ON FAST QR ALGORITHMS BASED ON BACKWARD PREDICTION ERRORS: NEW RESULTS AND COMPARISONS

José A. Apolinário Jr.[1], *Marcio G. Siqueira*[2], *and Paulo S. R. Diniz*[3]

[1]Escuela Politécnica del Ejército
Av. El Progreso s/n
Sangolquí
Apartado 231-B, Ecuador
apolin@espe.edu.ec

[2]Philips Semiconductors
811 E. Arques Ave.
M/S 71, P.O. Box 3409
Sunnyvale, CA 94088-3409, USA
mgs@ieee.org

[3]COPPE/UFRJ
P.O. Box 68504
Rio de Janeiro, RJ
CEP 21.945-970, Brazil
diniz@lps.ufrj.br

ABSTRACT

Fast QR decomposition algorithms based on backward prediction errors are well known for their good numerical behavior and low complexity. This paper examines two different versions of the fast QR algorithm based on *a priori* backward prediction errors as well as other two corresponding versions of the fast QR algorithm based on *a posteriori* backward prediction errors. The main matrix equations are presented and different versions of the discussed algorithms are derived. From this study, a new formulation for the fast QR algorithm based on *a posteriori* backward prediction error recursion is derived. The discussed algorithms are compared; differences in computational complexity and in finite precision behavior are shown.

1. INTRODUCTION

Fast Recursive Least Squares algorithms based on QR decomposition (FQR-RLS) are adaptive filtering algorithms widely studied due to their numerical robustness and to their regular structure that can lead to efficient implementations.

A number of fast algorithms (O[N]) have been derived [2]–[6] based on the conventional (or O[N^2]) QR decomposition method [1]. It is known [6] 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 updates the forward prediction errors while a lower triangularization updates backward prediction errors. Table 1 presents the classification used in [6] and introduces how these algorithms will be designated hereafter.

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

Table 1: Classification of fast QR algorithms.

Error Type	Prediction	
	Forward	Backward
A Posteriori	FQR_POS_F [2]	FQR_POS_B [3]
A Priori	FQR_PRI_F [6]	FQR_PRI_B [4, 5]

2. THE FQR ALGORITHMS BASED ON BACKWARD PREDICTION ERRORS

This section presents the matrix equations of the two algorithms being studied. The notation used is the same used in [1], [6], and [7]. The equations for the FQR_PRI_B and FQR_POS_B algorithms are shown in Tables 2 and 3 respectively.

3. THE DIFFERENT VERSIONS

The different versions presented in this section are based on the implementation of a particular matrix equation. For both algorithms, the FQR_PRI_B and the FQR_POS_B, we have a matrix equation with the following structure

$$\begin{bmatrix} \alpha \\ \mathbf{u} \end{bmatrix} = \mathbf{Q}'_{\theta f} \begin{bmatrix} \mathbf{v} \\ \beta \end{bmatrix} \tag{1}$$

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

Table 2: The FQR_PRI_B equations.

for each k

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

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

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

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

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:

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

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

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

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:

$e(k+1) = e_{q_1}(k+1)\gamma(k+1)$

}

$$\begin{bmatrix} \alpha \\ u_1 \\ \vdots \\ u_{N+1} \end{bmatrix} = \begin{bmatrix} \mathbf{I}_N & \mathbf{0} & \mathbf{0} \\ \mathbf{0}^T & cos\theta_0 & -sin\theta_0 \\ \mathbf{0}^T & sin\theta_0 & cos\theta_0 \end{bmatrix} \cdots \begin{bmatrix} cos\theta_N & \mathbf{0}^T & -sin\theta_N \\ \mathbf{0} & \mathbf{I}_N & \mathbf{0} \\ sin\theta_N & \mathbf{0}^T & cos\theta_N \end{bmatrix} \begin{bmatrix} v_1 \\ \vdots \\ v_{N+1} \\ \beta \end{bmatrix} \quad (2)$$

The vector $\mathbf{v}$ is updated without using α and with no prior knowledge of any element of $\mathbf{u}$.

The inverse of matrix $\mathbf{Q}'_{\theta f}$ corresponds to $\mathbf{Q}'^T_{\theta f}$. Using this relation, it is possible to derive the following implementation

$$\begin{bmatrix} v_1 \\ \vdots \\ v_{N+1} \\ \beta \end{bmatrix} = \begin{bmatrix} cos\theta_N & \mathbf{0}^T & sin\theta_N \\ \mathbf{0} & \mathbf{I}_N & \mathbf{0} \\ -sin\theta_N & \mathbf{0}^T & cos\theta_N \end{bmatrix} \cdots \begin{bmatrix} \mathbf{I}_N & \mathbf{0} & \mathbf{0} \\ \mathbf{0}^T & cos\theta_0 & sin\theta_0 \\ \mathbf{0}^T & -sin\theta_0 & cos\theta_0 \end{bmatrix} \begin{bmatrix} \alpha \\ u_1 \\ \vdots \\ u_{N+1} \end{bmatrix} \quad (3)$$

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

It can be observed during the derivation of the equation corresponding to the second step of this algorithm (*Obtaining* $\mathbf{a}(k+1)$) that 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 the same algorithm.

The first version of this algorithm is based upon the fact that the last element of $\mathbf{a}(k+1)$ is known prior to its calculation (or $a_{N+1}(k+1) = \frac{x(k+1)}{\sqrt{\lambda}\|\mathbf{e}_f^{(0)}(k)\|}$). This version was presented in [5]. Table 4 presents the different steps required for its implementation.

Table 4: FQR_PRI_B: Version 1.

$\vdots$

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

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

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

for $i = 1 : N$

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

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

}

$\vdots$

The second version of the FQR_PRI_B algorithm is based on the computation of $\mathbf{a}(k+1)$ according to the matrix equa-

tion and requires the calculation of $\frac{e'_f(k+1)}{\sqrt{\lambda}\|\boldsymbol{e}_f(k)\|}$. This version was shown in [4]. The corresponding implementation of this matrix equation is presented in Table 5.

Table 5: FQR_PRI_B: Version 2.

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

3.2. The FQR_POS_B: Versions 1 and 2

For the FQR algorithm based on *a posteriori* backward prediction (FQR_POS_B), it can be seen that in the derivation of the equation corresponding the fourth step (*Obtaining* $\boldsymbol{f}(k+1)$), the last element of this vector is known before its updating process.

The first version of the FQR_POS_B algorithm is based upon the fact that the last element of $\boldsymbol{f}(k+1)$ is known in advance or prior to its calculation ($f_{N+1}(k+1) = \frac{x(k+1)}{\|\boldsymbol{e}_f^{(0)}(k+1)\|}$). This version has only been reported previously in [7]. Table 6 shows the different steps required for its implementation.

Table 6: FQR_POS_B: Version 1.

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

The second version of the FQR_POS_B algorithm is based upon the computation of $\boldsymbol{f}(k+1)$ according to the matrix equation and requires the calculation of $\frac{e_f(k+1)}{\|\boldsymbol{e}_f(k+1)\|}$. This version was presented in [3] and Table 7 presents its implementation. Note that, for this version, 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 $\boldsymbol{Q}_f(k+1)$.

Table 7: FQR_POS_B: Version 2.

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

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). In Table 8, $p = N + 1$ is the number of coefficients.

Table 8: Comparison of computational complexity.

ALGORITHM	ADD	MULT.	DIV.	SQRT
FQR_PRI_B (V. 1)	8p-1	19p+2	5p+1	2p+1
FQR_PRI_B (V. 2)	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)	8p+1	20p+5	3p+1	2p+1

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 persistent excitation [3, 8]. Moreover, they present lower computational complexity if compared to fast QR algorithms with forward prediction error recursion [6], [2].

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 $SNR = 40dB$. The mean-square 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 the exponent wordlength was sufficient to represent all dynamic ranges. In all algorithms, the constraint of passive rotations ($\sin^2\theta + \cos^2\theta \leq 1$) was imposed. 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 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) from 0.9 to 1, the FQR_POS_B versions perform slightly better than the FQR_PRI_B versions.

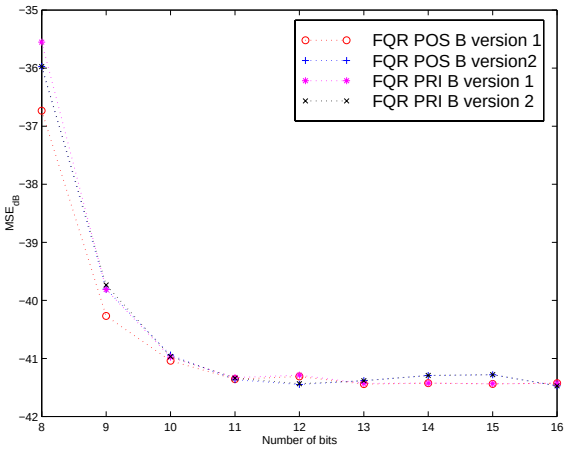

Figure 1: Performance of the algorithms in finite-precision (varying B, the number of bits in the mantissa).

6. CONCLUSIONS

In this paper, we have analyzed two Fast QR algorithms based on backward prediction with *a posteriori* and *a priori* error recursion. Two different versions for each algorithm were derived. An analysis followed, and it became clear that two of the discussed algorithms had been previously presented in [4] and [5]. Moreover, a new version, never reported in a publication before, was derived. The new version of this algorithm is a modified version from the algorithm previously shown in [3]. Computational complexity and finite precision characteristics were compared. From the simulation results it was observed that the performance of the discussed algorithms was similar in finite precision with different values of λ and number of bits in the mantissa, with the algorithms based on *a posteriori* error recursion slightly outperforming the algorithms based on *a priori* error recursion.

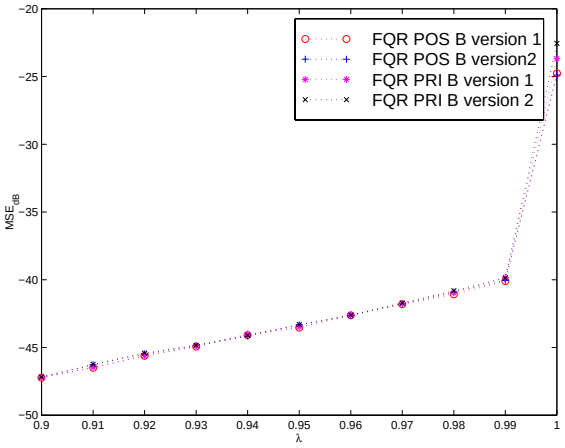

Figure 2: MSE in db for different values of λ.

7. REFERENCES

[1] Diniz, P. S. R. *Adaptive Filtering: Algorithms and Practical Implementation*. Norwell, MA, USA, Kluwer Academic Press, 1997.

[2] Cioffi, J. M., "The Fast Adaptive ROTOR's RLS Algorithm", *IEEE Transactions on Acoustic, Speech, and Signal Processing*, vol. ASSP-38, pp. 631–653, Apr. 1990.

[3] Regalia, P. A., Bellanger, M. G., "On the Duality between Fast QR Methods and Lattice Methods in Least Squares Adaptive Filtering", *IEEE Transactions on Signal Processing*, vol. SP-39, pp. 879–891, Apr. 1991.

[4] Miranda, M. D., Gerken, M., "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.

[5] Rontogiannis, A. A., Theodoridis, S., "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, USA, 1995.

[6] Apolinário Jr., J. A., Diniz, P. S. R., "A New Fast QR Algorithm Based on *a Priori* Errors", *IEEE Signal Processing Letters*, vol. 4, pp. 307–309, Nov. 1997.

[7] Apolinário Jr., J. A., "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.

[8] Miranda, M. D., Gerken, M., "A Hybrid Leat Squares QR-Lattice Algorithm Using *a Priori* Errors", *IEEE Transactions on Signal Processing*, vol. 45, pp. 2900–2911, Dec. 1997.