Circuits Systems Signal Processing
Vol. 22, No. 4, 2003, Pp. 335–349

Fast QR Algorithms Based on Backward Prediction Errors: A New Implementation and Its Finite Precision Performance*

José Antonio Apolinário Jr.,[1] *Marcio G. Siqueira,*[2] *and Paulo S. R. Diniz*[2]

Abstract. QR decomposition techniques are well known for their good numerical behavior and low complexity. Fast QRD recursive least squares adaptive algorithms benefit from these characteristics to offer robust and fast adaptive filters. This paper examines two different versions of the fast QR algorithm based on a priori backward prediction errors as well as two other corresponding versions of the fast QR algorithm based on a posteriori backward prediction errors. The main matrix equations are presented with different versions derived from two distinct deployments of a particular matrix equation. From this study, a new algorithm is derived. The discussed algorithms are compared, and differences in computational complexity and in finite-precision behavior are shown.

Key words: Adaptive system, fast RLS algorithm, QR decomposition.

1. Introduction

Fast recursive least-squares algorithms based on QR decomposition (FQR-RLS algorithms) are widely used in adaptive filtering because of their numerical robustness and their regular structure, which can lead to efficient implementations.

Starting from the conventional (or $O[N^2]$) QR decomposition method [5], a number of fast algorithms ($O[N]$) were derived [4], [8], [6], [9], [1]. It was shown in [1] that 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) used in the updating process. It can be seen from the Gram-Schmidt orthogonalization procedure that an upper triangularization (in

* Received July 10, 2000; revised September 18, 2002.

[1] Military Institute of Engineering-DE/3, Praça General Tibúrcio, 80, CEP 22290-270, Rio de Janeiro, RJ, Brazil. E-mail: apolin@ieee.org

[2] COPPE/Federal University of Rio de Janeiro-PEE, P. O. Box 68504, CEP 21945-970, Rio de Janeiro, RJ, Brazil. E-mail: mgs@ieee.org/diniz@lps.ufrj.br

Table 1. Classification of the fast QR algorithms

Error	Prediction	
Type	Forward	Backward
A Posteriori	FQR_POS_F [4]	FQR_POS_B [8]
A Priori	FQR_PRI_F [1]	FQR_PRI_B [6], [9]

the notation adopted in this work) updates the forward prediction errors, whereas, lower triangularization updates backward prediction errors. Table 1 presents the classification used in [1] and introduces how these algorithms will be designated hereafter.

In this paper, we propose a general framework to classify backward prediction-based algorithms, which leads to the derivation of a new algorithm. Our motivation is that from a number of algorithms of the QR family, the fast QR algorithms using backward prediction errors are known to be stable. Moreover, they present lower computational complexity compared to those based on forward prediction errors and hence are worth investigating.

This paper is organized as follows. In Section 2, we present the matrix equations of the two basic FQR algorithms under investigation. Section 3 examines the two different versions of each algorithm. Section 4 discusses the computational requirements. The simulation results are shown in Section 5, and the conclusions are summarized in Section 6. A complete description of the new algorithm is presented in the Appendix.

2. FQR algorithms based on backward prediction errors

We start by reviewing the basic concepts of the conventional QR algorithm. The RLS algorithms minimize $\xi(k) = \sum_{i=0}^{k} \lambda^{k-i} e^2(i) = \boldsymbol{e}^T(k)\boldsymbol{e}(k) = \| \boldsymbol{e}(k) \|^2$, where each component of the vector $\boldsymbol{e}(k)$ is the a posteriori error at instant i weighted by $\lambda^{(k-i)/2}$ (λ being the forgetting factor). Vector $\boldsymbol{e}(k)$ is given by

$$\boldsymbol{e}(k) = \boldsymbol{d}(k) - \boldsymbol{X}(k)\boldsymbol{w}(k). \tag{1}$$

In (1), $\boldsymbol{d}(k)$ is the weighted reference or desired signal vector, $\boldsymbol{X}(k)$ is the $(k+1) \times (N+1)$ weighted input data matrix, and $\boldsymbol{w}(k)$ is the coefficient vector. These vectors and matrices are defined as follows:

$$\boldsymbol{d}(k) = \begin{bmatrix} d(k) \\ \lambda^{1/2} d(k-1) \\ \vdots \\ \lambda^{k/2} d(0) \end{bmatrix} \tag{2}$$

$$\boldsymbol{w}(k) = \begin{bmatrix} w_0(k) \\ w_1(k) \\ \vdots \\ w_N(k) \end{bmatrix} \tag{3}$$

$$\boldsymbol{X}(k) = \begin{bmatrix} \boldsymbol{x}^T(k) \\ \lambda^{1/2}\boldsymbol{x}^T(k-1) \\ \vdots \\ \lambda^{k/2}\boldsymbol{x}^T(0) \end{bmatrix}, \tag{4}$$

where N is the filter order (number of coefficients minus one), $\boldsymbol{x}(k)$ is the input signal vector $[x(k) \;\; x(k-1) \;\; \cdots \;\; x(k-N)]^T$, and the samples before instant $k = 0$ are assumed nulls.

The premultiplication of (1) by the orthonormal matrix $\boldsymbol{Q}(k)$, as can be seen in the next equation, triangularizes $\boldsymbol{X}(k)$ without affecting the cost function:

$$\boldsymbol{Q}(k)\boldsymbol{e}(k) = \boldsymbol{e}_q(k) = \begin{bmatrix} \boldsymbol{e}_{q_1}(k) \\ \boldsymbol{e}_{q_2}(k) \end{bmatrix} = \begin{bmatrix} \boldsymbol{d}_{q_1}(k) \\ \boldsymbol{d}_{q_2}(k) \end{bmatrix} - \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{U}(k) \end{bmatrix} \boldsymbol{w}(k). \tag{5}$$

The weighted-square error is minimized by choosing $\boldsymbol{w}(k)$ such that $\boldsymbol{d}_{q2}(k) - \boldsymbol{U}(k)\boldsymbol{w}(k)$ is zero. Equation (5) can be written in a recursive form while avoiding ever-increasing order for the vectors and matrices involved [5],

$$\begin{bmatrix} e_{q_1}(k) \\ \boldsymbol{d}_{q_2}(k) \end{bmatrix} = \boldsymbol{Q}_\theta(k) \begin{bmatrix} d(k) \\ \lambda^{1/2}\boldsymbol{d}_{q_2}(k-1) \end{bmatrix}, \tag{6}$$

where $\boldsymbol{Q}_\theta(k)$ is a sequence of Givens rotations that annihilates the elements of the input vector $\boldsymbol{x}(k)$ in

$$\begin{bmatrix} \boldsymbol{0}^T \\ \boldsymbol{U}(k) \end{bmatrix} = \boldsymbol{Q}_\theta(k) \begin{bmatrix} \boldsymbol{x}^T(k) \\ \lambda^{1/2}\boldsymbol{U}(k-1) \end{bmatrix}. \tag{7}$$

The following relation is obtained in the conventional QR algorithm by post-multiplying $\boldsymbol{e}_q^T(k)\boldsymbol{Q}(k)$ by the pinning vector $[1 \; 0 \; \cdots \; 0]^T$:

$$e(k) = e_{q_1}(k) \prod_{i=0}^{N} \cos\theta_i(k) = e_{q_1}(k)\gamma(k), \tag{8}$$

where $\gamma(k)$ is the first element of the first row of $\boldsymbol{Q}_\theta(k)$.

In the development of the fast QR algorithms, the way matrix $\boldsymbol{U}(k)$ is triangularized, upper or lower triangular, determines the type of the prediction error updated. In both types, matrix $\boldsymbol{Q}_\theta(k)$ can be partitioned as

$$\boldsymbol{Q}_\theta(k) = \begin{bmatrix} \gamma(k) & -\gamma(k)\boldsymbol{a}^T(k) \\ \boldsymbol{f}(k) & \boldsymbol{E}(k) \end{bmatrix}, \tag{9}$$

where, using (9) in (7) and recalling that $\boldsymbol{Q}_\theta(k)$ is orthonormal, it is possible to prove that $\boldsymbol{f}(k) = [\boldsymbol{U}(k)]^{-T}\boldsymbol{x}(k)$ is the normalized a posteriori forward (upper triangularization) or backward (lower triangularization) prediction error vector [6]. Furthermore, $\boldsymbol{a}(k) = \boldsymbol{U}^{-T}(k-1)\boldsymbol{x}(k)/\sqrt{\lambda}$ is the normalized a priori forward (upper triangularization) or backward (lower triangularization) prediction error vector [6], and $\boldsymbol{E}(k) = \lambda^{1/2}[\boldsymbol{U}(k)]^{-T}[\boldsymbol{U}(k-1)]^T$.

In fast QR algorithms, the QR decomposition is applied to the forward and backward prediction problems whose prediction errors are respectively defined as follows:

$$\boldsymbol{e}_f(k) = \left[\begin{array}{cc} \boldsymbol{d}_f(k) & \begin{array}{c} \boldsymbol{X}(k-1) \\ \boldsymbol{0}^T \end{array} \end{array} \right] \left[\begin{array}{c} 1 \\ -\boldsymbol{w}_f(k) \end{array} \right] \tag{10}$$

$$\boldsymbol{e}_b(k) = [\boldsymbol{X}(k) \;\; \boldsymbol{d}_b(k)] \left[\begin{array}{c} -\boldsymbol{w}_b(k) \\ 1 \end{array} \right], \tag{11}$$

where the reference or *desired* forward vector is defined by $\boldsymbol{d}_f(k) = [x(k) \;\; \lambda^{1/2}x(k-1) \;\; \cdots \;\; \lambda^{k/2}x(0)]^T$ and the reference or *desired* backward vector corresponds to $\mathbf{d}_b(k) = [x(k-N-1) \;\; \lambda^{1/2}x(k-N-2) \;\; \cdots \;\; \lambda^{(k-N-1)/2)}x(0) \;\; \boldsymbol{0}^T_{N+1}]^T$.

Note that the partitioned matrices in the last two equations correspond to $\boldsymbol{X}^{(N+2)}(k)$; the weighted input data matrix of order $N+1$—$\boldsymbol{X}(k)$ without superscript corresponds to an order N matrix or $\boldsymbol{X}^{(N+1)}(k)$. Our aim is to triangularize $\boldsymbol{X}^{(N+2)}(k)$ to obtain $\boldsymbol{Q}^{(N+2)}(k)$ such that

$$\boldsymbol{Q}^{(N+2)}(k)\boldsymbol{X}^{(N+2)}(k) = \left[\begin{array}{c} \boldsymbol{O} \\ \boldsymbol{U}^{(N+2)}(k) \end{array} \right]. \tag{12}$$

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

$$\boldsymbol{U}^{(N+2)}(k) = \left[\begin{array}{cc} \boldsymbol{0}^T & \| \boldsymbol{e}_b(k) \| \\ \boldsymbol{U}(k) & \boldsymbol{d}_{bq_2}(k) \end{array} \right]. \tag{13}$$

In the forward prediction problem, the lower triangular of $\boldsymbol{U}^{(N+2)}(k)$ is implemented by premultiplying $\boldsymbol{e}_f(k)$ by the product $\boldsymbol{Q}'_f(k)\boldsymbol{Q}_f(k) \left[\begin{array}{cc} \boldsymbol{Q}(k-1) & \boldsymbol{0} \\ \boldsymbol{0}^T & 1 \end{array} \right]$, where $\boldsymbol{Q}_f(k)$ and $\boldsymbol{Q}'_f(k)$ are two sets of Givens rotations generating $\| \boldsymbol{e}_f(k) \|$ and $\| \boldsymbol{e}_f^{(0)}(k) \|$, respectively. The resulting expression is

$$\boldsymbol{U}^{(N+2)}(k) = \boldsymbol{Q}'_{\theta f}(k) \left[\begin{array}{cc} \boldsymbol{d}_{fq_2}(k) & \boldsymbol{U}(k-1) \\ \| \boldsymbol{e}_f(k) \| & \boldsymbol{0}^T \end{array} \right] = \left[\begin{array}{cc} \boldsymbol{0} & \boldsymbol{R}(k) \\ \| \boldsymbol{e}_f^{(0)}(k) \| & \boldsymbol{z}^T(k) \end{array} \right], \tag{14}$$

where $[\boldsymbol{R}^T(k) \;\; \boldsymbol{z}(k)]^T$ is the right part of $\boldsymbol{U}^{(N+2)}(k)$.

The inverse of the last two expressions yields the following results:

$$[\boldsymbol{U}^{(N+2)}(k)]^{-1} = \begin{bmatrix} \frac{-\boldsymbol{U}^{-1}(k)\boldsymbol{d}_{bq_2}(k)}{\|\boldsymbol{e}_b(k)\|} & \boldsymbol{U}^{-1}(k) \\ \frac{1}{\|\boldsymbol{e}_b(k)\|} & \mathbf{0}^T \end{bmatrix}$$

$$= \begin{bmatrix} \frac{-\boldsymbol{z}^T(k)\boldsymbol{R}^{-1}(k)}{\|\boldsymbol{e}_f^{(0)}(k)\|} & \frac{1}{\|\boldsymbol{e}_f^{(0)}(k)\|} \\ \boldsymbol{R}^{-1}(k) & \mathbf{0} \end{bmatrix}. \qquad (15)$$

With the results obtained in (15), we can express the vectors $\boldsymbol{f}^{(N+2)}(k+1)$ and $\boldsymbol{a}^{(N+2)}(k+1)$ in terms of the partitions of $\left[\boldsymbol{U}^{(N+2)}(k+1)\right]^{-1}$ and $\left[\boldsymbol{U}^{(N+2)}(k)\right]^{-1}$, respectively. If we update $\boldsymbol{f}(k)$, the resulting algorithm will be the FQR_POS_B, whereas, updating $\boldsymbol{a}(k)$ leads to the FQR_PRI_B algorithm.

2.1. *The FQR_PRI_B algorithm*

This algorithm is obtained by expressing vector $\boldsymbol{a}^{(N+2)}(k+1) = [\boldsymbol{U}^{(N+2)}(k)]^{-T}$ $\boldsymbol{x}^{(N+2)}(k+1)/\sqrt{\lambda}$ in terms of the matrices in (15) and premultiplying the one that comes from the forward prediction problem by $\boldsymbol{Q}'_{\theta f}(k)\boldsymbol{Q}'^{T}_{\theta f}(k)$. The updating equation is

$$\begin{bmatrix} \frac{e'_b(k+1)}{\sqrt{\lambda}\|\boldsymbol{e}_b(k)\|} \\ \boldsymbol{a}(k+1) \end{bmatrix} = \boldsymbol{Q}'_{\theta f}(k) \begin{bmatrix} \boldsymbol{a}(k) \\ \frac{e'_f(k+1)}{\sqrt{\lambda}\|\boldsymbol{e}_f(k)\|} \end{bmatrix}. \qquad (16)$$

It is important to mention that, during the derivation of the previous equation, it was observed that the last element of $\boldsymbol{a}(k+1)$ in (16) is known in advance and is equal to $\frac{x(k+1)}{\sqrt{\lambda}\|\boldsymbol{e}_f^{(0)}(k)\|}$. The matrix equations for the FQR_PRI_B algorithm are shown in Table 2.

2.2. *The FQR_POS_B algorithm*

Expressing $\boldsymbol{f}^{(N+2)}(k+1) = [\boldsymbol{U}^{(N+2)}(k+1)]^{-T}\boldsymbol{x}^{(N+2)}(k+1)$ in terms of the matrices in (15) and premultiplying the one that comes from the forward prediction problem by $\boldsymbol{Q}'_{\theta f}(k+1)\boldsymbol{Q}'^{T}_{\theta f}(k+1)$ yields

$$\begin{bmatrix} \frac{e_b(k+1)}{\|\boldsymbol{e}_b(k+1)\|} \\ \boldsymbol{f}(k+1) \end{bmatrix} = \boldsymbol{Q}'_{\theta_f}(k+1) \begin{bmatrix} \boldsymbol{f}(k) \\ \frac{e_f(k+1)}{\|\boldsymbol{e}_f(k+1)\|} \end{bmatrix}. \qquad (17)$$

During the derivation of (17), it was observed that the last element of $\boldsymbol{f}(k+1)$ is $\frac{x(k+1)}{\|\boldsymbol{e}_f^{(0)}(k+1)\|}$. The term $\frac{e_f(k+1)}{\|\boldsymbol{e}_f(k+1)\|}$ can be calculated as $\gamma(k)\sin\theta_f(k+1)$, where $\sin\theta_f(k+1) = \frac{e_{fq_1}(k+1)}{\|\boldsymbol{e}_f(k+1)\|}$ is the sine of the angle of rotation matrix $\boldsymbol{Q}_f(k+1)$. The matrix equations for the FQR_POS_B algorithm are shown in Table 3.

Table 2. The FQR_PRI_B equations

```
for each $k$
{   1. Obtaining $\boldsymbol{d}_{fq_2}(k+1)$:
    $\begin{bmatrix} e_{fq_1}(k+1) \\ \boldsymbol{d}_{fq_2}(k+1) \end{bmatrix} = \mathbf{Q}_\theta(k) \begin{bmatrix} x(k+1) \\ \lambda^{1/2}\mathbf{d}_{fq_2}(k) \end{bmatrix}$
    2. Obtaining $\mathbf{a}(k+1)$:
    $\begin{bmatrix} \frac{e'_b(k+1)}{\sqrt{\lambda}\|\mathbf{e}_b(k)\|} \\ \mathbf{a}(k+1) \end{bmatrix} = \mathbf{Q}'_{\theta f}(k) \begin{bmatrix} \mathbf{a}(k) \\ \frac{e'_f(k+1)}{\sqrt{\lambda}\|\mathbf{e}_f(k)\|} \end{bmatrix}$
    3. Obtaining $\| \mathbf{e}_f(k+1) \|$:
    $\| \mathbf{e}_f(k+1) \| = \sqrt{e^2_{fq_1}(k+1) + \lambda \| \mathbf{e}_f(k) \|^2}$
    4. Obtaining $\mathbf{Q}'_{\theta f}(k+1)$:
    $\begin{bmatrix} \mathbf{0} \\ \| \mathbf{e}^{(0)}_f(k+1) \| \end{bmatrix} = \mathbf{Q}'_{\theta f}(k+1) \begin{bmatrix} \boldsymbol{d}_{fq_2}(k+1) \\ \| \mathbf{e}_f(k+1) \| \end{bmatrix}$
    5. Obtaining $\mathbf{Q}_\theta(k+1)$:
    $\begin{bmatrix} 1/\gamma(k+1) \\ \mathbf{0} \end{bmatrix} = \mathbf{Q}_\theta(k+1) \begin{bmatrix} 1 \\ -\mathbf{a}(k+1) \end{bmatrix}$
    6. Joint Process Estimation:
    $\begin{bmatrix} e_{q_1}(k+1) \\ \boldsymbol{d}_{q_2}(k+1) \end{bmatrix} = \mathbf{Q}_\theta(k+1) \begin{bmatrix} d(k+1) \\ \lambda^{1/2}\boldsymbol{d}_{q_2}(k) \end{bmatrix}$
    7. Updating the output error:
    $e(k+1) = e_{q_1}(k+1)\gamma(k+1)$
}
```

Table 3. The FQR_POS_B equations

```
for each $k$
{   1. Obtaining $\mathbf{d}_{fq_2}(k+1)$:
    $\begin{bmatrix} e_{fq_1}(k+1) \\ \boldsymbol{d}_{fq_2}(k+1) \end{bmatrix} = \boldsymbol{Q}_\theta(k) \begin{bmatrix} x(k+1) \\ \lambda^{1/2}\boldsymbol{d}_{fq_2}(k) \end{bmatrix}$
    2. Obtaining $\| \boldsymbol{e}_f(k+1) \|$:
    $\| \boldsymbol{e}_f(k+1) \| = \sqrt{e^2_{fq_1}(k+1) + \lambda \| \boldsymbol{e}_f(k) \|^2}$
    3. Obtaining $\boldsymbol{Q}'_{\theta f}(k+1)$:
    $\begin{bmatrix} \mathbf{0} \\ \| \boldsymbol{e}^{(0)}_f(k+1) \| \end{bmatrix} = \boldsymbol{Q}'_{\theta f}(k+1) \begin{bmatrix} \boldsymbol{d}_{fq_2}(k+1) \\ \| \boldsymbol{e}_f(k+1) \| \end{bmatrix}$
    4. Obtaining $\boldsymbol{f}(k+1)$:
    $\begin{bmatrix} \frac{e_b(k+1)}{\|\boldsymbol{e}_b(k+1)\|} \\ \boldsymbol{f}(k+1) \end{bmatrix} = \boldsymbol{Q}'_{\theta f}(k+1) \begin{bmatrix} \boldsymbol{f}(k) \\ \frac{e_f(k+1)}{\|\boldsymbol{e}_f(k+1)\|} \end{bmatrix}$
    5. Obtaining $\boldsymbol{Q}_\theta(k+1)$:
    $\begin{bmatrix} 1 \\ \mathbf{0} \end{bmatrix} = \boldsymbol{Q}^T_\theta(k+1) \begin{bmatrix} \gamma(k+1) \\ \boldsymbol{f}(k+1) \end{bmatrix}$
    6. Joint Process Estimation:
    $\begin{bmatrix} e_{q_1}(k+1) \\ \boldsymbol{d}_{q_2}(k+1) \end{bmatrix} = \boldsymbol{Q}_\theta(k+1) \begin{bmatrix} d(k+1) \\ \lambda^{1/2}\boldsymbol{d}_{q_2}(k) \end{bmatrix}$
    7. Updating the output error:
    $e(k+1) = e_{q_1}(k+1)\gamma(k+1)$
}
```

3. The different versions

The different versions presented in this section are based on the implementation of a particular matrix equation. For both the FQR_POS_B and FQR_PRI_B algorithms, equations (17) and (16) can be generically written as

$$\begin{bmatrix} \alpha \\ \mathbf{u} \end{bmatrix} = \mathbf{Q}'_{\theta f} \begin{bmatrix} \mathbf{v} \\ \beta \end{bmatrix}, \tag{18}$$

where α and β are scalars, matrix $\mathbf{Q}'_{\theta f}$ consists of a set of Givens rotations, and $\mathbf{u}$ may be computed as follows:

$$\begin{bmatrix} \alpha \\ u_1 \\ \vdots \\ u_{N+1} \end{bmatrix} = \begin{bmatrix} \mathbf{I}_N & \mathbf{0} & \mathbf{0} \\ \mathbf{0}^T & \cos\theta_0 & -\sin\theta_0 \\ \mathbf{0}^T & \sin\theta_0 & \cos\theta_0 \end{bmatrix} \cdots \begin{bmatrix} \cos\theta_N & \mathbf{0}^T & -\sin\theta_N \\ \mathbf{0} & \mathbf{I}_N & \mathbf{0} \\ \sin\theta_N & \mathbf{0}^T & \cos\theta_N \end{bmatrix} \begin{bmatrix} v_1 \\ \vdots \\ v_{N+1} \\ \beta \end{bmatrix} \tag{19}$$

Note that vector $\mathbf{v}$ is updated to $\mathbf{u}$ without using α and with no prior knowledge of any element of $\mathbf{u}$. On the other hand, using the fact that the inverse of matrix $\mathbf{Q}'_{\theta f}$ corresponds to $\mathbf{Q}'^{T}_{\theta f}$, it is also possible to derive the following implementation:

$$\begin{bmatrix} v_1 \\ \vdots \\ v_{N+1} \\ \beta \end{bmatrix} = \begin{bmatrix} \cos\theta_N & \mathbf{0}^T & \sin\theta_N \\ \mathbf{0} & \mathbf{I}_N & \mathbf{0} \\ -\sin\theta_N & \mathbf{0}^T & \cos\theta_N \end{bmatrix} \cdots \begin{bmatrix} \mathbf{I}_N & \mathbf{0} & \mathbf{0} \\ \mathbf{0}^T & \cos\theta_0 & \sin\theta_0 \\ \mathbf{0}^T & -\sin\theta_0 & \cos\theta_0 \end{bmatrix} \begin{bmatrix} \alpha \\ u_1 \\ \vdots \\ u_{N+1} \end{bmatrix}, \tag{20}$$

and the updating process from $\mathbf{v}$ to $\mathbf{u}$ can be carried out if we know u_{N+1} in advance.

3.1. The FQR_PRI_B: Versions 1 and 2

As mentioned before in the equation corresponding to the second step of this algorithm, the last element of $\mathbf{a}(k+1)$ is known before the updating process of this vector. This fact leads to the two different versions of this algorithm.

Table 4. The FQR_PRI_B equations Version 1

```
⋮
2. Obtaining $\mathbf{a}(k+1)$:
$aux_0 = \frac{x(k+1)}{\lambda^{1/2}\|\mathbf{e}_f^{(0)}(k)\|}$;
$a_{N+1}(k+1) = aux_0$;
for $i = 1 : N$
{  $a_{N+1-i}(k+1) = \frac{a_{N+2-i}(k) - \sin\theta'_{f_{i-1}}(k)aux_{i-1}}{\cos\theta'_{f_{i-1}}(k)}$;
   $aux_i = -\sin\theta'_{f_{i-1}}(k)a_{N+1-i}(k+1)$
                                   $+\cos\theta'_{f_{i-1}}(k)aux_{i-1}$;
}
⋮
```

Table 5. FQR_PRI_B: Version 2

```
⋮
2. Obtaining $\mathbf{a}(k+1)$:
$aux_0 = \frac{e_{fq_1}(k+1)}{\gamma(k)\lambda^{1/2}\|\mathbf{e}_f(k)\|}$;
for $i = 1 : N+1$
{  $a_{i-1}(k+1) = \cos\theta'_{f_{N-1-i}}(k)a_i(k)$
                                   $-\sin\theta'_{f_{N+1-i}}(k)aux_{i-1}$;
   $aux_i = \sin\theta'_{f_{N+1-i}}(k)a_i(k) + \cos\theta'_{f_{N+1-i}}(k)aux_{i-1}$;
}
\* $a_0(k+1)$ corresponds to $\frac{e'_b(k+1)}{\lambda^{1/2}\|\mathbf{e}_b(k)\|}$ *\
$a_{N+1}(k+1) = aux_{N+1}$;
⋮
```

Its first version is based upon the fact that the last element of $\mathbf{a}(k+1)$ is known prior to its calculation. This version was introduced in [9]. Table 4 presents the different steps required for its implementation.

The second version of the FQR_PRI_B algorithm is based on the computation of $\mathbf{a}(k+1)$ according to the matrix equation and requires the calculation of $\frac{e'_f(k+1)}{\sqrt{\lambda}\|\mathbf{e}_f(k)\|}$. This version was first presented in [6]. The corresponding implementation of this matrix equation is presented in Table 5.

Table 6. FQR_POS_B: Version 1

```
⋮
4. Obtaining $f(k+1)$:
$aux_0 = x(k+1)/ \parallel e_f^{(0)}(k+1) \parallel$;
$f_{N+1}(k+1) = aux_0$;
for $i = 1 : N$
{ $f_{N+1-i}(k+1) = \frac{f_{N+2-i}(k) - \sin\theta'_{f_{i-1}}(k+1) aux_{i-1}}{\cos\theta'_{f_{i-1}}(k+1)}$;
  $aux_i = -\sin\theta'_{f_{i-1}}(k+1) f_{N+1-i}(k)$
                $+ \cos\theta'_{f_{i-1}}(k+1) aux_{i-1}$;
}
⋮
```

3.2. The FQR_POS_B: Versions 1 and 2

For the FQR with a posteriori backward prediction (FQR_POS_B), it was seen that in its fourth step, the last element of $\mathbf{f}(k+1)$ is known before its updating process. This fact also leads to two different versions of the algorithm.

The first version of the FQR_POS_B algorithm is based upon the fact that the last element of $f(k+1)$ is known in advance or prior to its calculation ($f_{N+1}(k+1) = \frac{x(k+1)}{\|e_f^{(0)}(k+1)\|}$). This version has been previously reported only in [2] and [3]. Table 6 presents its implementation (step 4 corresponds to the only distinct part between the two versions). A detailed and complete algorithmic description of this new version in presented in the Appendix.

The second version of the FQR_POS_B algorithm is based upon the computation of $f(k+1)$ according to the matrix equation and requires the calculation of $\frac{e_f(k+1)}{\|e_f(k+1)\|}$. This version was introduced in [8], and Table 7 presents its implementation. Note that, for this version, the term $\frac{e_f(k+1)}{\|e_f(k+1)\|}$ can be calculated as $\gamma(k)\sin\theta_f(k+1)$, where $\sin\theta_f(k+1) = \frac{e_{fq_1}(k+1)}{\|e_f(k+1)\|}$ is the sine of the angle of rotation matrix $Q_f(k+1)$.

4. Computational complexity

The computational complexity for each of the discussed algorithms is shown in this section. Table 8 compares the four versions of the fast QR algorithms in terms of number of operations (additions, multiplications, divisions, and squared roots).

It is important to note that fast QR algorithms with backward prediction error recursion are of minimal complexity and are known to be backward stable under

Table 7. FQR_POS_B: Version 2

```
⋮
4. Obtaining $\boldsymbol{f}(k+1)$:
$aux_0 = \frac{\gamma(k) e_{fq_1}(k+1)}{\|\boldsymbol{e}_f(k+1)\|}$;
for $i = 1 : N+1$
{  $f_{i-1}(k+1) = \cos\theta'_{f_{N+1-i}}(k+1) f_i(k)$
                  $- \sin\theta'_{f_{N+1-i}}(k+1) aux_{i-1}$;
   $aux_i = \sin\theta'_{f_{N+1-i}}(k+1) f_i(k)$
                  $+ \cos\theta'_{f_{N+1-i}}(k+1) aux_{i-1}$;
}
\* $f_0(k+1)$ corresponds to $\frac{e_b(k+1)}{\|\boldsymbol{e}_b(k+1)\|}$ *\
$f_{N+1}(k+1) = aux_{N+1}$;
⋮
```

Table 8. Comparison of computational complexity

Algorithm	Add	Mult.	Div.	Sqr. root
FQR_PRI_B (V. 1) [9]	8p − 1	19p + 2	5p + 1	2p + 1
FQR_PRI_B (V. 2) [6]	8p + 1	20p + 6	4p + 2	2p + 1
FQR_POS_B (V. 1)	8p + 1	19p + 4	4p + 1	2p + 1
FQR_POS_B (V. 2) [8]	8p + 1	20p + 5	3p + 1	2p + 1

persistent excitation [8], [7]. Moreover, they present lower computational complexity (note from Table 8 that all algorithms analyzed in this text, including the proposed one, have similar complexity) if compared to fast QR algorithms with forward prediction error recursion [4], [1].

5. Performance in finite precision

This section presents some simulation results to compare the discussed algorithms in finite precision. The setup is a system identification problem of order $N = 10$. The input signal to the unknown system was colored noise with eigenvalue spread equal to 187 and SNR=40 dB. The mean-squared error (MSE) was measured by using floating-point arithmetic with quantization applied to the mantissa in all operations. The mantissa was rounded excluding the sign bit and assuming that the exponent wordlength was sufficient to represent all dynamic ranges. In the first experiment, the mantissa wordlength was varied (8 to 16 bits excluding the sign bit) while keeping fixed the value of the forgetting factor ($\lambda = 0.98$). Next, the λ was varied (0.90 to 1.0) for a fixed mantissa wordlength (10 bits). For the

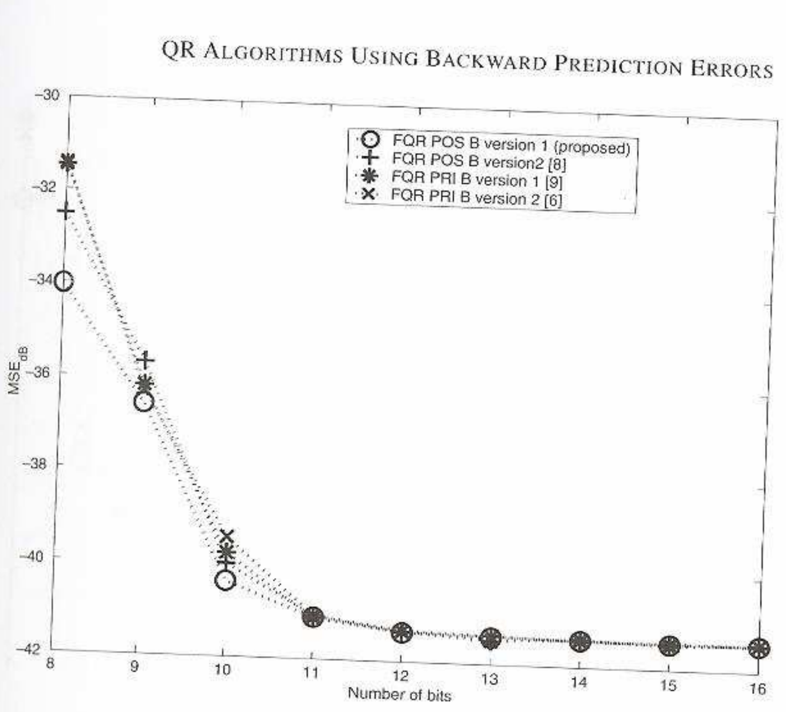

Figure 1. Performance of the algorithms in finite precision environment when varying B, the number of bits in the mantissa ($\lambda = 0.98$, no passive orthogonal rotation constraints)

computation of the MSE (in dB) in both experiments, the last 4000 iterations of simulations with 5000 samples were averaged after an average of 10 independent runs. The results can be observed in Figure 1 and Figure 2. The figures indicate that for typical values of forgetting factors (λ) ranging from 0.9 to 1, the FQR_POS_B versions perform slightly better than the FQR_PRI_B versions. In Figures 1 and 2 the orthogonal rotations were performed without passive rotation constraints in finite precision.

Additional simulations with passive orthogonal rotation constraints were run. The results obtained from these simulations are shown in Figures 3 and 4. It can be seen from these figures that, among the compared algorithms, those based on a posteriori errors outperform the ones based on a priori errors. It can also be seen that the FQR_POS_B version 1 algorithm, which was proposed in this paper, outperforms all the other discussed algorithms in finite precision when passive orthogonal rotation constraint is used.

6. Conclusions

In this paper, we have considered the two fast QR algorithms based on backward prediction errors (a posteriori and a priori). Two different versions for each

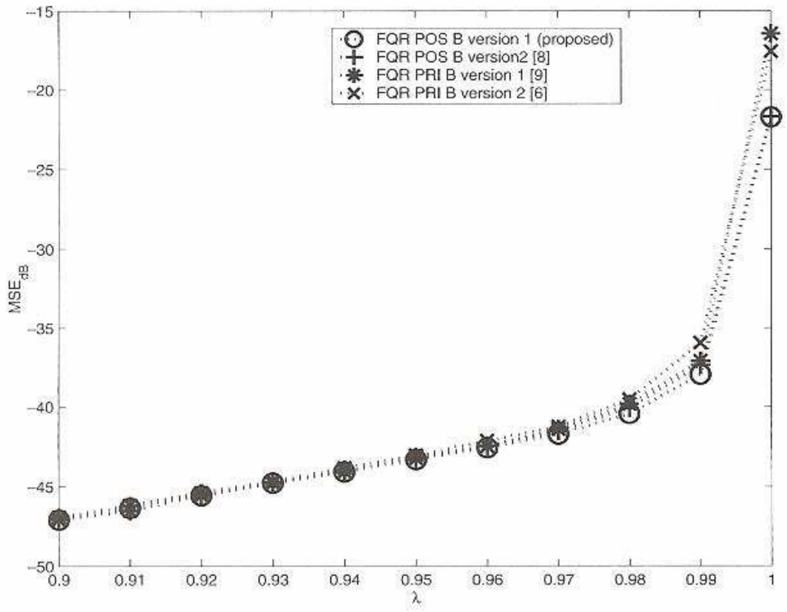

Figure 2. MSE in dB for different values of λ (B, the number of bits was set to 10, no passive orthogonal rotation constraints)

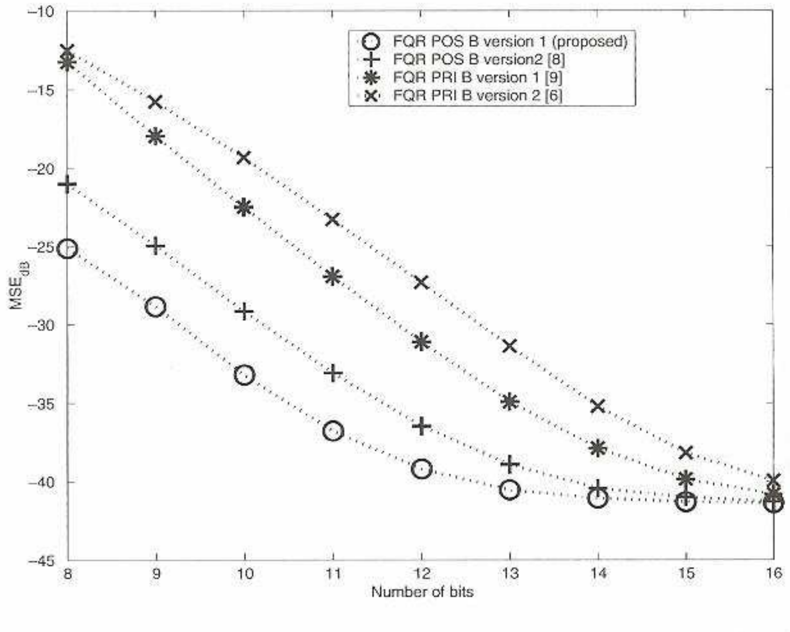

Figure 3. Performance of the algorithms in finite precision when varying B, the number of bits in the mantissa ($\lambda = 0.98$, with passive orthogonal rotation constraints)

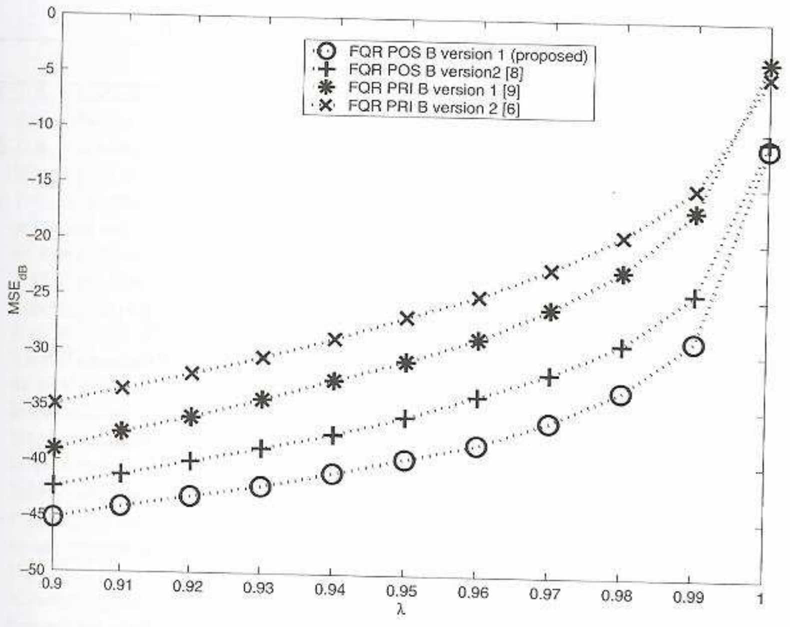

Figure 4. MSE in dB for different values of λ (B, the number of bits was set to 10, with passive orthogonal rotation constraints)

algorithm were derived and, for the a priori case, it became clear that these two versions were actually the algorithms previously introduced in [6] and [9]. For the a posteriori case, a new version was derived as the alternative implementation of the same matrix equations used in [8].

Comparisons were carried out in terms of computational complexity and performance in finite precision. From simulation results, it was observed that the performances of the discussed algorithms were similar in finite precision for different values of λ and number of bits in the mantissa when orthogonal rotations were performed without passive constraints. It was also observed that the algorithms based on a posteriori error recursion outperformed algorithms based on a priori error recursion when passive rotations are used, possibly because of the reduced number of division operations used by the algorithms based on a posteriori error recursion.

Appendix

This appendix provides a detailed algorithmic description of the FQR_POS_B version 1 in Table 9.

Table 9. The new algorithm

FQR_POS_B - Version 1

```
Initialization: $\| \boldsymbol{e}_f(k) \| = \epsilon = $ small positive value;
                $\boldsymbol{d}_{fq2}(k) = \boldsymbol{d}_{q2}(k) = \text{zeros}(N+1, 1);$
                $\cos\boldsymbol{\theta}(k) = \text{ones}(N+1, 1);$
                $\sin\boldsymbol{\theta}(k) = \boldsymbol{f}(k) = \text{zeros}(N+1, 1);$
for $k = 1, 2, \ldots$
{ $e^{(0)}_{fq_1}(k+1) = x(k+1);$
  for $i = 1 : N+1$
  { $e^{(i)}_{fq_1}(k+1) = \cos\theta_{i-1}(k) e^{(i-1)}_{fq_1}(k+1) - \sin\theta_{i-1}(k)\lambda^{1/2} d_{fq2_{N+2-i}}(k);$
    $d_{fq2_{N+2-i}}(k+1) = \sin\theta_{i-1}(k) e^{(i-1)}_{fq_1}(k+1) + \cos\theta_{i-1}(k)\lambda^{1/2} d_{fq2_{N+2-i}}(k);$
  }
  $e_{fq_1}(k+1) = e^{(N+1)}_{fq_1}(k+1);$
  $\| \boldsymbol{e}_f(k+1) \| = \sqrt{e^2_{fq_1}(k+1) + \lambda \| \boldsymbol{e}_f(k) \|^2};$
  $\| \boldsymbol{e}^{(N+1)}_f(k+1) \| = \| \boldsymbol{e}_f(k+1) \|;$
  for $i = 1 : N+1$
  { $\| \boldsymbol{e}^{(N+1-i)}_f(k+1) \| = \sqrt{\| \boldsymbol{e}^{(N+2-i)}_f(k+1) \|^2 + d^2_{fq2_i}(k+1)};$
    $\cos\theta'_{f_{N+1-i}}(k+1) = \| \boldsymbol{e}^{(N+2-i)}_f(k+1) \| / \| \boldsymbol{e}^{(N+1-i)}_f(k+1) \|;$
    $\sin\theta'_{f_{N+1-i}}(k+1) = d_{fq2_i}(k+1) / \| \boldsymbol{e}^{(N+1-i)}_f(k+1) \|;$
  }
  $aux_0 = x(k+1) / \| \boldsymbol{e}^{(0)}_f(k+1) \|;$
  $f_{N+1}(k+1) = aux_0;$
  for $i = 1 : N$
  { $f_{N+1-i}(k+1) = \frac{f_{N+2-i}(k) - \sin\theta'_{f_{i-1}}(k+1) aux_{i-1}}{\cos\theta'_{f_{i-1}}(k+1)};$
    $aux_i = -\sin\theta'_{f_{i-1}}(k+1) f_{N+1-i}(k+1) + \cos\theta'_{f_{i-1}}(k+1) aux_{i-1};$
  }
  $\gamma^{(0)}(k+1) = 1;$
  for $i = 1 : N+1$
  { $\sin\theta_{i-1}(k+1) = f_{N+2-i}(k+1) / \gamma^{(i-1)}(k+1);$
    $\cos\theta_{i-1}(k+1) = \sqrt{1 - \sin^2\theta_{i-1}(k+1)};$
    $\gamma^{(i)}(k+1) = \cos\theta_{i-1}(k+1)\gamma^{(i-1)}(k+1);$
  }
  $\gamma(k+1) = \gamma^{(N+1)}(k+1);$
  $e^{(0)}_{q_1}(k+1) = d(k+1);$
  for $i = 1 : N+1$
  { $e^{(i)}_{q_1}(k+1) = \cos\theta_{i-1}(k+1) e^{(i-1)}_{q_1}(k+1) - \sin\theta_{i-1}(k+1)\lambda^{1/2} d_{q2_{N+2-i}}(k);$
    $d_{q2_{N+2-i}}(k+1) = \sin\theta_{i-1}(k+1) e^{(i-1)}_{q_1}(k+1) + \cos\theta_{i-1}(k+1)\lambda^{1/2} d_{q2_{N+2-i}}(k);$
  }
  $e_{q_1}(k+1) = e^{(N+1)}_{q_1}(k+1);$
  $e(k+1) = e_{q_1}(k+1)\gamma(k+1);$
}
```

References

[1] J. A. Apolinário Jr., and P. S. R. Diniz, A new fast QR algorithm based on *a Priori* errors, *IEEE Signal Process. Lett.*, vol. 4, pp. 307–309, Nov. 1997.

[2] J. A. Apolinário Jr., New algorithms of adaptive filtering: LMS with data-reusing and fast RLS based on QR decomposition, D.Sc. Thesis, COPPE/UFRJ, Rio de Janeiro, RJ, Brazil, 1998.

[3] 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, in *Proceedings of the First Balkan Conference on Signal Processing, Communications, Circuits, and Systems*, Istanbul, Turkey, June 2000.

[4] J. M. Cioffi, The Fast Adaptive ROTOR's RLS Algorithm, *IEEE Trans. Acoust. Speech, Signal Process.*, vol. ASSP-38, pp. 631–653, Apr. 1990.

[5] P. S. R. Diniz, *Adaptive Filtering: Algorithms and Practical Implementation*, second edition, Kluwer Academic Press, Norwell, MA, 2002.

[6] M. D. Miranda and M. Gerken, A hybrid QR-lattice least squares algorithm using *a Priori* errors, in *Proceedings of the 38 Midwest Symposium on Circuits and Systems*, pp. 983–986, Rio de Janeiro, RJ, Brazil, Aug. 1995.

[7] M. D. Miranda and M. Gerken, A hybrid least squares QR-lattice algorithm using *a Priori* errors, *IEEE Trans. Signal Process.*, vol. 45, pp. 2900–2911, Dec. 1997.

[8] P. A. Regalia and M. G. Bellanger, On the duality between fast QR methods and lattice methods in least squares adaptive filtering, *IEEE Trans. Signal Process.*, vol. SP-39, pp. 879–891, Apr. 1991.

[9] A. A. Rontogiannis and S. Theodoridis, New fast inverse QR least squares adaptive algorithms, in *Proceedings of the IEEE International Conference on Acoustic, Speech, and Signal Processing*, pp. 1412–1415, Detroit, MI, 1995.