is worth mentioning that the FQR_PRI_B algorithm was independently developed in [8] and in [4] using distinct approaches and leading to two different versions. The approach which will be used here [4] was derived from concepts used in the inverse QRD-RLS algorithm [9]. The same algorithm (FQR_PRI_B) was also derived in [10] as a lattice extension of the inverse QRD-RLS algorithm [11].

In the derivation of fast QRD-RLS algorithms, we start by applying the QR decomposition to the backward and forward prediction problems whose prediction errors were defined in the previous chapter. We aim the triangularization of the extended order input data matrix $\mathbf{X}^{(N+2)}(k)$, from the expressions involving backward and forward predictions, in order to obtain $\mathbf{Q}^{(N+2)}(k)$, such that

$$\mathbf{Q}^{(N+2)}(k)\mathbf{X}^{(N+2)}(k) = \begin{bmatrix} \mathbf{0} \\ \mathbf{U}^{(N+2)}(k) \end{bmatrix}, \tag{4.1}$$

where the null matrix $\mathbf{0}$ above has dimension $(k-N-1)\times(N+2)$.

4.2 Upper Triangularization Algorithms (Updating Forward Prediction Errors)

The first algorithms derived here are those based on forward prediction errors, namely the FQR_POS_F [3] and the FQR_PRI_F [7] algorithms. These algorithms are presented here for completeness, due to their historical importance, and to pave the way for the next section which deals with a more attractive class of algorithm in terms of complexity and stability. As such, we suggest to a reader more interested in practical implementation to skip this section, and focus on the algorithms based on the updating of backward prediction errors.

Let us start, from the definition of the weighted backward prediction error vector $\mathbf{e}_b(k) = \mathbf{d}_b(k) - \mathbf{X}(k)\mathbf{w}_b(k)$, by pre-multiplying the weighted backward desired vector $\mathbf{d}_b(k) = \left[x(k-N-1) \ \cdots \ \lambda^{(k-N-1)/2}x(0) \ \mathbf{0}_{N+1}^{\mathrm{T}}\right]^{\mathrm{T}}$ by $\mathbf{Q}(k)$ and use the recursive structure of $\mathbf{Q}(k)$.[1] As a result, two important relations follow.

$$\| \mathbf{e}_b(k) \|^2 = e_{bq_1}^2(k) + \lambda \| \mathbf{e}_b(k-1) \|^2, \text{ and} \tag{4.2}$$

$$\begin{bmatrix} e_{bq_1}(k) \\ \mathbf{d}_{bq_2}(k) \end{bmatrix} = \mathbf{Q}_\theta(k) \begin{bmatrix} d_b(k) \\ \lambda^{1/2}\mathbf{d}_{bq_2}(k-1) \end{bmatrix}, \tag{4.3}$$

where $d_b(k) = x(k-N-1)$.

The (upper) triangularization, as seen in (4.1), of $\mathbf{X}^{(N+2)}(k)$, as defined in the previous chapter for the backward prediction problem, is achieved using three distinct matrices: $\mathbf{Q}^{(N+2)}(k) = \mathbf{Q}'_b(k)\mathbf{Q}_b(k)\mathbf{Q}(k)$, where $\mathbf{Q}_b(k)$ and $\mathbf{Q}'_b(k)$ are two sets of Givens rotations applied to generate, respectively, $\| \mathbf{e}_b(k) \|$ and $\| \mathbf{e}_b^{(0)}(k) \|$. The latter quantities are defined in the sequel. As a result, we have

$$\begin{aligned} \mathbf{U}^{(N+2)}(k) &= \mathbf{Q}'_{\theta b}(k) \begin{bmatrix} \mathbf{0}^{\mathrm{T}} & \| \mathbf{e}_b(k) \| \\ \mathbf{U}(k) & \mathbf{d}_{bq_2}(k) \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{z}^{\mathrm{T}}(k) & \| \mathbf{e}_b^{(0)}(k) \| \\ \mathbf{R}(k) & \mathbf{0} \end{bmatrix}, \end{aligned} \tag{4.4}$$

where $\mathbf{Q}'_{\theta b}(k)$ is a submatrix of $\mathbf{Q}'_b(k)$, $[\mathbf{z}(k)\mathbf{R}^{\mathrm{T}}(k)]^{\mathrm{T}}$ is the left part of $\mathbf{U}^{(N+2)}(k)$, just excluding the last column, and $\| \mathbf{e}_b^{(0)}(k) \|$ is the norm of the backward error of a predictor whose number of coefficients is zero.

In the forward prediction problem, the pre-multiplication of the forward weighted desired vector, $\mathbf{d}_f(k) = \left[x(k) \ \cdots \ \lambda^{k/2}x(0)\right]^{\mathrm{T}}$, by $\begin{bmatrix} \mathbf{Q}(k-1) & \mathbf{0} \\ \mathbf{0}^{\mathrm{T}} & 1 \end{bmatrix}$ and the use of the recurse expression of $\mathbf{Q}(k)$ in the weighted forward error vector $\mathbf{e}_f(k) = \mathbf{d}_f(k) -$

[1] The recursive structure of $\mathbf{Q}(k)$ is expressed by $\mathbf{Q}(k) = \tilde{\mathbf{Q}}(k) \begin{bmatrix} 1 & \mathbf{0}^{\mathrm{T}} \\ \mathbf{0} & \mathbf{Q}(k-1) \end{bmatrix}$.

$\begin{bmatrix} \mathbf{X}(k-1) \\ \mathbf{0}^{\mathrm{T}} \end{bmatrix} \mathbf{w}_f(k)$, leads to two other important relations given by

$$\| \mathbf{e}_f(k) \|^2 = e_{fq_1}^2(k) + \lambda \| \mathbf{e}_f(k-1) \|^2, \text{ and} \tag{4.5}$$

$$\begin{bmatrix} e_{fq_1}(k) \\ \mathbf{d}_{fq_2}(k) \end{bmatrix} = \mathbf{Q}_\theta(k-1) \begin{bmatrix} d_f(k) \\ \lambda^{1/2}\mathbf{d}_{fq_2}(k-1) \end{bmatrix}, \tag{4.6}$$

where $d_f(k) = x(k)$.

The upper triangularization of $\mathbf{U}^{(N+2)}(k)$ in the forward prediction problem is implemented by pre-multiplying $\mathbf{e}_f(k)$ by the product $\mathbf{Q}_f(k) \begin{bmatrix} \mathbf{Q}(k-1) & \mathbf{0} \\ \mathbf{0}^{\mathrm{T}} & 1 \end{bmatrix}$, where $\mathbf{Q}_f(k)$ is a set of Givens rotations generating $\| \mathbf{e}_f(k) \|$ by eliminating the first $k-N$ elements of the rotated desired vector of the forward predictor. The result is

$$\mathbf{U}^{(N+2)}(k) = \begin{bmatrix} \mathbf{d}_{fq_2}(k) & \mathbf{U}(k-1) \\ \| \mathbf{e}_f(k) \| & \mathbf{0}^{\mathrm{T}} \end{bmatrix}. \tag{4.7}$$

In order to avoid working with matrices of increasing dimensions, it is possible to eliminate the identity matrices that are part of the rotation matrices and are the source of the dimension increase. By eliminating these internal identity matrices, one can show that [2]

$$\mathbf{Q}_\theta^{(N+2)}(k) = \mathbf{Q}_{\theta f}(k) \begin{bmatrix} \mathbf{Q}_\theta(k-1) & \mathbf{0} \\ \mathbf{0}^{\mathrm{T}} & 1 \end{bmatrix}, \tag{4.8}$$

where $\mathbf{Q}_{\theta f}(k)$ is a single Givens rotation generating $\| \mathbf{e}_f(k) \|$ as in (4.5).

If we take the inverses of (4.4) and (4.7), the results are

$$\begin{aligned} [\mathbf{U}^{(N+2)}(k)]^{-1} &= \begin{bmatrix} \mathbf{0} & \mathbf{R}^{-1}(k) \\ \dfrac{1}{\|\mathbf{e}_b^{(0)}(k)\|} & \dfrac{-\mathbf{z}^{\mathrm{T}}(k)\mathbf{R}^{-1}(k)}{\|\mathbf{e}_b^{(0)}(k)\|} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{0}^{\mathrm{T}} & \dfrac{1}{\|\mathbf{e}_f(k)\|} \\ \mathbf{U}^{-1}(k-1) & \dfrac{-\mathbf{U}^{-1}(k-1)\mathbf{d}_{fq_2}(k)}{\|\mathbf{e}_f(k)\|} \end{bmatrix}. \end{aligned} \tag{4.9}$$

We can use the expressions of $[\mathbf{U}^{(N+2)}(k)]^{-1}$ given in (4.9) to obtain the vectors $\mathbf{f}^{(N+2)}(k+1)$ and $\mathbf{a}^{(N+2)}(k+1)$. The choices of these vectors generate distinct algorithms, that is, updating $\mathbf{f}(k)$ (*a posteriori* forward errors) leads to the FQR_POS_F algorithm [3] whereas updating $\mathbf{a}(k)$ (*a priori* forward errors) leads to the FQR_PRI_F algorithm [7].

4.2.1 The FQR_POS_F algorithm

In the FQR_POS_F algorithm, vector $\mathbf{f}^{(N+2)}(k+1) = [\mathbf{U}^{(N+2)}(k+1)]^{-\mathrm{T}} \mathbf{x}^{(N+2)}(k+1)$ is expressed in terms of the relations obtained in the forward and

backward prediction problems. We shall first use the expression for $[\mathbf{U}^{(N+2)}(k)]^{-1}$ in (4.9) that comes from the backward prediction evaluated at instant $k+1$ to calculate $\mathbf{f}^{(N+2)}(k+1)$. In this case, we replace $\mathbf{x}^{(N+2)}(k+1)$ by $[\mathbf{x}^{\mathrm{T}}(k+1)\ x(k-N)]^{\mathrm{T}}$ and then pre-multiply the result by $\mathbf{Q}'^{\mathrm{T}}_{\theta b}(k+1)$. The outcome, after some algebraic manipulations (using Equation (4.4) to help with the simplification of the expression), is

$$\mathbf{f}^{(N+2)}(k+1) = \mathbf{Q}'_{\theta b}(k+1) \begin{bmatrix} \frac{\varepsilon_b(k+1)}{\|\mathbf{e}_b(k+1)\|} \\ \mathbf{f}(k+1) \end{bmatrix}. \tag{4.10}$$

Using the expression for $[\mathbf{U}^{(N+2)}(k)]^{-1}$ originated from the forward prediction case, at instant $k+1$, and replacing $\mathbf{x}^{(N+2)}(k+1)$ by $[x(k+1)\ \mathbf{x}^{\mathrm{T}}(k)]^{\mathrm{T}}$, we obtain

$$\mathbf{f}^{(N+2)}(k+1) = \begin{bmatrix} \mathbf{f}(k) \\ \frac{\varepsilon_f(k+1)}{\|\mathbf{e}_f(k+1)\|} \end{bmatrix}. \tag{4.11}$$

By combining (4.10) and (4.11), it is possible to derive an expression to update $\mathbf{f}(k)$, which is given by

$$\begin{bmatrix} \frac{\varepsilon_b(k+1)}{\|\mathbf{e}_b(k+1)\|} \\ \mathbf{f}(k+1) \end{bmatrix} = \mathbf{Q}'_{\theta b}{}^{\mathrm{T}}(k+1) \begin{bmatrix} \mathbf{f}(k) \\ \frac{\varepsilon_f(k+1)}{\|\mathbf{e}_f(k+1)\|} \end{bmatrix}. \tag{4.12}$$

Once we have $\mathbf{f}(k+1)$, we can extract the angles of $\mathbf{Q}_\theta(k+1)$ by post-multiplying this matrix by the pinning vector $[1\,0\,\cdots\,0]^{\mathrm{T}}$. From the partitioned expression of $\mathbf{Q}_\theta(k)$, we can see that the result is

$$\mathbf{Q}_\theta(k+1) \begin{bmatrix} 1 \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \gamma(k+1) \\ \mathbf{f}(k+1) \end{bmatrix}. \tag{4.13}$$

However, the quantities required to compute the angles of $\mathbf{Q}'_{\theta_b}(k+1)$ are not available at instant k so that a special strategy is required. The updated $\mathbf{Q}'_{\theta_b}(k+1)$ is obtained [2, 12] with the use of vector $\mathbf{c}(k+1)$ defined as

$$\begin{aligned} \mathbf{c}(k+1) &= \hat{\mathbf{Q}}^{(N+2)}_\theta(k+1)\mathbf{Q}'_{\theta b}(k) \begin{bmatrix} 1 \\ \mathbf{0} \end{bmatrix} \\ &= \mathbf{Q}'_{\theta b}(k+1) \begin{bmatrix} b \\ \mathbf{0} \end{bmatrix}. \end{aligned} \tag{4.14}$$

The submatrix $\hat{\mathbf{Q}}^{(N+2)}_\theta(k+1)$ consisting of the last $(N+2)\times(N+2)$ elements of $\mathbf{Q}^{(N+2)}_\theta(k+1)$ is available from (4.8) (forward prediction) and b does not need to be explicitly calculated in order to obtain the angles θ'_{b_i}.

Finally, the joint process estimation is calculated with the same expressions previously used for the conventional QRD-RLS algorithm. The FQR_POS_F equations are presented in Table 4.2. A detailed description of this algorithm is found in Appendix 1.

Table 4.2 The FQRD_POS_F equations.

FQR_POS_F

for each k

{ Obtaining $e_{f_{q_1}}(k+1)$:

$$\begin{bmatrix} e_{fq_1}(k+1) \\ \mathbf{d}_{fq_2}(k+1) \end{bmatrix} = \mathbf{Q}_\theta(k) \begin{bmatrix} x(k+1) \\ \lambda^{1/2}\mathbf{d}_{fq_2}(k) \end{bmatrix}$$

Obtaining $\mathbf{Q}_{\theta f}(k+1)$:

$$\| \mathbf{e}_f(k+1) \| = \sqrt{e^2_{fq_1}(k+1) + \lambda \| \mathbf{e}_f(k) \|^2}$$

$$cos\theta_f(k+1) = \lambda^{1/2} \| \mathbf{e}_f(k) \| / \| \mathbf{e}_f(k+1) \|$$

$$sin\theta_f(k+1) = e_{fq_1}(k+1) / \| \mathbf{e}_f(k+1) \|$$

Obtaining $\mathbf{c}(k+1)$:

$$\mathbf{Q}_\theta^{(N+2)}(k+1) = \mathbf{Q}_{\theta f}(k+1) \begin{bmatrix} \mathbf{Q}_\theta(k) & \mathbf{0} \\ \mathbf{0}^{\mathrm{T}} & 1 \end{bmatrix}$$

$$\hat{\mathbf{Q}}_\theta^{(N+2)}(k+1) = \text{last } (N+2) \times (N+2) \text{ elements of } \mathbf{Q}_\theta^{(N+2)}(k+1)$$

$$\mathbf{c}(k+1) = \hat{\mathbf{Q}}_\theta^{(N+2)}(k+1)\mathbf{Q}'_{\theta b}(k) \begin{bmatrix} 1 \\ \mathbf{0} \end{bmatrix}$$

Obtaining $\mathbf{Q}'_{\theta b}(k+1)$:

$$\begin{bmatrix} b \\ \mathbf{0} \end{bmatrix} = {\mathbf{Q}'_{\theta b}}^{\mathrm{T}}(k+1)\mathbf{c}(k+1)$$

Obtaining $\mathbf{f}(k+1)$:

$$\begin{bmatrix} \frac{\varepsilon_b(k+1)}{\|\mathbf{e}_b(k+1)\|} \\ \mathbf{f}(k+1) \end{bmatrix} = {\mathbf{Q}'_{\theta b}}^{\mathrm{T}}(k+1) \begin{bmatrix} \mathbf{f}(k) \\ \frac{\varepsilon_f(k+1)}{\|\mathbf{e}_f(k+1)\|} \end{bmatrix}$$

Obtaining $\mathbf{Q}_\theta(k+1)$:

$$\begin{bmatrix} 1 \\ \mathbf{0} \end{bmatrix} = \mathbf{Q}_\theta^{\mathrm{T}}(k+1) \begin{bmatrix} \gamma(k+1) \\ \mathbf{f}(k+1) \end{bmatrix}$$

Joint Process Estimation:

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

$e(k+1) = e_{q_1}(k+1)/\gamma(k+1)$ % *a priori* error

$\varepsilon(k+1) = e_{q_1}(k+1)\gamma(k+1)$ % *a posteriori* error

}

4.2.2 The FQR_PRI_F algorithm

Expressing $\mathbf{a}^{(N+2)}(k+1) = [\mathbf{U}^{(N+2)}(k)]^{-\mathrm{T}}\mathbf{x}^{(N+2)}(k+1)/\sqrt{\lambda}$ in terms of the matrices in (4.9) and pre-multiplying the expression for $[\mathbf{U}^{(N+2)}(k)]^{-1}$ originated from the backward prediction problem by $\mathbf{Q}'_{\theta b}(k)\mathbf{Q}'^{\mathrm{T}}_{\theta b}(k)$ yields

$$\begin{bmatrix} \frac{e_b(k+1)}{\sqrt{\lambda}\|\mathbf{e}_b(k)\|} \\ \mathbf{a}(k+1) \end{bmatrix} = {\mathbf{Q}'_{\theta b}}^{\mathrm{T}}(k) \begin{bmatrix} \mathbf{a}(k) \\ \frac{e_f(k+1)}{\sqrt{\lambda}\|\mathbf{e}_f(k)\|} \end{bmatrix}. \tag{4.15}$$

Given $\mathbf{a}(k+1)$, the angles of $\mathbf{Q}_\theta(k+1)$ are found through the following relation obtained by post-multiplying $\mathbf{Q}_\theta^{\mathrm{T}}(k+1)$ by the pinning vector.

$$\begin{bmatrix} 1/\gamma(k+1) \\ \mathbf{0} \end{bmatrix} = \mathbf{Q}_\theta(k+1) \begin{bmatrix} 1 \\ -\mathbf{a}(k+1) \end{bmatrix} \tag{4.16}$$

By noting that the angles of $\mathbf{Q}'_{\theta_b}(k+1)$ can be updated with the same procedure used in the FQR_POS_F algorithm, we already have all the necessary equations of the fast QR-RLS algorithm as presented in Table 4.3. The detailed description of this algorithm is also found in Appendix 1.

Table 4.3 The FQRD_PRI_F equations.

FQR_PRI_F

for each k

{ Obtaining $e_f(k+1)$:

$$\begin{bmatrix} e_{fq_1}(k+1) \\ \mathbf{d}_{fq_2}(k+1) \end{bmatrix} = \mathbf{Q}_\theta(k) \begin{bmatrix} x(k+1) \\ \lambda^{1/2}\mathbf{d}_{fq_2}(k) \end{bmatrix}$$

$$e_f(k+1) = e_{fq_1}(k+1)/\gamma(k)$$

Obtaining $\mathbf{a}(k+1)$:

$$\begin{bmatrix} \frac{e_b(k+1)}{\sqrt{\lambda}\|\mathbf{e}_b(k)\|} \\ \mathbf{a}(k+1) \end{bmatrix} = \mathbf{Q}'_{\theta b}{}^{\mathrm{T}}(k) \begin{bmatrix} \mathbf{a}(k) \\ \frac{e_f(k+1)}{\sqrt{\lambda}\|\mathbf{e}_f(k)\|} \end{bmatrix}$$

Obtaining $\mathbf{Q}_{\theta f}(k+1)$:

$$\| \mathbf{e}_f(k+1) \| = \sqrt{e^2_{fq_1}(k+1) + \lambda \| \mathbf{e}_f(k) \|^2}$$

$$cos\theta_f(k+1) = \lambda^{1/2} \| \mathbf{e}_f(k) \| / \| \mathbf{e}_f(k+1) \|$$

$$sin\theta_f(k+1) = e_{fq_1}(k+1) / \| \mathbf{e}_f(k+1) \|$$

Obtaining $\mathbf{c}(k+1)$:

$$\mathbf{Q}_\theta^{(N+2)}(k+1) = \mathbf{Q}_{\theta f}(k+1) \begin{bmatrix} \mathbf{Q}_\theta(k) & \mathbf{0} \\ \mathbf{0}^{\mathrm{T}} & 1 \end{bmatrix}$$

$$\hat{\mathbf{Q}}_\theta^{(N+2)}(k+1) = \text{last } (N+2) \times (N+2) \text{ elements of } \mathbf{Q}_\theta^{(N+2)}(k+1)$$

$$\mathbf{c}(k+1) = \hat{\mathbf{Q}}_\theta^{(N+2)}(k+1)\mathbf{Q}'_{\theta b}(k) \begin{bmatrix} 1 \\ \mathbf{0} \end{bmatrix}$$

Obtaining $\mathbf{Q}'_{\theta b}(k+1)$:

$$\begin{bmatrix} b \\ \mathbf{0} \end{bmatrix} = \mathbf{Q}'_{\theta b}{}^{\mathrm{T}}(k+1)\mathbf{c}(k+1)$$

Obtaining $\mathbf{Q}_\theta(k+1)$:

$$\begin{bmatrix} 1/\gamma(k+1) \\ \mathbf{0} \end{bmatrix} = \mathbf{Q}_\theta(k+1) \begin{bmatrix} 1 \\ -\mathbf{a}(k+1) \end{bmatrix}$$

Joint Process Estimation:

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

$e(k+1) = e_{q_1}(k+1)/\gamma(k+1)$ % *a priori* error

$\varepsilon(k+1) = e_{q_1}(k+1)\gamma(k+1)$ % *a posteriori* error

}

4.3 Lower Triangularization Algorithms (Updating Backward Prediction Errors)

Following similar steps as in the upper triangularization, it is possible to obtain the lower triangular matrix $\mathbf{U}^{(N+2)}(k)$ from the forward and backward prediction problems.

Before presenting the derivations, we remark that the fast QRD-RLS algorithms with lower triangularization of the input data matrix or, equivalently, updating backward prediction errors, are of minimal complexity and backward stable under persistent excitation [5, 11].

In the backward prediction problem, the lower triangular $\mathbf{U}^{(N+2)}(k)$ is obtained through the use of $\mathbf{Q}^{(N+2)}(k) = \mathbf{Q}_b(k)\mathbf{Q}(k)$, where $\mathbf{Q}_b(k)$ is a set of Givens rotations applied to generate $\| \mathbf{e}_b(k) \|$. The resulting Cholesky factor is

$$\mathbf{U}^{(N+2)}(k) = \begin{bmatrix} \mathbf{0}^{\mathrm{T}} & \| \mathbf{e}_b(k) \| \\ \mathbf{U}(k) & \mathbf{d}_{bq_2}(k) \end{bmatrix}. \tag{4.17}$$

On the other hand, in the forward prediction problem, the lower triangular matrix $\mathbf{U}^{(N+2)}(k)$ is formed by pre-multiplying the forward weighted error vector $\mathbf{e}_f(k)$ by the product $\mathbf{Q}'_f(k)\mathbf{Q}_f(k)\begin{bmatrix} \mathbf{Q}(k-1) & \mathbf{0} \\ \mathbf{0}^{\mathrm{T}} & 1 \end{bmatrix}$, where $\mathbf{Q}_f(k)$ and $\mathbf{Q}'_f(k)$ are two sets of Givens rotations generating $\| \mathbf{e}_f(k) \|$ and $\| \mathbf{e}_f^{(0)}(k) \|$, respectively. The resulting expression is

$$\mathbf{U}^{(N+2)}(k) = \mathbf{Q}'_{\theta f}(k) \begin{bmatrix} \mathbf{d}_{fq_2}(k) & \mathbf{U}(k-1) \\ \| \mathbf{e}_f(k) \| & \mathbf{0}^{\mathrm{T}} \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{R}(k) \\ \| \mathbf{e}_f^{(0)}(k) \| & \mathbf{z}^{\mathrm{T}}(k) \end{bmatrix}, \tag{4.18}$$

where $[\mathbf{R}^{\mathrm{T}}(k)\ \mathbf{z}(k)]^{\mathrm{T}}$ represents the last $(N+1)$ columns of $\mathbf{U}^{(N+2)}(k)$. By considering the fact that (4.2), (4.3), (4.5), and (4.6) hold, $\| \mathbf{e}_f(k) \|$ can be recursively computed using (4.5).

Taking the inverse of (4.17) and (4.18), we have the following results:

$$\begin{aligned} [\mathbf{U}^{(N+2)}(k)]^{-1} &= \begin{bmatrix} \frac{-\mathbf{U}^{-1}(k)\mathbf{d}_{bq_2}(k)}{\|\mathbf{e}_b(k)\|} & \mathbf{U}^{-1}(k) \\ \frac{1}{\|\mathbf{e}_b(k)\|} & \mathbf{0}^{\mathrm{T}} \end{bmatrix} \\ &= \begin{bmatrix} \frac{-\mathbf{z}^{\mathrm{T}}(k)\mathbf{R}^{-1}(k)}{\|\mathbf{e}_f^{(0)}(k)\|} & \frac{1}{\|\mathbf{e}_f^{(0)}(k)\|} \\ \mathbf{R}^{-1}(k) & \mathbf{0} \end{bmatrix}. \end{aligned} \tag{4.19}$$

With the results obtained from (4.19), we can once more express vectors $\mathbf{f}^{(N+2)}(k+1)$ and $\mathbf{a}^{(N+2)}(k+1)$ in terms of the submatrices of $\left[\mathbf{U}^{(N+2)}(k+1)\right]^{-1}$. If we update $\mathbf{f}(k)$, the resulting algorithm is the FQR_POS_B whereas, by updating $\mathbf{a}(k)$, one generates the FQR_PRI_B algorithm.

4.3.1 The FQR_POS_B algorithm

Expressing $\mathbf{f}^{(N+2)}(k+1) = [\mathbf{U}^{(N+2)}(k+1)]^{-\mathrm{T}}\mathbf{x}^{(N+2)}(k+1)$ in terms of the matrix in (4.19) originating from the the forward prediction problem, and pre-multiplying the resulting expression by $\mathbf{Q}'_{\theta f}(k+1)\mathbf{Q}'^{\mathrm{T}}_{\theta f}(k+1)$ yields

$$\begin{bmatrix} \frac{\varepsilon_b(k+1)}{\|\mathbf{e}_b(k+1)\|} \\ \mathbf{f}(k+1) \end{bmatrix} = \mathbf{Q}'_{\theta_f}(k+1) \begin{bmatrix} \mathbf{f}(k) \\ \frac{\varepsilon_f(k+1)}{\|\mathbf{e}_f(k+1)\|} \end{bmatrix}. \tag{4.20}$$

For this particular algorithm, we provide extra implementation details which are similar to the other fast QRD-RLS algorithms. We start by pointing out that matrix $\mathbf{Q}'_{\theta_f}(k)$ in (4.20) corresponds to the set of rotations used in (4.18) to eliminate $\mathbf{d}_{fq_2}(k)$ over $||\mathbf{e}_f(k)||$ such that the resulting matrix $\mathbf{U}^{(N+2)}(k)$ is lower triangular.

Therefore, making explicit the structure of $\mathbf{Q}'_{\theta_f}(k)$, the part of (4.18) given by the product $\mathbf{Q}'_{\theta_f}(k)\left[\mathbf{d}^{\mathrm{T}}_{fq_2}(k) \;\; \| \, \mathbf{e}_f(k) \, \|\right]^{\mathrm{T}}$ can be expressed as

$$\begin{bmatrix} \mathbf{0} \\ \| \, \mathbf{e}_f^{(0)}(k) \, \| \end{bmatrix} = \mathbf{Q}'_{\theta f}(k) \begin{bmatrix} \mathbf{d}_{fq_2}(k) \\ \| \, \mathbf{e}_f(k) \, \| \end{bmatrix} = \begin{bmatrix} \mathbf{I}_N & \mathbf{0} & \mathbf{0} \\ \mathbf{0}^{\mathrm{T}} & \cos\theta'_{f_N}(k) & -\sin\theta'_{f_N}(k) \\ \mathbf{0}^{\mathrm{T}} & \sin\theta'_{f_N}(k) & \cos\theta'_{f_N}(k) \end{bmatrix} \cdots$$

$$\cdots \underbrace{\begin{bmatrix} \cos\theta'_{f_0}(k) & \mathbf{0}^{\mathrm{T}} & -\sin\theta'_{f_0}(k) \\ \mathbf{0} & \mathbf{I}_N & \mathbf{0} \\ \sin\theta'_{f_0}(k) & \mathbf{0}^{\mathrm{T}} & \cos\theta'_{f_0}(k) \end{bmatrix} \begin{bmatrix} d_{fq2_1}(k) \\ \vdots \\ d_{fq2_{N+1}}(k) \\ \| \, \mathbf{e}_f^{(N+1)}(k) \, \| \end{bmatrix}}_{\begin{bmatrix} 0 \\ \vdots \\ d_{fq2_{N+1}}(k) \\ \| \, \mathbf{e}_f^{(N)}(k) \, \| \end{bmatrix}}, \tag{4.21}$$

where, for this first multiplication, the quantity $\cos\theta'_{f_0}(k)d_{fq2_1}(k) - \sin\theta'_{f_0}(k) \, \| \, \mathbf{e}_f^{(N+1)}(k) \, \|$ was made zero and the quantity $\sin\theta'_{f_0}(k)d_{fq2_1}(k) + \cos\theta'_{f_0}(k) \, \| \, \mathbf{e}_f^{(N+1)}(k) \, \|$ corresponds to $\| \, \mathbf{e}_f^{(N)}(k) \, \|$.

In the ith multiplication, for i ranging from 1 to $N+1$, we have:

$$\begin{cases} \cos\theta'_{f_{i-1}}(k)d_{fq2_i}(k) = \sin\theta'_{f_{i-1}}(k) \, \| \, \mathbf{e}_f^{(N+2-i)}(k) \, \| \\ \| \, \mathbf{e}_f^{(N+1-i)}(k) \, \| = \sin\theta'_{f_{i-1}}(k)d_{fq2_i}(k) + \cos\theta'_{f_{i-1}}(k) \, \| \, \mathbf{e}_f^{(N+2-i)}(k) \, \| \end{cases} \tag{4.22}$$

The two expressions in (4.22) are represented graphically in Figure 4.1. From this figure, one clearly observes that $\| \, \mathbf{e}_f^{(N+1-i)}(k) \, \|$ can also be computed as follows.

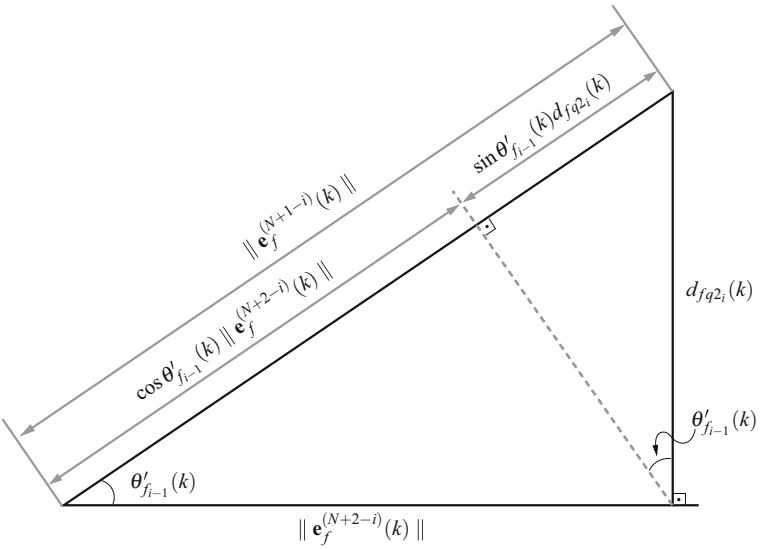

Fig. 4.1 Multiplication of $\mathbf{Q}'_{\theta f_i}(k)$: computing $\cos\theta'_{f_{i-1}}(k)$ and $\sin\theta'_{f_{i-1}}(k)$.

$$\| \mathbf{e}_f^{(N+1-i)}(k) \| = \sqrt{d_{fq2_i}^2(k) + \| \mathbf{e}_f^{(N+2-i)}(k) \|^2} \tag{4.23}$$

After computing $\| \mathbf{e}_f^{(N+1-i)}(k) \|$, the sine and the cosine are given below.

$$\begin{cases} \cos\theta'_{f_{i-1}}(k) = \dfrac{\|\mathbf{e}_f^{(N+2-i)}(k)\|}{\|\mathbf{e}_f^{(N+1-i)}(k)\|} \\ \sin\theta'_{f_{i-1}}(k) = \dfrac{d_{fq2_i}(k+1)}{\|\mathbf{e}_f^{(N+1-i)}(k)\|} \end{cases} \tag{4.24}$$

In the derivation of (4.20), it can be observed that the last element of $\mathbf{f}(k+1)$ is $\frac{x(k+1)}{\|\mathbf{e}_f^{(0)}(k+1)\|}$. The term $\frac{\varepsilon_f(k+1)}{\|\mathbf{e}_f(k+1)\|}$ can be calculated as $\gamma(k)\sin\theta_f(k+1)$ where $\sin\theta_f(k+1) = \frac{e_{fq_1}(k+1)}{\|\mathbf{e}_f(k+1)\|}$ is the sine of the angle of rotation matrix $\mathbf{Q}_f(k+1)$.

Using or not this information, the prior knowledge of the last element of $\mathbf{f}(k+1)$, leads to two distinct versions of the FQR_POS_B algorithm. The first version [6] uses this information as following discussed. Multiplying (4.20) by $\mathbf{Q}'^{\mathrm{T}}_{\theta f}$, we obtain the relation (4.25) with $\frac{\varepsilon_b(k+1)}{\|\mathbf{e}_b(k+1)\|}$ being represented by $f_0(k+1)$, $\frac{\varepsilon_f(k+1)}{\|\mathbf{e}_f(k+1)\|}$ by $f_{N+2}(K)$, and the ith element of $\mathbf{f}(k)$ by $f_i(k)$, for i ranging from 1 to $N+1$.

$$\begin{bmatrix} \mathbf{f}(k) \\ f_{N+2}(k) \end{bmatrix} = \begin{bmatrix} f_1(k) \\ \vdots \\ f_{N+1}(k) \\ f_{N+2}(k) \end{bmatrix} = \mathbf{Q}'^{\mathrm{T}}_{\theta f}(k) \begin{bmatrix} f_0(k+1) \\ \mathbf{f}(k+1) \end{bmatrix}$$

$$= \begin{bmatrix} \cos\theta'_{f_0}(k) & \mathbf{0}^{\mathrm{T}} & \sin\theta'_{f_0}(k) \\ \mathbf{0} & \mathbf{I}_N & \mathbf{0} \\ -\sin\theta'_{f_0}(k) & \mathbf{0}^{\mathrm{T}} & \cos\theta'_{f_0}(k) \end{bmatrix} \cdots \mathbf{Q}'^{\mathrm{T}}_{\theta f_{N+1-i}}(k+1) \cdots$$

$$\cdots \begin{bmatrix} \mathbf{I}_N & \mathbf{0} & \mathbf{0} \\ \mathbf{0}^{\mathrm{T}} & \cos\theta'_{f_N}(k) & \sin\theta'_{f_N}(k) \\ \mathbf{0}^{\mathrm{T}} & -\sin\theta'_{f_N}(k) & \cos\theta'_{f_N}(k) \end{bmatrix} \underbrace{\begin{bmatrix} f_0(k+1) \\ \vdots \\ f_{N-1}(k+1) \\ f_N(k+1) \\ f_{N+1}(k+1) \end{bmatrix}}_{\begin{bmatrix} f_0(K+1) \\ \vdots \\ f_{N-1}(k+1) \\ f_{N+1}(k) \\ aux_1 \end{bmatrix}}, \tag{4.25}$$

In the last multiplication above, the results (as shown below the *underbrace*) denoted by $f_{N+1}(k)$ and aux_1 are such that, for the ith multiplication (with i from 1 to $N+1$), we have:

$$f_{N+2-i}(k) = \cos\theta'_{f_{N+1-i}}(k+1) f_{N+1-i}(k+1) - \sin\theta'_{f_{N+1-i}}(k+1) aux_{i-1} \tag{4.26}$$

$$aux_i = -\sin\theta'_{f_{N+1-i}}(k+1) f_{N+1-i}(k+1) + \cos\theta'_{f_{N+1-i}}(k+1) aux_{i-1} \tag{4.27}$$

where $aux_0 = f_{N+1}(k+1)$ is the last element of $\mathbf{f}(k+1)$ which is known *a priori*.

From (4.26), we can easily obtain $f_{N+1-i}(k+1)$ as in Equation (4.28) and with this value compute aux_i in (4.27) such that $f_{N+2}(k) = aux_{N+1}$ (which is actually not used afterwards).

$$f_{N+1-i}(k+1) = \frac{f_{N+2-i}(k) - \sin\theta'_{f_{N+1-i}}(k+1) aux_{i-1}}{\cos\theta'_{f_{N+1-i}}(k+1)} \tag{4.28}$$

In the second version of this algorithm [5], we compute first $\frac{e_f(k+1)}{\|\mathbf{e}_f(k+1)\|} = \frac{\gamma(k) e_{fq_1}(k+1)}{\|\mathbf{e}_f(k+1)\|}$ and then (4.20) is used in a straightforward manner.

Once we have $\mathbf{f}(k+1)$, we find $\mathbf{Q}_\theta(k+1)$ with the same relation used in the upper triangularization algorithms, (4.13). Moreover, the joint process estimation is carried out in the same manner as in the forward prediction-based algorithms.

Table 4.4 The FQR_POS_B algorithm.

FQR_POS_B

for each k

{ Obtaining $\mathbf{d}_{fq_2}(k+1)$:

$$\begin{bmatrix} e_{fq_1}(k+1) \\ \mathbf{d}_{fq_2}(k+1) \end{bmatrix} = \mathbf{Q}_\theta(k) \begin{bmatrix} x^*(k+1) \\ \lambda^{1/2}\mathbf{d}_{fq_2}(k) \end{bmatrix}$$

Obtaining $\| \mathbf{e}_f(k+1) \|$:

$$\| \mathbf{e}_f(k+1) \| = \sqrt{|e_{fq_1}(k+1)|^2 + \lambda \| \mathbf{e}_f(k) \|^2}$$

Obtaining $\mathbf{Q}'_{\theta f}(k+1)$:

$$\begin{bmatrix} \mathbf{0} \\ \| \mathbf{e}_f^{(0)}(k+1) \| \end{bmatrix} = \mathbf{Q}'_{\theta f}(k+1) \begin{bmatrix} \mathbf{d}_{fq_2}(k+1) \\ \| \mathbf{e}_f(k+1) \| \end{bmatrix}$$

Obtaining $\mathbf{f}(k+1)$:

$$\begin{bmatrix} \frac{\varepsilon_b(k+1)}{\|\mathbf{e}_b(k+1)\|} \\ \mathbf{f}(k+1) \end{bmatrix} = \mathbf{Q}'_{\theta f}(k+1) \begin{bmatrix} \mathbf{f}(k) \\ \frac{\varepsilon_f(k+1)}{\|\mathbf{e}_f(k+1)\|} \end{bmatrix}$$

Obtaining $\mathbf{Q}_\theta(k+1)$:

$$\begin{bmatrix} 1 \\ \mathbf{0} \end{bmatrix} = \mathbf{Q}_\theta^{\mathrm{T}}(k+1) \begin{bmatrix} \gamma(k+1) \\ \mathbf{f}(k+1) \end{bmatrix}$$

Joint Process Estimation:

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

$e(k+1) = e^*_{q_1}(k+1)/\gamma(k+1)$ % *a priori* error

$\varepsilon(k+1) = e^*_{q_1}(k+1)\gamma(k+1)$ % *a posteriori* error

}

As a result, we have now available all the required expressions composing the FQR_POS_B algorithm as presented in Table 4.4. The detailed descriptions of two different versions of this algorithm is found in Appendix 2.

4.3.2 The FQR_PRI_B algorithm

This last algorithm of this family is obtained by expressing vector $\mathbf{a}^{(N+2)}(k+1) = [\mathbf{U}^{(N+2)}(k)]^{-\mathrm{T}}\mathbf{x}^{(N+2)}(k+1)/\sqrt{\lambda}$ in terms of the expression for $[\mathbf{U}^{(N+2)}(k)]^{-1}$ in (4.19) originated from the forward prediction problem. Next, by pre-multiplying $\mathbf{a}^{(N+2)}(k+1)$ by $\mathbf{Q}'_{\theta f}(k)\mathbf{Q}'^{\mathrm{T}}_{\theta f}(k)$, the following updating equation results.

$$\begin{bmatrix} \frac{e_b(k+1)}{\sqrt{\lambda}\|\mathbf{e}_b(k)\|} \\ \mathbf{a}(k+1) \end{bmatrix} = \mathbf{Q}'_{\theta f}(k) \begin{bmatrix} \mathbf{a}(k) \\ \frac{e_f(k+1)}{\sqrt{\lambda}\|\mathbf{e}_f(k)\|} \end{bmatrix} \tag{4.29}$$

It is again important to mention that the last element of $\mathbf{a}(k+1)$ in (4.29) is already known to be equal to $\frac{x(k+1)}{\sqrt{\lambda}\|\mathbf{e}_f^{(0)}(k)\|}$. This fact leads to two different versions of the same algorithm: the same approach used in (4.20) for vector $\mathbf{f}(k)$ can be employed here for vector $\mathbf{a}(k)$.

Table 4.5 The FQR_PRI_B algorithm.

FQR_PRI_B

for each k

{ Obtaining $\mathbf{d}_{fq_2}(k+1)$:

$$\begin{bmatrix} e_{fq_1}(k+1) \\ \mathbf{d}_{fq_2}(k+1) \end{bmatrix} = \mathbf{Q}_\theta(k) \begin{bmatrix} x^*(k+1) \\ \lambda^{1/2}\mathbf{d}_{fq_2}(k) \end{bmatrix}$$

Obtaining $\mathbf{a}(k+1)$:

$$\begin{bmatrix} \frac{e_b(k+1)}{\sqrt{\lambda}\|\mathbf{e}_b(k)\|} \\ \mathbf{a}(k+1) \end{bmatrix} = \mathbf{Q}'_{\theta f}(k) \begin{bmatrix} \mathbf{a}(k) \\ \frac{e_f(k+1)}{\sqrt{\lambda}\|\mathbf{e}_f(k)\|} \end{bmatrix}$$

Obtaining $\| \mathbf{e}_f(k+1) \|$:

$$\| \mathbf{e}_f(k+1) \| = \sqrt{|e_{fq_1}(k+1)|^2 + \lambda \| \mathbf{e}_f(k) \|^2}$$

Obtaining $\mathbf{Q}'_{\theta f}(k+1)$:

$$\begin{bmatrix} \mathbf{0} \\ \| \mathbf{e}_f^{(0)}(k+1) \| \end{bmatrix} = \mathbf{Q}'_{\theta f}(k+1) \begin{bmatrix} \mathbf{d}_{fq_2}(k+1) \\ \| \mathbf{e}_f(k+1) \| \end{bmatrix}$$

Obtaining $\mathbf{Q}_\theta(k+1)$:

$$\begin{bmatrix} 1/\gamma(k+1) \\ \mathbf{0} \end{bmatrix} = \mathbf{Q}_\theta(k+1) \begin{bmatrix} 1 \\ -\mathbf{a}(k+1) \end{bmatrix}$$

Joint Process Estimation:

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

$e(k+1) = e^*_{q_1}(k+1)/\gamma(k+1)$ % *a priori* error

$\varepsilon(k+1) = e^*_{q_1}(k+1)\gamma(k+1)$ % *a posteriori* error

}

Once more, if we have $\mathbf{a}(k+1)$, we can find $\mathbf{Q}_\theta(k+1)$ using (4.16) and the joint process estimation is carried out as seen in the previous chapter, i.e., updating $\mathbf{d}_{q_2}(k)$ from $d(k)$ and $\mathbf{d}_{q_2}(k-1)$ as well as computing the error (*a priori* or *a posteriori*) from the *rotated* error $e_{q_1}(k)$. The FQR_PRI_B equations are presented in Table 4.5. The detailed descriptions of the two versions of this algorithm is found in Appendix 2.

In terms of computational complexity, Table 4.6 shows the comparisons among the four fast QRD algorithms according to their detailed pseudo-codes in Appendices 1 and 2. Note that $p = N+1$ is the number of coefficients.

Table 4.6 Comparison of computational complexity.

ALGORITHM	ADD	MULT.	DIV.	SQRT
FQR_POS_F [3]	10p+3	26p+10	3p+2	2p+1
FQR_PRI_F [7]	10p+3	26p+11	4p+4	2p+1
FQR_POS_B (VERSION 1) [6]	8p+1	19p+4	4p+1	2p+1
FQR_POS_B (VERSION 2) [5]	8p+1	20p+5	3p+1	2p+1
FQR_PRI_B (VERSION 1) [4]	8p-1	19p+2	5p+1	2p+1
FQR_PRI_B (VERSION 2) [8]	8p+1	20p+6	4p+2	2p+1

4.4 The Order Recursive Versions of the Fast QRD Algorithms

The fast QRD-RLS algorithms employing lower triangularization of the input data matrix are known as "hybrid QR-lattice least squares algorithms". It is clear from previous sections that these algorithms may update the *a posteriori* or the *a priori* backward prediction errors. Moreover, they are known for their robust numerical behavior and minimal complexity but lack the pipelining property of the lattice algorithms.

The main goal of this section is the presentation of the order recursive (or lattice) versions of the fast QR algorithms using *a posteriori* and *a priori* backward errors or FQR_POS_B and FQR_PRI_B algorithms according to our classification. The equations of these algorithms are combined in an order recursive manner such that they may be represented as increasing order single-loop lattice algorithms [13]. These order recursive versions can then be implemented with a modular structure, which utilizes a unique type of lattice stage for each algorithm.

Before their derivation, in order to help their understanding, let us specify in Table 4.7 the meaning of each variable used in both algorithms. It is worth men-

Table 4.7 Summary of variables used in FQR_POS_B and FQR_PRI_B algorithms.

$\mathbf{d}_{fq}(k)$	: rotated forward desired vector
$\mathbf{d}_{fq2}(k)$	: last $N+1$ elements of $\mathbf{d}_{fq}(k)$
$\mathbf{e}_f(k)$	: forward error vector
$\| \mathbf{e}_f(k) \|$	: norm of $\mathbf{e}_f(k)$
$\mathbf{e}_{fq}(k)$	: rotated $\mathbf{e}_f(k)$
$e_{fq_1}(k)$	: first element of $\mathbf{e}_{fq}(k)$
$\mathbf{Q}_\theta(k)$	: Givens matrix (updates the Cholesky factor)
$x(k)$	: input signal
λ	: forgetting factor
$\mathbf{Q}'_{\theta f}(k+1)$	: Givens matrix that annihilates $\mathbf{d}_{fq2}(k+1)$ in (4.18)
$\| \mathbf{e}_f^{(0)}(k) \|$	: norm of $\mathbf{e}_f(k)$ in a zero coefficient case
$\mathbf{f}(k)$	: *a posteriori* normalized errors
$\mathbf{a}(k)$	: *a priori* normalized errors
$\mathbf{e}_b(k)$	: backward error vector
$\| \mathbf{e}_b(k) \|$	: norm of $\mathbf{e}_b(k)$
$\varepsilon_f(k)$	: *a posteriori* forward prediction error
$e_f(k)$	: *a priori* forward prediction error
$\varepsilon_b(k)$	: *a posteriori* backward prediction error
$e_b(k)$	: *a priori* backward prediction error
$\gamma(k)$	: product of cosines of the angles of $\mathbf{Q}_\theta(k)$
$\mathbf{e}_q(k)$	: rotated error vector
$e_{q_1}(k)$	: first element of $\mathbf{e}_q(k)$
$\mathbf{d}_q(k)$	: rotated desired vector
$\mathbf{d}_{q_2}(k)$	: last $N+1$ elements of $\mathbf{d}_q(k)$
$d(k)$	: desired signal
$e(k)$	: *a priori* output error

tioning here that a variable with no superscript implies in an N-th order quantity or, equivalently, is related to an $N+1$ coefficients filtering. Let us take as illustration the norm of the forward energy: $\| \mathbf{e}_f(k) \| = \| \mathbf{e}_f^{(N+1)}(k) \|$.

The internal variables found in fast QR algorithms are closely related to those found in conventional lattice algorithms. This was indeed the approach used in [5, 8] to develop these algorithms originally and the implications are well explained in those two references. As pointed out in [5], within this framework was the solution to the parameter identification problem first addressed using fast QR algorithms. The work of [14] stresses the fact that $\sin\theta'_{f_i}(k)$ and $\sin\theta'_{f_i}(k-1)$ represent the reflection coefficients of the normalized lattice RLS algorithms (*a priori* and *a posteriori*).

On the other hand, the main idea behind the generation of a lattice (or *fully* lattice) version of the fast QR algorithms is the merging of their equations using order updating instead of fixed order variables. This can be done when partial results possess this order updating property. This is indeed the case of the lower triangularization type algorithms since the internal variables are synchronized at instant k or $k-1$ (only order updating). The same facility in obtaining lattice versions is not observed in those algorithms employing upper triangularization (FQR_POS_F and FQR_PRI_F) since the normalized errors present in the orthogonal matrix $\mathbf{Q}_\theta(k)$ are of different orders **at distinct instants of time** (order and time updating).

We next show how to combine the equations of FQR_POS_B in order to obtain its order recursive version. Starting from (4.6), we rewrite this equation evaluated at $k+1$, with an explicit form of $\mathbf{Q}_\theta(k)$ in terms of a product of $N+1$ Givens rotations $\mathbf{Q}_{\theta_i}(k)$ and with $e_{fq_1}^{(0)}(k+1) = x(k+1)$; we suggest the reader to check, in the previous chapter, the structure of $\mathbf{Q}_{\theta_i}(k)$ for the lower triangularization case.

$$\begin{bmatrix} e_{fq_1}(k+1) \\ d_{fq2_1}(k+1) \\ \vdots \\ d_{fq2_{N+1}}(k+1) \end{bmatrix} = \begin{bmatrix} \cos\theta_N(k) & -\sin\theta_N(k) & \mathbf{0}^{\mathrm{T}} \\ \sin\theta_N(k) & \cos\theta_N(k) & \mathbf{0}^{\mathrm{T}} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_N \end{bmatrix} \cdots$$

$$\cdots \begin{bmatrix} \cos\theta_0(k) & \mathbf{0}^{\mathrm{T}} & -\sin\theta_0(k) \\ \mathbf{0} & \mathbf{I}_N & \mathbf{0} \\ \sin\theta_0(k) & \mathbf{0}^{\mathrm{T}} & \cos\theta_0(k) \end{bmatrix} \begin{bmatrix} e_{fq_1}^{(0)}(k+1) \\ \lambda^{1/2} d_{fq2_1}(k) \\ \vdots \\ \lambda^{1/2} d_{fq2_{N+1}}(k) \end{bmatrix} \tag{4.30}$$

The product of the first two terms, from right to left, results in

$$\begin{bmatrix} \cos\theta_0(k) e_{fq_1}^{(0)}(k+1) - \sin\theta_0 \lambda^{1/2} d_{fq2_{N+1}}(k) \\ \lambda^{1/2} d_{fq2_1}(k) \\ \vdots \\ \lambda^{1/2} d_{fq2_N}(k) \\ \sin\theta_0(k) e_{fq_1}^{(0)}(k+1) + \cos\theta_0(k) \lambda^{1/2} d_{fq2_{N+1}}(k) \end{bmatrix}. \tag{4.31}$$

The first and last terms of the above equation are, respectively, $e_{fq_1}^{(1)}(k+1)$ and $d_{fq2_{N+1}}(k+1)$. If the other products are computed, one can reach the following relations:

$$e_{fq_1}^{(i)}(k+1) = \cos\theta_{i-1}(k)e_{fq_1}^{(i-1)}(k+1) - \sin\theta_{i-1}(k)\lambda^{1/2}d_{fq2_{N+2-i}}(k) \tag{4.32}$$

$$d_{fq2_{N+2-i}}(k+1) = \sin\theta_{i-1}(k)e_{fq_1}^{(i-1)}(k+1) + \cos\theta_{i-1}(k)\lambda^{1/2}d_{fq2_{N+2-i}}(k) \tag{4.33}$$

where i belongs to the closed interval between 1 and $N+1$.

If we use a similar procedure with the equation $\begin{bmatrix} \mathbf{0} \\ \| \mathbf{e}_f^{(0)}(k+1) \| \end{bmatrix} = \mathbf{Q}'_{\theta f}(k+1) \begin{bmatrix} \mathbf{d}_{fq_2}(k+1) \\ \| \mathbf{e}_f(k+1) \| \end{bmatrix}$ which is part of (4.18) used in the FQR_POS_B algorithm, we will find

$$\cos\theta'_{f_{i-1}}(k+1) = \frac{\| \mathbf{e}_f^{(i)}(k+1) \|}{\| \mathbf{e}_f^{(i-1)}(k+1) \|}, \text{ and} \tag{4.34}$$

$$\sin\theta'_{f_{i-1}}(k+1) = \frac{d_{fq2_{N+2-i}}(k+1)}{\| \mathbf{e}_f^{(i-1)}(k+1) \|}. \tag{4.35}$$

In the last equation, i varies from 1 to $N+1$ and the updating of the forward error energy is performed by the following generalization of (4.5)

$$\| \mathbf{e}_f^{(i)}(k+1) \| = \sqrt{\lambda \| \mathbf{e}_f^{(i)}(k) \|^2 + [e_{fq_1}^{(i)}(k+1)]^2}. \tag{4.36}$$

All other equations are combined in a single loop by computing partial results from the partial results of the previous equations. The resulting algorithm is described in detail in Appendix 3 and, although not identical, is similar to the one presented in [15]. A stage of its lattice structure is depicted in Figure 4.2, where the rotation and angle processors can be easily understood from the algorithmic description.

Finally, the lattice version of the FQR_PRI_B algorithm is obtained in a way which is very similar to the one used to derive the lattice version of the FQR_POS_B algorithm [4]. The algorithm is shown in Appendix 3 and Figure 4.3 depicts one stage of the lattice structure for this algorithm.

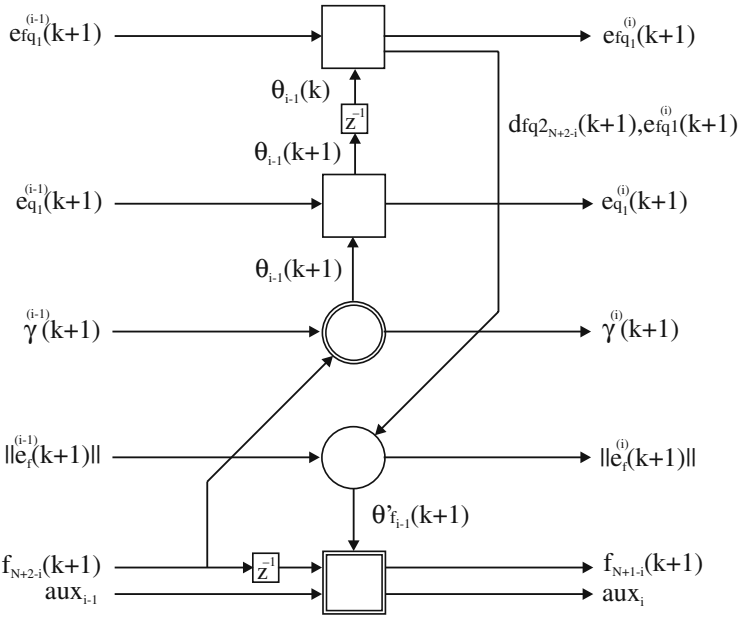

Fig. 4.2 One stage of the FQR_POS_B lattice structure.

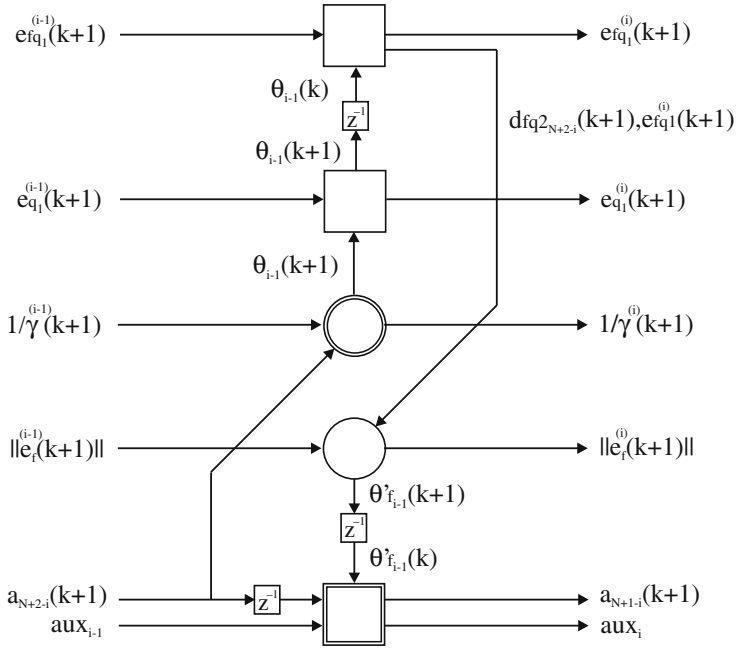

Fig. 4.3 One stage of the FQR_PRI_B lattice structure.

This section presented the order recursive or lattice versions of the fast QRD-RLS algorithms that update *a posteriori* and *a priori* backward errors. Results from the Gram–Schmidt orthogonalization can be used to conjecture that only the fast QR algorithms using lower triangularization, i.e., those updating backward prediction errors, would have their lattice versions easily implementable.

It can be observed from simulation results (not shown here) that the performance of the order recursive versions, in finite-precision implementations, is comparable to that of the original algorithms. The lattice versions have basically the same complexity as their original algorithms (FQR_POS_B and FQR_PRI_B) and lower complexity than the fast QR lattice algorithms previously proposed in [16].

4.5 Conclusion

This chapter presented the so-called fast versions of the QRD-RLS family of algorithms. In here, fast QRD-RLS algorithms using *a priori* and *a posteriori* forward and backward errors (based on upper or lower triangularization of the input data matrix) were derived. A comprehensive framework was used to classify the four fast QRD-RLS algorithms and their presentation were based on simple matrix equations, while detailed algorithmic descriptions were provided in appendices.

For those with a suitable application at hands and willing to select a fast QRD-RLS algorithm, as a finger rule, we stress the importance of those based on the updating of backward prediction errors (lower triangular algorithms as per the notation adopted in this chapter). Although they all might have similar computational complexity, the FQR_PRI_B and the FQR_POS_B algorithms present the desired feature of proven stability. The fast QRD-RLS algorithms can be used whenever the input signal vector consists of a delay line; in case where we have distinct signals each consisting of a delay line, multichannel versions are appropriate; this topic is addressed in Chapter 6.

Finally, it is worth mentioning that fast QRD-RLS algorithms do not provide the filter coefficients (transversal form) explicitly; they are however embedded in the internal variables. In applications where these weights are desirable, weight extraction techniques can be used as described in Chapter 11.

Appendix 1 - Pseudo-Code for the Fast QRD-RLS Algorithms Based on Forward Prediction Errors

1. The FQR_POS_F algorithm:

FQR_POS_F [3]

Initialization:

$\| \mathbf{e}_f(k) \| = \delta =$ small positive value; $\quad \mathbf{d}_{fq2}(k) = \mathbf{d}_{q2}(k) =$ zeros$(N+1,1)$;

$cos\theta(k) = cos\theta'_b(k) =$ ones$(N+1,1)$; $\quad sin\theta(k) = sin\theta'_b(k) =$ zeros$(N+1,1)$;

$\gamma(k) = 1$; $\quad \mathbf{f}(k) =$ zeros$(N+1,1)$;

for $k = 1, 2, \ldots$

$\{ \; e^{(0)}_{fq_1}(k+1) = x(k+1)$;

for $i = 1 : N+1$

$\{ \; e^{(i)}_{fq_1}(k+1) = cos\theta_{i-1}(k) e^{(i-1)}_{fq_1}(k) - sin\theta_{i-1}(k) \lambda^{1/2} d_{fq2_i}(k)$;

$d_{fq2_i}(k+1) = sin\theta_{i-1}(k) e^{(i-1)}_{fq_1}(k) + cos\theta_{i-1}(k) \lambda^{1/2} d_{fq2_i}(k)$;

$\}$

$e_{fq_1}(k+1) = e^{(N+1)}_{fq_1}(k+1)$;

$\| \mathbf{e}_f(k+1) \| = \sqrt{e^2_{fq_1}(k+1) + \lambda \| \mathbf{e}_f(k) \|^2}$; $cos\theta_f(k+1) = \lambda^{1/2} \| \mathbf{e}_f(k) \| / \| \mathbf{e}_f(k+1) \|$;

$sin\theta_f(k+1) = e_{fq_1}(k+1) / \| \mathbf{e}_f(k+1) \|$; $\quad \mathbf{c}(k+1) = [1;$ zeros$(N+1,1)]$;

for $i = 1 : N+1$

$\{ \; c_{N+3-i}(k+1) = -sin\theta'_{b_{N+1-i}}(k) c_1$; $\quad c_1(k+1) = cos\theta'_{b_{N+1-i}}(k) c_1$;

$\}$

$\mathbf{c}_{aux} = [0;\, c(k+1)]$;

for $i = 1 : N+1$

$\{ \; oldvalue = c_{aux_{i+1}}$;

$c_{aux_{i+1}} = sin\theta_{i-1}(k) c_{aux_1} + cos\theta_{i-1}(k) c_{aux_{i+1}}$;

$c_{aux_1} = cos\theta_{i-1}(k) c_{aux_1} - sin\theta_{i-1}(k) oldvalue$;

$\}$

$oldvalue = c_{aux_1}$;

$c_{aux_1} = cos\theta_f(k+1) c_{aux_1} - sin\theta_f(k+1) c_{aux_{N+3}}$;

$c_{aux_{N+3}} = sin\theta_f(k+1) oldvalue + cos\theta_f(k+1) c_{aux_{N+3}}$;

$\mathbf{c}(k+1) = \mathbf{c}_{aux}(2 : N+3)$;

for $i = 1 : N+1$

$\{ \; oldvalue = c_1$; $\quad c_1 = \sqrt{c_1^2 + c_{i+1}^2}$;

$cos\theta'_{b_{i-1}}(k+1) = oldvalue / c_1$; $\quad sin\theta'_{b_{i-1}}(k+1) = -c_{i+1} / c_1$;

$\}$

$\mathbf{f}^{(N+2)}(k+1) = [\mathbf{f}(k);\, sin\theta_f(k+1)\gamma(k)]$; $\quad aux_0 = f_1^{(N+2)}(k+1)$;

for $i = 1 : N+1$

$\{ \; aux_i = cos\theta'_{b_{i-1}}(k+1) aux_{i-1} - sin\theta'_{b_{i-1}}(k+1) f_{i+1}^{(N+2)}(k+1)$;

$f_i(k+1) = sin\theta'_{b_{i-1}}(k+1) aux_{i-1} + cos\theta'_{b_{i-1}}(k+1) f_{i+1}^{(N+2)}(k+1)$;

$\}$

$\gamma^{(0)}(k+1) = 1$;

for $i = 1 : N+1$

$\{ \; sin\theta_{i-1}(k+1) = f_i(k+1) / \gamma^{(i-1)}(k+1)$;

$cos\theta_{i-1}(k+1) = \sqrt{1 - sin^2\theta_{i-1}(k+1)}$;

$\gamma^{(i)}(k+1) = cos\theta_{i-1}(k+1) \gamma^{(i-1)}(k+1)$;

$\}$

$\gamma(k+1) = \gamma^{(N+1)}(k+1)$; $\quad e^{(0)}_{q_1}(k+1) = d(k+1)$;

for $i = 1 : N+1$

$\{ \; e^{(i)}_{q_1}(k+1) = cos\theta_{i-1}(k+1) e^{(i-1)}_{q_1}(k+1) - sin\theta_{i-1}(k+1) \lambda^{1/2} d_{q2_i}(k)$;

$d_{q2_i}(k+1) = sin\theta_{i-1}(k+1) e^{(i-1)}_{q_1}(k+1) + cos\theta_{i-1}(k+1) \lambda^{1/2} d_{q2_i}(k)$;

$\}$

$e_{q_1}(k+1) = e^{(N+1)}_{q_1}(k+1)$; $\quad e(k+1) = e_{q_1}(k+1) / \gamma(k+1)$;

$\}$

2. The FQR_PRI_F algorithm:

FQR_PRI_F [7]

Initialization:
$\| \mathbf{e}_f(k) \| = \delta$ = small positive value; $\mathbf{d}_{fq2}(k) = \mathbf{d}_{q2}(k) = \text{zeros}(N+1,1)$;
$cos\theta(k) = cos\theta'_b(k) = \text{ones}(N+1,1)$; $sin\theta(k) = sin\theta'_b(k) = \text{zeros}(N+1,1)$;
$1/\gamma(k) = 1$; $\mathbf{a}(k) = \text{zeros}(N+1,1)$;
for $k = 1,2,\ldots$
{ $e^{(0)}_{fq_1}(k+1) = x(k+1)$;
for $i = 1 : N+1$
{ $e^{(i)}_{fq_1}(k+1) = cos\theta_{i-1}(k) e^{(i-1)}_{fq_1}(k) - sin\theta_{i-1}(k)\lambda^{1/2} d_{fq2_i}(k)$;
$d_{fq2_i}(k+1) = sin\theta_{i-1}(k) e^{(i-1)}_{fq_1}(k) + cos\theta_{i-1}(k)\lambda^{1/2} d_{fq2_i}(k)$;
}
$e_{fq_1}(k+1) = e^{(N+1)}_{fq_1}(k+1)$; $e_f(k+1) = e_{fq_1}(k+1)[1/\gamma(k)]$;
$\mathbf{a}^{(N+2)}(k+1) = [\mathbf{a}(k);\ \frac{e_f(k+1)}{\lambda^{1/2}\|\mathbf{e}_f(k)\|}]$; $aux_0 = a^{(N+2)}_1(k+1)$;
for $i = 1 : N+1$
{ $aux_i = cos\theta'_{b_{i-1}}(k) aux_{i-1} - sin\theta'_{b_{i-1}}(k) a^{(N+2)}_{i+1}(k+1)$;
$a_i(k+1) = sin\theta'_{b_{i-1}}(k) aux_{i-1} + cos\theta'_{b_{i-1}}(k) a^{(N+2)}_{i+1}(k+1)$;
}
$\| \mathbf{e}_f(k+1) \| = \sqrt{e^2_{fq_1}(k+1) + \lambda \| \mathbf{e}_f(k) \|^2}$;
$cos\theta_f(k+1) = \lambda^{1/2} \| \mathbf{e}_f(k) \| / \| \mathbf{e}_f(k+1) \|$;
$sin\theta_f(k+1) = e_{fq_1}(k+1) / \| \mathbf{e}_f(k+1) \|$
$\mathbf{c}(k+1) = [1; \text{zeros}(N+1,1)]$;
for $i = 1 : N+1$
{ $c_{N+3-i}(k+1) = -sin\theta'_{b_{N+1-i}}(k) c_1$;
$c_1(k+1) = cos\theta'_{b_{N+1-i}}(k) c_1$;
}
$\mathbf{c}_{aux} = [0;\ c(k+1)]$;
for $i = 1 : N+1$
{ $oldvalue = c_{aux_{i+1}}$;
$c_{aux_{i+1}} = sin\theta_{i-1}(k) c_{aux_1} + cos\theta_{i-1}(k) c_{aux_{i+1}}$;
$c_{aux_1} = cos\theta_{i-1}(k) c_{aux_1} - sin\theta_{i-1}(k) oldvalue$;
}
$oldvalue = c_{aux_1}$;
$c_{aux_1} = cos\theta_f(k+1) c_{aux_1} - sin\theta_f(k+1) c_{aux_{N+3}}$;
$c_{aux_{N+3}} = sin\theta_f(k+1) oldvalue + cos\theta_f(k+1) c_{aux_{N+3}}$;
$\mathbf{c}(k+1) = c_{aux}(2 : N+3)$;
for $i = 1 : N+1$
{ $oldvalue = c_1$; $c_1 = \sqrt{c_1^2 + c_{i+1}^2}$;
$cos\theta'_{b_{i-1}}(k+1) = oldvalue / c_1$; $sin\theta'_{b_{i-1}}(k+1) = -c_{i+1}/c_1$;
}
$1/\gamma^{(0)}(k+1) = 1$;
for $i = 1 : N+1$
{ $1/\gamma^{(i)}(k+1) = \sqrt{[1/\gamma^{(i-1)}(k+1)]^2 + a_i^2(k+1)}$;
$cos\theta_{i-1}(k+1) = \frac{1/\gamma^{(i-1)}(k+1)}{1/\gamma^{(i)}(k+1)}$;
$sin\theta_{i-1}(k+1) = \frac{a_i(k+1)}{1/\gamma^{(i)}(k+1)}$;
}
$\gamma(k+1) = 1/[1/\gamma^{(N+1)}(k+1)]$; $e^{(0)}_{q_1}(k+1) = d(k+1)$;
for $i = 1 : N+1$
{ $e^{(i)}_{q_1}(k+1) = cos\theta_{i-1}(k+1) e^{(i-1)}_{q_1}(k+1) - sin\theta_{i-1}(k+1)\lambda^{1/2} d_{q2_i}(k)$;
$d_{q2_i}(k+1) = sin\theta_{i-1}(k+1) e^{(i-1)}_{q_1}(k+1) + cos\theta_{i-1}(k+1)\lambda^{1/2} d_{q2_i}(k)$;
}
$e_{q_1}(k+1) = e^{(N+1)}_{q_1}(k+1)$; $e(k+1) = e_{q_1}(k+1)/\gamma(k+1)$;
}

Appendix 2 - Pseudo-Code for the Fast QRD-RLS Algorithms Based on Backward Prediction Errors

1. The FQR_POS_B algorithm:

The first version of this algorithm is based on the fact that the last element of $\mathbf{f}(k+1)$, $f_{N+1}(k+1) = \frac{x(k+1)}{\|\mathbf{e}_f^{(0)}(k+1)\|}$, is known in advance.

FQR_POS_B - Version 1 [6]

Initialization:

$\mathbf{d}_{fq2}(k) = \text{zeros}(N+1,1)$; $\mathbf{d}_{q2}(k) = \text{zeros}(N+1,1)$;

$cos\theta(k) = \text{ones}(N+1,1)$; $sin\theta(k) = \text{zeros}(N+1,1)$;

$\| \mathbf{e}_f(k) \| = \delta = $ small positive value; $\mathbf{f}(k) = \text{zeros}(N+1,1)$;

for $k = 1,2,\ldots$

$\{\ e_{fq_1}^{(0)}(k+1) = x^*(k+1)$;

for $i = 1 : N+1$

$\{\ e_{fq_1}^{(i)}(k+1) = cos\theta_{i-1}(k) e_{fq_1}^{(i-1)}(k+1) - sin^*\theta_{i-1}(k)\lambda^{1/2} d_{fq2_{N+2-i}}(k)$;

$d_{fq2_{N+2-i}}(k+1) = sin\theta_{i-1}(k) e_{fq_1}^{(i-1)}(k+1) + cos\theta_{i-1}(k)\lambda^{1/2} d_{fq2_{N+2-i}}(k)$;

$\}$

$e_{fq_1}(k+1) = e_{fq_1}^{(N+1)}(k+1)$; $\| \mathbf{e}_f(k+1) \| = \sqrt{|e_{fq_1}(k+1)|^2 + \lambda \| \mathbf{e}_f(k) \|^2}$;

$\| \mathbf{e}_f^{(N+1)}(k+1) \| = \| \mathbf{e}_f(k+1) \|$;

for $i = 1 : N+1$

$\{\ \| \mathbf{e}_f^{(N+1-i)}(k+1) \| = \sqrt{\| \mathbf{e}_f^{(N+2-i)}(k+1) \|^2 + |d_{fq2_i}(k+1)|^2}$;

$cos\theta'_{f_i}(k+1) = \| \mathbf{e}_f^{(N+2-i)}(k+1) \| \ / \ \| \mathbf{e}_f^{(N+1-i)}(k+1) \|$;

$sin\theta'_{f_i}(k+1) = d^*_{fq2_i}(k+1) / \| \mathbf{e}_f^{(N+1-i)}(k+1) \|$;

$\}$

$aux_0 = x(k+1) / \| \mathbf{e}_f^{(0)}(k+1) \|$; $f_{N+1}(k+1) = aux_0$;

for $i = 1 : N$

$\{\ f_{N+1-i}(k+1) = \frac{f_{N+2-i}(k) - sin\theta'^*_{f_{N+1-i}}(k+1) aux_{i-1}}{cos\theta'_{f_{N+1-i}}(k+1)}$;

$aux_i = -sin\theta'_{f_{N+1-i}}(k+1) f_{N+1-i}(k) + cos\theta'_{f_{N+1-i}}(k+1) aux_{i-1}$;

$\}$

$\gamma^{(0)}(k+1) = 1$;

for $i = 1 : N+1$

$\{\ sin\theta_{i-1}(k+1) = f_{N+2-i}(k+1) / \gamma^{(i-1)}(k+1)$;

$cos\theta_{i-1}(k+1) = \sqrt{1 - |sin\theta_{i-1}(k+1)|^2}$;

$\gamma^{(i)}(k+1) = cos\theta_{i-1}(k+1)\gamma^{(i-1)}(k+1)$;

$\}$

$\gamma(k+1) = \gamma^{(N+1)}(k+1)$; $e_{q_1}^{(0)}(k+1) = d^*(k+1)$;

for $i = 1 : N+1$

$\{\ e_{q_1}^{(i)}(k+1) = cos\theta_{i-1}(k+1) e_{q_1}^{(i-1)}(k+1) - sin^*\theta_{i-1}(k+1)\lambda^{1/2} d_{q2_{N+2-i}}(k)$;

$d_{q2_{N+2-i}}(k+1) = sin\theta_{i-1}(k+1) e_{q_1}^{(i-1)}(k+1) + cos\theta_{i-1}(k+1)\lambda^{1/2} d_{q2_{N+2-i}}(k)$;

$\}$

$e_{q_1}(k+1) = e_{q_1}^{(N+1)}(k+1)$; $e(k+1) = e_{q_1}^*(k+1)/\gamma(k+1)$;

$\}$

The second version of the FQR_POS_B algorithm is based on the straightforward computation of $\mathbf{f}(k+1)$ according to (4.20) and requires the calculation of $\frac{e_f(k+1)}{\|\mathbf{e}_f(k+1)\|}$.

FQR_POS_B - Version 2 [5]

Initialization:

$\mathbf{d}_{fq2}(k) = \text{zeros}(N+1,1)$; $\mathbf{d}_{q2}(k) = \text{zeros}(N+1,1)$;

$cos\theta(k) = \text{ones}(N+1,1)$; $sin\theta(k) = \text{zeros}(N+1,1)$;

$\| \mathbf{e}_f(k) \| = \delta = $ small positive value; $\mathbf{f}(k) = \text{zeros}(N+1,1)$; $\gamma(k) = 1$

for $k = 1, 2, \ldots$

$\{ \; e_{fq_1}^{(0)}(k+1) = x^*(k+1)$;

for $i = 1 : N+1$

$\{ \; e_{fq_1}^{(i)}(k+1) = cos\theta_{i-1}(k) e_{fq_1}^{(i-1)}(k+1) - sin^*\theta_{i-1}(k)\lambda^{1/2} d_{fq2_{N+2-i}}(k)$;

$d_{fq2_{N+2-i}}(k+1) = sin\theta_{i-1}(k) e_{fq_1}^{(i-1)}(k+1) + cos\theta_{i-1}(k)\lambda^{1/2} d_{fq2_{N+2-i}}(k)$;

$\}$

$e_{fq_1}(k+1) = e_{fq_1}^{(N+1)}(k+1)$;

$\| \mathbf{e}_f(k+1) \| = \sqrt{|e_{fq_1}(k+1)|^2 + \lambda \| \mathbf{e}_f(k) \|^2}$;

$\| \mathbf{e}_f^{(N+1)}(k+1) \| = \| \mathbf{e}_f(k+1) \|$;

for $i = 1 : N+1$

$\{ \; \| \mathbf{e}_f^{(N+1-i)}(k+1) \| = \sqrt{\| \mathbf{e}_f^{(N+2-i)}(k+1) \|^2 + |d_{fq2_i}(k+1)|^2}$;

$cos\theta'_{f_{N+1-i}}(k+1) = \| \mathbf{e}_f^{(N+2-i)}(k+1) \| \; / \; \| \mathbf{e}_f^{(N+1-i)}(k+1) \|$;

$sin\theta'_{f_{N+1-i}}(k+1) = d^*_{fq2_i}(k+1) / \| \mathbf{e}_f^{(N+1-i)}(k+1) \|$;

$\}$

$aux_0 = \frac{\gamma(k) e^*_{fq_1}(k+1)}{\|\mathbf{e}_f(k+1)\|}$;

for $i = 1 : N+1$

$\{ \; f_{i-1}(k+1) = cos\theta'_{f_{N+1-i}}(k+1) f_i(k) - sin\theta'^*_{f_{N+1-i}}(k+1) aux_{i-1}$;

$aux_i = sin\theta'_{f_{N+1-i}}(k+1) f_i(k) + cos\theta'_{f_{N+1-i}}(k+1) aux_{i-1}$;

$\}$

$\frac{\varepsilon_b(k+1)}{\|\mathbf{e}_b(k+1)\|} = f_0(k+1)$;

$f_{N+1}(k+1) = aux_{N+1}$;

$\gamma^{(0)}(k+1) = 1$;

for $i = 1 : N+1$

$\{ \; sin\theta_{i-1}(k+1) = f_{N+2-i}(k+1) / \gamma^{(i-1)}(k+1)$;

$cos\theta_{i-1}(k+1) = \sqrt{1 - |sin\theta_{i-1}(k+1)|^2}$;

$\gamma^{(i)}(k+1) = cos\theta_{i-1}(k+1) \gamma^{(i-1)}(k+1)$;

$\}$

$\gamma(k+1) = \gamma^{(N+1)}(k+1)$;

$e_{q_1}^{(0)}(k+1) = d^*(k+1)$;

for $i = 1 : N+1$

$\{ \; e_{q_1}^{(i)}(k+1) = cos\theta_{i-1}(k+1) e_{q_1}^{(i-1)}(k+1) - sin^*\theta_{i-1}(k+1)\lambda^{1/2} d_{q2_{N+2-i}}(k)$;

$d_{q2_{N+2-i}}(k+1) = sin\theta_{i-1}(k+1) e_{q_1}^{(i-1)}(k+1) + cos\theta_{i-1}(k+1)\lambda^{1/2} d_{q2_{N+2-i}}(k)$;

$\}$

$e_{q_1}(k+1) = e_{q_1}^{(N+1)}(k+1)$; $e(k+1) = e^*_{q_1}(k+1) / \gamma(k+1)$;

$\}$

2. The FQR_PRI_B algorithm:

The first version of this algorithm is based on the fact that the last element of $\mathbf{a}(k+1)$ (or $a_{N+1}(k+1) = \frac{x(k+1)}{\sqrt{\lambda}\|\mathbf{e}_f^{(0)}(k)\|}$) is known in advance.

FQR_PRI_B - Version 1 [4]

Initialization:

$\| \mathbf{e}_f^{(0)}(k) \| = \| \mathbf{e}_f(k) \| = \delta$ = small positive value; $\mathbf{a}(k) = \text{zeros}(N+1,1)$;

$\mathbf{d}_{fq2}(k) = \text{zeros}(N+1,1)$; $\mathbf{d}_{q2}(k) = \text{zeros}(N+1,1)$;

$cos\theta(k) = \text{ones}(N+1,1)$; $cos\theta'_f(k) = \text{ones}(N+1,1)$;

$sin\theta(k) = \text{zeros}(N+1,1)$; $sin\theta'_f(k) = \text{zeros}(N+1,1)$;

for $k = 1, 2, \ldots$

$\{\ e_{fq_1}^{(0)}(k+1) = x^*(k+1)$;

for $i = 1 : N+1$

$\{\ e_{fq_1}^{(i)}(k+1) = cos\theta_{i-1}(k) e_{fq_1}^{(i-1)}(k+1) - sin^*\theta_{i-1}(k)\lambda^{1/2} d_{fq2_{N+2-i}}(k)$;

$d_{fq2_{N+2-i}}(k+1) = sin\theta_{i-1}(k) e_{fq_1}^{(i-1)}(k+1) + cos\theta_{i-1}(k)\lambda^{1/2} d_{fq2_{N+2-i}}(k)$;

$\}$

$e_{fq_1}(k+1) = e_{fq_1}^{(N+1)}(k+1)$;

$aux_0 = \frac{x^*(k+1)}{\lambda^{1/2}\|\mathbf{e}_f^{(0)}(k)\|}$; $a_{N+1}(k+1) = aux_0$;

for $i = 1 : N$

$\{\ a_{N+1-i}(k+1) = \frac{a_{N+2-i}(k) - sin\theta'_{f_{i-1}}(k) aux_{i-1}}{cos\theta'_{f_{i-1}}(k)}$;

$aux_i = -sin\theta'^*_{f_{i-1}}(k) a_{N+1-i}(k+1) + cos\theta'_{f_{i-1}}(k) aux_{i-1}$;

$\}$

$\| \mathbf{e}_f^{(N+1)}(k+1) \| = \| \mathbf{e}_f(k+1) \| = \sqrt{|e_{fq_1}(k+1)|^2 + \lambda \| \mathbf{e}_f(k) \|^2}$;

for $i = 1 : N+1$

$\{\ \| \mathbf{e}_f^{(N+1-i)}(k+1) \| = \sqrt{\| \mathbf{e}_f^{(N+2-i)}(k+1) \|^2 + |d_{fq2_i}(k+1)|^2}$;

$cos\theta'_{f_{N+1-i}}(k+1) = \| \mathbf{e}_f^{(N+2-i)}(k+1) \| / \| \mathbf{e}_f^{(N+1-i)}(k+1) \|$;

$sin\theta'_{f_{N+1-i}}(k+1) = d^*_{fq2_i}(k+1) / \| \mathbf{e}_f^{(N+1-i)}(k+1) \|$;

$\}$

$1/\gamma^{(0)}(k+1) = 1$;

for $i = 1 : N+1$

$\{\ 1/\gamma^{(i)}(k+1) = \sqrt{[1/\gamma^{(i-1)}(k+1)]^2 + |a_{N+2-i}(k+1)|^2}$

$cos\theta_{i-1}(k+1) = \frac{1/\gamma^{(i-1)}(k+1)}{1/\gamma^{(i)}(k+1)}$;

$sin\theta_{i-1}(k+1) = \frac{a^*_{N+2-i}(k+1)}{1/\gamma^{(i)}(k+1)}$;

$\}$

$\gamma(k+1) = 1/[1/\gamma^{(N+1)}(k+1)]$; $e_{q_1}^{(0)}(k+1) = d^*(k+1)$;

for $i = 1 : N+1$

$\{\ e_{q_1}^{(i)}(k+1) = cos\theta_{i-1}(k+1) e_{q_1}^{(i-1)}(k+1) - sin^*\theta_{i-1}(k+1)\lambda^{1/2} d_{q2_{N+2-i}}(k)$;

$d_{q2_{N+2-i}}(k+1) = sin\theta_{i-1}(k+1) e_{q_1}^{(i-1)}(k+1) + cos\theta_{i-1}(k+1)\lambda^{1/2} d_{q2_{N+2-i}}(k)$;

$\}$

$e_{q_1}(k+1) = e_{q_1}^{(N+1)}(k+1)$; $e(k+1) = e^*_{q_1}(k+1)/\gamma(k+1)$;

$\}$

The second version of the FQR_PRI_B algorithm is based on the straightforward computation of $\mathbf{a}(k+1)$ according to (4.29) and requires the calculation of $\frac{e'_f(k+1)}{\sqrt{\lambda}\|\mathbf{e}_f(k)\|}$.

FQR_PRI_B - Version 2 [8]

Initialization:

$\| \mathbf{e}_f(k) \| = \delta =$ small positive value; $\mathbf{a}(k) = \text{zeros}(N+1,1)$; $\gamma(k) = 1$;

$\mathbf{d}_{fq2}(k) = \text{zeros}(N+1,1)$; $\mathbf{d}_{q2}(k) = \text{zeros}(N+1,1)$;

$cos\theta(k) = \text{ones}(N+1,1)$; $cos\theta'_f(k) = \text{ones}(N+1,1)$;

$sin\theta(k) = \text{zeros}(N+1,1)$; $sin\theta'_f(k) = \text{zeros}(N+1,1)$;

for $k = 1,2,\ldots$

$\{\ e^{(0)}_{fq_1}(k+1) = x^*(k+1)$;

for $i = 1 : N+1$

$\{\ e^{(i)}_{fq_1}(k+1) = cos\theta_{i-1}(k) e^{(i-1)}_{fq_1}(k+1) - sin^*\theta_{i-1}(k)\lambda^{1/2} d_{fq2_{N+2-i}}(k)$;

$d_{fq2_{N+2-i}}(k+1) = sin\theta_{i-1}(k) e^{(i-1)}_{fq_1}(k+1) + cos\theta_{i-1}(k)\lambda^{1/2} d_{fq2_{N+2-i}}(k)$;

$\}$

$e_{fq_1}(k+1) = e^{(N+1)}_{fq_1}(k+1)$;

$aux_0 = \frac{e_{fq_1}(k+1)}{\gamma(k)\lambda^{1/2}\|\mathbf{e}_f(k)\|}$;

for $i = 1 : N+1$

$\{\ a_{i-1}(k+1) = cos\theta'_{f_{N+1-i}}(k) a_i(k) - sin\theta'_{f_{N+1-i}}(k) aux_{i-1}$;

$aux_i = sin^*\theta'_{f_{N+1-i}}(k) a_i(k) + cos\theta'_{f_{N+1-i}}(k) aux_{i-1}$;

$\}$

$\frac{e_b(k+1)}{\lambda^{1/2}\|\mathbf{e}_b(k)\|} = a_0(k+1)$;

$a_{N+1}(k+1) = aux_{N+1}$;

$\| \mathbf{e}_f(k+1) \| = \sqrt{|e_{fq_1}(k+1)|^2 + \lambda \| \mathbf{e}_f(k) \|^2}$;

$\| \mathbf{e}^{(N+1)}_f(k+1) \| = \| \mathbf{e}_f(k+1) \|$;

for $i = 1 : N+1$

$\{\ \| \mathbf{e}^{(N+1-i)}_f(k+1) \| = \sqrt{\| \mathbf{e}^{(N+2-i)}_f(k+1) \|^2 + |d_{fq2_i}(k+1)|^2}$;

$cos\theta'_{f_{N+1-i}}(k+1) = \| \mathbf{e}^{(N+2-i)}_f(k+1) \| / \| \mathbf{e}^{(N+1-i)}_f(k+1) \|$;

$sin\theta'_{f_{N+1-i}}(k+1) = d^*_{fq2_i}(k+1) / \| \mathbf{e}^{(N+1-i)}_f(k+1) \|$;

$\}$

$1/\gamma^{(0)}(k+1) = 1$;

for $i = 1 : N+1$

$\{\ 1/\gamma^{(i)}(k+1) = \sqrt{[1/\gamma^{(i-1)}(k+1)]^2 + |a_{N+2-i}(k+1)|^2}$;

$cos\theta_{i-1}(k+1) = \frac{1/\gamma^{(i-1)}(k+1)}{1/\gamma^{(i)}(k+1)}$;

$sin\theta_{i-1}(k+1) = \frac{a^*_{N+2-i}(k+1)}{1/\gamma^{(i)}(k+1)}$;

$\}$

$\gamma(k+1) = 1/[1/\gamma^{(N+1)}(k+1)]$; $e^{(0)}_{q_1}(k+1) = d^*(k+1)$;

for $i = 1 : N+1$

$\{\ e^{(i)}_{q_1}(k+1) = cos\theta_{i-1}(k+1) e^{(i-1)}_{q_1}(k+1) - sin^*\theta_{i-1}(k+1)\lambda^{1/2} d_{q2_{N+2-i}}(k)$;

$d_{q2_{N+2-i}}(k+1) = sin\theta_{i-1}(k+1) e^{(i-1)}_{q_1}(k+1) + cos\theta_{i-1}(k+1)\lambda^{1/2} d_{q2_{N+2-i}}(k)$;

$\}$

$e_{q_1}(k+1) = e^{(N+1)}_{q_1}(k+1)$; $e(k+1) = e^*_{q_1}(k+1)/\gamma(k+1)$;

$\}$

Appendix 3 - Pseudo-Code for the Order Recursive FQRD-RLS Algorithms

1. Order recursive version of the FQR_POS_B algorithm:

LATTICE FQR_POS_B

Soft-constrained initialization:
ε = small positive value;
for $i = 0 : N+1$
$\{ \parallel e_f^{(i)}(k) \parallel = \varepsilon;$
$\}$
$d_{fq2}(k) = \text{zeros}(N+1,1);$ $d_{q2}(k) = \text{zeros}(N+1,1);$
$cos\theta(k) = \text{ones}(N+1,1);$ $sin\theta(k) = \text{zeros}(N+1,1);$
$f(k) = \text{zeros}(N+1,1);$
for each k
$\{ \; e_{fq_1}^{(0)}(k+1) = x(k+1);$
$\parallel e_f^{(0)}(k+1) \parallel = \sqrt{[e_{fq_1}^{(0)}(k+1)]^2 + \lambda \parallel e_f^{(0)}(k) \parallel^2};$
$aux_0 = \frac{x(k+1)}{\parallel e_f^{(0)}(k+1) \parallel};$
$f_{N+1}(k+1) = aux_0;$
$\gamma^{(0)}(k+1) = 1;$
$e_{q_1}^{(0)}(k+1) = d(k+1);$
for $i = 1 : N+1$
$\{ d_{fq2_{N+2-i}}(k+1) = sin\theta_{i-1}(k) e_{fq_1}^{(i-1)}(k+1) +$
$cos\theta_{i-1}(k)\lambda^{1/2} d_{fq2_{N+2-i}}(k);$
$e_{fq_1}^{(i)}(k+1) = cos\theta_{i-1}(k) e_{fq_1}^{(i-1)}(k+1) -$
$sin\theta_{i-1}(k)\lambda^{1/2} d_{fq2_{N+2-i}}(k);$
$\parallel e_f^{(i)}(k+1) \parallel = \sqrt{[e_{fq_1}^{(i)}(k+1)]^2 + \lambda \parallel e_f^{(i)}(k) \parallel^2};$
$cos\theta'_{f_{i-1}}(k+1) = \parallel e_f^{(i)}(k+1) \parallel / \parallel e_f^{(i-1)}(k+1) \parallel;$
$sin\theta'_{f_{i-1}}(k+1) = d_{fq2_{N+2-i}}(k+1) / \parallel e_f^{(i-1)}(k+1) \parallel;$
$f_{N+1-i}(k+1) = \frac{f_{N+2-i}(k) - sin\theta'_{f_{i-1}}(k+1) aux_{i-1}}{cos\theta'_{f_{i-1}}(k+1)};$
$aux_i = -sin\theta'_{f_{i-1}}(k+1) f_{N+1-i}(k+1) + cos\theta'_{f_{i-1}}(k+1) aux_{i-1};$
$\gamma^{(i)}(k+1) = \sqrt{[\gamma^{(i-1)}(k+1)]^2 - [f_{N+2-i}(k+1)]^2};$
$cos\theta_{i-1}(k+1) = \frac{\gamma^{(i)}(k+1)}{\gamma^{(i-1)}(k+1)};$
$sin\theta_{i-1}(k+1) = \frac{f_{N+2-i}(k+1)}{\gamma^{(i-1)}(k+1)};$
$dq2_{N+2-i}(k+1) = sin\theta_{i-1}(k+1) e_{q1}^{(i-1)}(k+1) +$
$cos\theta_{i-1}(k+1)\lambda^{1/2} d_{q2_{N+2-i}}(k);$
$e_{q_1}^{(i)}(k+1) = cos\theta_{i-1}(k+1) e_{q_1}^{(i-1)}(k+1) -$
$sin\theta_{i-1}(k+1)\lambda^{1/2} d_{q2_{N+2-i}}(k);$
$\}$
$e(k+1) = e_{q_1}^{(N+1)}(k+1) / \gamma^{(N+1)}(k+1);$
$\}$

2. Order recursive version of the FQR_PRI_B algorithm:

LATTICE FQR_PRI_B

Soft-constrained initialization:

ε = small positive value;

for $i = 0 : N+1$

$\{ \parallel e_f^{(i)}(k) \parallel = \varepsilon;$

$\}$

$d_{fq2}(k) = \text{zeros}(N+1,1);$ $\quad d_{q2}(k) = \text{zeros}(N+1,1);$

$cos\theta(k) = \text{ones}(N+1,1);$ $\quad cos\theta'_f(k) = \text{ones}(N+1,1);$

$sin\theta(k) = \text{zeros}(N+1,1);$ $\quad sin\theta'_f(k) = \text{zeros}(N+1,1);$

$a(k) = \text{zeros}(N+1,1);$

for each k

$\{ \; aux_0 = \frac{x(k+1)}{\sqrt{\lambda}\parallel e_f^{(0)}(k)\parallel};$

$a_{N+1}(k+1) = aux_0;$

$e_{fq_1}^{(0)}(k+1) = x(k+1);$

$\parallel e_f^{(0)}(k+1) \parallel = \sqrt{[e_{fq_1}^{(0)}(k+1)]^2 + \lambda \parallel e_f^{(0)}(k) \parallel^2};$

$1/\gamma^{(0)}(k+1) = 1;$

$e_{q_1}^{(0)}(k+1) = d(k+1);$

for $i = 1 : N+1$

$\{ \; a_{N+1-i}(k+1) = \frac{a_{N+2-i}(k) - sin\theta'_{f_{i-1}}(k) aux_{i-1}}{cos\theta'_{f_{i-1}}(k)};$

$aux_i = -sin\theta'_{f_{i-1}}(k) a_{N+1-i}(k+1) + cos\theta'_{f_{i-1}}(k) aux_{i-1};$

$d_{fq2_{N+2-i}}(k+1) = sin\theta_{i-1}(k) e_{fq_1}^{(i-1)}(k+1) +$
$cos\theta_{i-1}(k)\lambda^{1/2} d_{fq2_{N+2-i}}(k);$

$e_{fq_1}^{(i)}(k+1) = cos\theta_{i-1}(k) e_{fq_1}^{(i-1)}(k+1) -$
$sin\theta_{i-1}(k)\lambda^{1/2} d_{fq2_{N+2-i}}(k);$

$\parallel e_f^{(i)}(k+1) \parallel = \sqrt{[e_{fq_1}^{(i)}(k+1)]^2 + \lambda \parallel e_f^{(i)}(k) \parallel^2};$

$cos\theta'_{f_{i-1}}(k+1) = \parallel e_f^{(i)}(k+1) \parallel / \parallel e_f^{(i-1)}(k+1) \parallel;$

$sin\theta'_{f_{i-1}}(k+1) = d_{fq2_{N+2-i}}(k+1) / \parallel e_f^{(i-1)}(k+1) \parallel;$

$1/\gamma^{(i)}(k+1) = \sqrt{[1/\gamma^{(i-1)}(k+1)]^2 + [a_{N+2-i}(k+1)]^2};$

$cos\theta_{i-1}(k+1) = \frac{1/\gamma^{(i-1)}(k+1)}{1/\gamma^{(i)}(k+1)};$

$sin\theta_{i-1}(k+1) = \frac{a_{N+2-i}(k+1)}{1/\gamma^{(i)}(k+1)};$

$dq2_{N+2-i}(k+1) = sin\theta_{i-1}(k+1) e_{q1}^{(i-1)}(k+1) +$
$cos\theta_{i-1}(k+1)\lambda^{1/2} d_{q2_{N+2-i}}(k);$

$e_{q_1}^{(i)}(k+1) = cos\theta_{i-1}(k+1) e_{q_1}^{(i-1)}(k+1) -$
$sin\theta_{i-1}(k+1)\lambda^{1/2} d_{q2_{N+2-i}}(k);$

$\}$

$e(k+1) = e_{q_1}^{(N+1)}(k+1)[1/\gamma^{(N+1)}(k+1)];$

$\}$

References

1. S. Haykin, Adaptive Filter Theory. 2nd edition Prentice-Hall, Englewood Cliffs, NJ, USA (1991)
2. P. S. R. Diniz, Adaptive Filtering: Algorithms and Practical Implementation. 3rd edition Springer, New York, NY, USA (2008)
3. J. M. Cioffi, The fast adaptive ROTOR's RLS algorithm. IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. ASSP-38, no. 4, pp. 631–653 (April 1990)
4. A. A. Rontogiannis and S. Theodoridis, New fast inverse QR least squares adaptive algorithms. IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP'95, Detroit, USA, pp. 1412–1415 (May 1995)
5. 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. SP-39, no. 4, pp. 879–891 (April 1991)
6. J. A. Apolinário Jr., M. G. Siqueira, and P. S. R. Diniz, On fast QR algorithms based on backward prediction errors: New results and comparisons. First Balkan Conference on Signal Processing, Communications, Circuits, and Systems, Istanbul, Turkey (June 2000)
7. J. A. Apolinário Jr. and P. S. R. Diniz, A new fast QR algorithm based on a priori errors. IEEE Signal Processing Letters, vol. 4, no. 11, pp. 307–309 (November 1997)
8. M. D. Miranda and M. Gerken, A hybrid QR-lattice least squares algorithm using a priori errors. 38th Midwest Symposium on Circuits and Systems, MWSCAS'95, Rio de Janeiro, Brazil, pp. 983–986 (August 1995)
9. S. T. Alexander and A. L. Ghirnikar, A method for recursive least squares adaptive filtering based upon an inverse QR decomposition. IEEE Transactions on Signal Processing, vol. SP-41, no. 1, pp. 20–30 (January 1993)
10. P. G. Park and T. Kailath, A lattice algorithm dual to the extended inverse QR algorithm. Signal Processing, vol. 47, no. 2, pp. 115–133 (November 1995)
11. M. D. Miranda and M. Gerken, A hybrid least squares QR-lattice algorithm using a priori errors. IEEE Transactions on Signal Processing, vol. 45, no. 12, pp. 2900–2911 (December 1997)
12. M. G. Bellanger, The FLS-QR algorithm for adaptive filtering. Signal Processing, vol. 17, no. 4, pp. 291–304 (August 1989)
13. J. A. Apolinário Jr., *New algorithms of adaptive filtering: LMS with data-reusing and fast RLS based on QR decomposition*. DSc thesis, UFRJ - Federal University of Rio de Janeiro, Rio de Janeiro, Brazil (1998)
14. Maria D. Miranda, *Sobre algoritmos dos mínimos quadrados rápidos, recursivos na ordem, que utilizam triangularização ortogonal* (in Portuguese). PhD thesis, USP - University of São Paulo, São Paulo, Brazil (1996)
15. M. Terré and M. Bellanger, A systolic QRD-based algorithm for adaptive filtering and its implementation. IEEE International Conference on Acoustic, Speech, and Signal Processing, ICASSP'94, Adelaide, Australia, pp. 373–376 (April 1994)
16. I. K. Proudler, J. G. McWhirter, and T. J. Shepard, Computationally efficient QR decomposition approach to least squares adaptive filtering. IEE Proceedings-F, vol. 138, no. 4, pp. 341–353 (August 1991)