SOLUTION TO THE WEIGHT EXTRACTION PROBLEM IN FAST QR-DECOMPOSITION RLS ALGORITHMS

Mobien Shoaib,[1] *Stefan Werner*,[1] *José A. Apolinário Jr.,*[2] *and Timo I. Laakso*[1]

[1]**Helsinki University of Technology**
Signal Processing Laboratory
Espoo, Finland
{mobien, werner, timo}@wooster.hut.fi

[2]**Instituto Militar de Engenharia**
Depto. Engenharia Elétrica
Rio de Janeiro, Brazil
apolin@ieee.org

ABSTRACT

Fast QR decomposition RLS (FQRD-RLS) algorithms are well known for their good numerical properties and low computational complexity. However, the FQRD-RLS algorithms do not provide access to the filter weights, and so far their use has been limited to problems seeking an estimate of the output error signal. In this paper we present a novel technique to obtain the filter weights of the FQRD-RLS algorithm at any time instant. As a consequence, we extend the range of applications to include problems where explicit knowledge of the filter weights is required. The proposed weight extraction technique is tested in a system identification setup. The results verify our claim that the extracted coefficients of the FQRD-RLS algorithm are identical to those obtained by any RLS algorithm such as the inverse QRD-RLS algorithm.

1. INTRODUCTION

The recursive least-squares (RLS) is one of the fastest converging adaptive filtering algorithms. The convergence speed of the RLS algorithm usually serves as a benchmark for adaptive filtering algorithms. However, there are numerical stability issues associated with it, mainly when implemented in a finite precision environment [1]. The conventional QRD-RLS algorithm exhibits RLS like convergence and numerical robustness at the same computational complexity of the RLS, i.e., $\mathcal{O}(N^2)$, N being the number of filter coefficients [2]. A number of low-complexity derivatives of the QRD-RLS algorithm have been proposed [2–8]. In this paper we focus on one efficient subset, commonly called FQRD-RLS algorithms, with computational complexity of $\mathcal{O}(N)$.

The idea in FQRD-RLS algorithms is to exploit the underlying time-shift structure of the input data vector in order to replace matrix update equations with vector update equations [8]. The vector update equations are derived from forward and backward predictions. This paper considers algorithms based on the update of backward prediction errors which are numerically robust [7].

The main limitation of the FQRD-RLS algorithms is the unavailability of an explicit weight vector term. Furthermore, it does not directly provide the variables allowing for a straightforward computation of the weight vector, as is the case with the conventional QRD-RLS algorithm, where a back-substitution procedure can be used to compute the coefficients. Therefore, the applications are limited to output error based (e.g., noise or echo cancellation), or to those requiring a decision-feedback estimate of the training signal (e.g., continuously updated equalizer). The absence of weights in FQRD-RLS algorithms makes the problem of system identification non-trivial.

This paper addresses the problem of extracting the weight vector from the internal variables of the FQRD-RLS algorithm. The main results, summarized by two lemmas, provides us with an algorithm that allows us at any time instant during adaptation to sequentially extract the columns of the Cholesky factor embedded in the FQRD-RLS algorithm. From the Cholesky factor we can obtain the true weights of the underlying LS problem by reusing the known FQRD-RLS variables. We emphasize that the proposed method relies on the knowledge of only vector updates present in the FQRD RLS algorithms, as opposed to the matrix-embedded structure of the conventional QR-RLS described in [9]. The problem of parameter identification has been addressed in [10] using the duality between the FQR-RLS algorithm proposed in [6, 10] and the normalized Lattice structure. The relation between the results presented in this paper and those of [10], is under investigation.

In the following we present the basic principles of the FQRD-RLS algorithm. Thereafter, the new weight extraction (WE) algorithm is presented. Simulation results verifying our claims are followed by conclusions.

2. THE FQRD-RLS ALGORITHM

In this section we provide the basic concepts of QRD-RLS algorithms and one version of the FQRD-RLS algorithms, herein named the FQR_ PRI_B algorithm [7, 8], to aid the explanation of the novel weight extraction technique.

2.1. Basic concepts of QR decomposition algorithms

The RLS algorithm minimizes the following cost function

$$\xi(k) = \sum_{i=0}^{k} \lambda^{k-i}[d(i) - \mathbf{x}^{\mathrm{T}}(i)\mathbf{w}(k-1)]^2 = \|\mathbf{e}(k)\|^2 \qquad (1)$$

where λ is the forgetting factor and $\mathbf{e}(k) \in \mathbb{R}^{(k+1)\times 1}$ is the *a posteriori* error vector given as

$$\mathbf{e}(k) = \mathbf{d}(k) - \mathbf{X}(k)\mathbf{w}(k-1) \qquad (2)$$

The authors thank Academy of Finland, Smart and Novel Radios (SMARAD) Center of Excellence, CNPq, CAPES, and FAPERJ for partial funding of this paper.

where $\mathbf{d}(k) \in \mathbb{R}^{(k+1)\times 1}$ is the desired signal vector, $\mathbf{X}(k) \in \mathbb{R}^{(k+1)\times N}$ is the input data matrix, and $\mathbf{w}(k) \in \mathbb{R}^{N\times 1}$. The QRD-RLS algorithm uses an orthogonal rotation matrix $\mathbf{Q}(k)$ to triangularize matrix $\mathbf{X}(k)$ [2] as in

$$\begin{bmatrix} \mathbf{0}_{(k+1-N)\times N} \\ \mathbf{U}(k) \end{bmatrix} = \mathbf{Q}(k)\mathbf{X}(k) \tag{3}$$

where $\mathbf{U}(k) \in \mathbb{R}^{N\times N}$ is the Cholesky factor of the deterministic autocorrelation matrix $\mathbf{R}(k) = \mathbf{X}^T(k)\mathbf{X}(k)$.

Pre-multiplying (2) with $\mathbf{Q}(k)$ gives

$$\mathbf{Q}(k)\mathbf{e}(k) = \begin{bmatrix} \mathbf{e}_{q1}(k) \\ \mathbf{e}_{q2}(k) \end{bmatrix} = \begin{bmatrix} \mathbf{d}_{q1}(k) \\ \mathbf{d}_{q2}(k) \end{bmatrix} - \begin{bmatrix} \mathbf{0}_{(k+1-N)\times N} \\ \mathbf{U}(k) \end{bmatrix} \mathbf{w}(k) \tag{4}$$

The cost function in (1) is minimized by choosing $\mathbf{w}(k)$ such that $\mathbf{d}_{q2}(k) - \mathbf{U}(k)\mathbf{w}(k)$ is zero, i.e.,

$$\mathbf{w}(k) = \mathbf{U}^{-1}(k)\mathbf{d}_{q2}(k) \tag{5}$$

The QRD-RLS algorithm updates vector $\mathbf{d}_{q2}(k)$ and matrix $\mathbf{U}(k)$ as follows

$$\begin{bmatrix} e_{q1}(k) \\ \mathbf{d}_{q2}(k) \end{bmatrix} = \mathbf{Q}_\theta(k) \begin{bmatrix} d(k) \\ \lambda^{1/2}\mathbf{d}_{q2}(k-1) \end{bmatrix} \tag{6}$$

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

where $\mathbf{Q}_\theta(k) \in \mathbb{R}^{(N+1)\times(N+1)}$ is a sequence of Givens rotation matrices which annihilates the input vector $\mathbf{x}(k)$ in (7) and can be partitioned as [4]

$$\mathbf{Q}_\theta(k) = \begin{bmatrix} \gamma(k) & \mathbf{g}^{\mathrm{T}}(k) \\ \mathbf{f}(k) & \mathbf{E}(k) \end{bmatrix} \tag{8}$$

The QRD-RLS algorithm is complete with the definition of the *a priori* error value $e(k) = e_{q1}(k)/\gamma(k)$ where $\gamma(k)$ is a scalar found in matrix $\mathbf{Q}_\theta(k)$, see (8).

2.2. The FQR_PRI_B algorithm

The idea of an FQRD-RLS algorithm is to replace the matrix update equation (7) with a vector update equation. The *a priori* error is given as

$$\begin{aligned} e(k) &= d(k) - \mathbf{w}^{\mathrm{T}}(k-1)\mathbf{x}(k) \\ &= d(k) - \mathbf{d}_{q2}^{\mathrm{T}}(k-1)\underbrace{\mathbf{U}^{-\mathrm{T}}(k-1)\mathbf{x}(k)}_{\lambda^{1/2}\mathbf{a}(k)} \end{aligned} \tag{9}$$

where $\mathbf{g}(k) = -\gamma(k)\mathbf{a}(k)$, $\mathbf{g}(k)$ as found in (8).

In the FQR_PRI_B algorithm [7, 8], the update equation for vector $\mathbf{a}(k)$ is used. The update equation, obtained by using forward and backward prediction equations and applying rotation matrices to triangularize the data matrix, is given by

$$\begin{bmatrix} \frac{e_b(k)}{\lambda^{1/2}\|\mathbf{e}_b(k-1)\|} \\ \mathbf{a}(k) \end{bmatrix} = \mathbf{Q}_{\theta f}(k-1) \begin{bmatrix} \mathbf{a}(k-1) \\ \frac{e_f(k)}{\lambda^{1/2}\|\mathbf{e}_f(k-1)\|} \end{bmatrix} \tag{10}$$

The FQR_PRI_B algorithm is given at the beginning of Table 1. See [6] and [7] for two different implementations (versions) of the same algorithm. Note that this algorithm was used here but a similar weight extraction procedure can be derived for the FQRD-RLS algorithm based on *a posteriori* backward prediction errors, FQR_POS_B of [5] and [6].

3. NOVEL WEIGHT EXTRACTION METHOD

The novel weight extraction (WE) technique presented below can be invoked at any iteration of the conventional FQRD-RLS algorithm. The internal variables of the FQRD-RLS algorithm at the time of interest are computed in a serial manner, i.e., N iterations for an N coefficient vector.

Consider the output of the adaptive filter $y(k)$ given as

$$y(k) = \mathbf{w}^{\mathrm{T}}(k-1)\mathbf{x}(k) = \mathbf{d}_{q2}^{\mathrm{T}}(k-1)\mathbf{U}^{-\mathrm{T}}(k-1)\mathbf{x}(k) \tag{11}$$

Let us define $\boldsymbol{\delta}_i = \begin{bmatrix} 0 & \dots & 0 & 1 & 0 & \dots & 0 \end{bmatrix}^{\mathrm{T}}$ to be a vector containing a '1' at the i^{th} position. We can now get the i^{th} coefficient of vector $\mathbf{w}(k-1)$ as

$$w_i(k-1) = \mathbf{d}_{q2}^{\mathrm{T}}(k-1)\mathbf{U}^{-\mathrm{T}}(k-1)\boldsymbol{\delta}_i = \mathbf{d}_{q2}^{\mathrm{T}}(k-1)\mathbf{u}_i(k-1) \tag{12}$$

where $\mathbf{u}_i(k-1)$ denotes the i^{th} column of matrix $\mathbf{U}^{-\mathrm{T}}(k-1)$. This means that when $\mathbf{d}_{q2}(k-1)$ is given, the elements of the weight vector $\mathbf{w}(k-1)$ can be computed if all the columns of matrix $\mathbf{U}^{-\mathrm{T}}(k-1)$ are known. Using the following two lemmas we show how all the column vectors $\mathbf{u}_i(k-1)$ can be obtained in a serial manner given $\mathbf{u}_0(k-1)$. The main result is that the column vector $\mathbf{u}_i(k-1)$ can be obtained from the column vector $\mathbf{u}_{i-1}(k-1)$.

Lemma 1. *Let* $\mathbf{u}_i^T(k) = \begin{bmatrix} u_{i,0}(k) & \dots & u_{i,N-1}(k) \end{bmatrix}^T \in \mathbb{R}^{N\times 1}$ *denote the* i^{th} *column of the upper triangular matrix* $\mathbf{U}^{-T}(k) \in \mathbb{R}^{N\times N}$. *Given* $\mathbf{Q}_\theta(k-1) \in \mathbb{R}^{(N+1)\times(N+1)}$ *from Table 1, then* $\mathbf{u}_i(k-2)$ *can be obtained from* $\mathbf{u}_i(k-1)$ *using the relation below*

$$\begin{bmatrix} 0 \\ \lambda^{-1/2}\mathbf{u}_i(k-2) \end{bmatrix} = \mathbf{Q}_\theta^T(k-1) \begin{bmatrix} z_i \\ \mathbf{u}_i(k-1) \end{bmatrix}, \; i = 0, \dots, N-1 \tag{13}$$

where $z_i = -\mathbf{f}^T(k-1)\mathbf{u}_i(k-1)/\gamma(k-1)$.

Lemma 2. *Let* $\mathbf{u}_i(k) = \begin{bmatrix} u_{i,0}(k) & \dots & u_{i,N-1}(k) \end{bmatrix}^T \in \mathbb{R}^{N\times 1}$ *denote the* i^{th} *column of the upper triangular matrix* $\mathbf{U}^{-T}(k-1) \in \mathbb{R}^{N\times N}$. *Given* $\tilde{\mathbf{Q}}_{\theta f}(k) \in \mathbb{R}^{(N+1)\times(N+1)}$ *from Table 1, then* $\mathbf{u}_i(k-1)$ *can be obtained from* $\mathbf{u}_{i-1}(k-2)$ *using the following relation*

$$\begin{bmatrix} \frac{-w_{b,i-1}(k-1)}{\|\mathbf{e}_b(k-1)\|} \\ \mathbf{u}_i(k-1) \end{bmatrix} = \tilde{\mathbf{Q}}_{\theta f}(k-1) \begin{bmatrix} \mathbf{u}_{i-1}(k-2) \\ \frac{-w_{f,i-1}(k-1)}{\|\mathbf{e}_f(k-1)\|} \end{bmatrix}, \; i = 0, \dots, N-1 \tag{14}$$

where

$$w_{f,i-1} = \begin{cases} -1 & \text{for } i = 0 \\ \mathbf{u}_{i-1}^T(k-2)\mathbf{d}_{fq2}(k-1) & \text{otherwise} \end{cases} \tag{15}$$

and $\mathbf{u}_{-1}(k-2) = \mathbf{0}_{N\times 1}$.

The proofs of Lemmas 1 and 2 are in the Appendix. In order to extract the implicit weights $\mathbf{w}(k-1)$ of the FQRD-RLS, Lemma 2 is initialized with $\mathbf{u}_{-1}(k-2)$, and as a result we get column vector $\mathbf{u}_0(k-1)$. Lemma 1 is then invoked to compute column vector $\mathbf{u}_0(k-2)$. From $\mathbf{u}_0(k-2)$ we can compute $\mathbf{u}_1(k-1)$ using Lemma 2. By induction we can conclude that all $\mathbf{u}_i(k-1)$ can be obtained. The procedure is illustrated in Fig 1. As a consequence, the elements of $\mathbf{w}(k-1)$ can be obtained from (12) in a serial manner. The WE algorithm is summarized in Table 1.

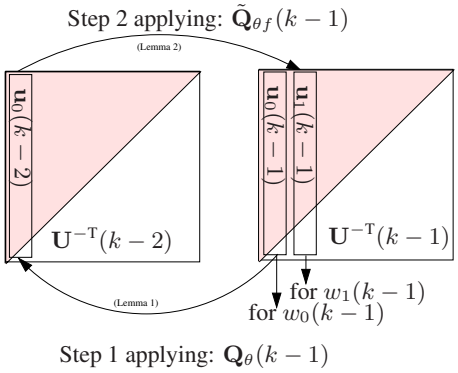

Fig. 1: The procedure for updating $\mathbf{u}_i(k-1)$ for weight extraction.

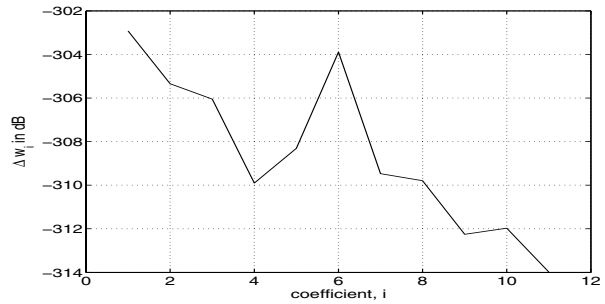

Fig. 2: Weight difference of the algorithms.

The number of operations required to completely extract all the coefficients is given in Table 2. For comparison, the computational cost of the FQRD-RLS algorithm based on *a priori* backward prediction errors and the Inverse QRD-RLS algorithm updates are also given.

4. SIMULATIONS

In this section, the proposed WE algorithm is applied to the FQRD-RLS algorithm in a system identification setup. The plant has $N = 11$ coefficients and a colored noise input signal was used with SNR set to 30 dB. The condition number of the input-signal autocorrelation matrix is 821. The extracted weights of the FQRD-RLS algorithm are compared to those of the IQRD-RLS algorithm [2] which, with proper initialization, provides an identical solution. As a measure of accuracy, the squared weight-difference from both algorithms was calculated and averaged over $K = 100$ ensemble using

$$\Delta\bar{w}_i = \frac{1}{K}\sum_{j=0}^{K-1}[w^j_{IQRD,i} - w^j_{FQRD,i}]^2 \qquad (16)$$

where for the j^{th} simulation run, $w^j_{IQRD,i}$ and $w^j_{FQRD,i}$ are the i^{th} coefficients of the IQRD-RLS and the FQRD-RLS algorithms, respectively. Figure 2 shows that the difference between the extracted weights of the FQRD-RLS and those of the IQRD-RLS are with in machine precision. The learning-curves for both algorithms are plotted in Figure 3. As can be seen from the figure, they are identical up to numerical accuracy.

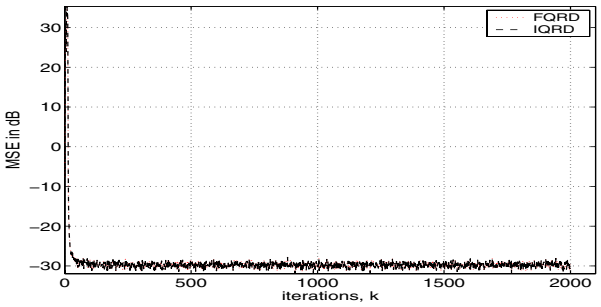

Fig. 3: Learning-curve for FQRD-RLS and IQRD-RLS algorithm.

5. CONCLUSIONS

This paper showed how to reuse the internal variables of the fast QRD-RLS (FQRD-RLS) algorithm to extract the weights in a serial manner. The presented technique enables new applications of the FQRD-RLS algorithm which are different to the standard output-error type applications. The new weight extraction technique was used in a system identification setup to make the weight vector explicitly available. The results were compared with those using a design based on the inverse QRD-RLS algorithm. It was verified that identical results are obtained using the proposed design method at a much lower computational cost.

6. APPENDIX

Proof of Lemma 1:
The update equation for $\mathbf{U}^{-\mathrm{T}}(k-2)$ in the IQRD-RLS algorithm is given by

$$\begin{bmatrix}\mathbf{z}^{\mathrm{T}}(k-1)\\ \mathbf{U}^{-\mathrm{T}}(k-1)\end{bmatrix} = \mathbf{Q}_\theta(k-1)\begin{bmatrix}\mathbf{0}^{\mathrm{T}}\\ \lambda^{-1/2}\mathbf{U}^{-\mathrm{T}}(k-2)\end{bmatrix} \qquad (17)$$

where $\mathbf{z}(k-1) = \gamma^{-1}(k-1)\mathbf{f}^{\mathrm{T}}(k-1)\mathbf{U}^{-\mathrm{T}}(k-1)$. Premultiplying both sides with $\mathbf{Q}^{\mathrm{T}}_\theta(k-1)$ and considering each column we get

$$\begin{bmatrix}0\\ \lambda^{-1/2}\mathbf{u}_i(k-2)\end{bmatrix} = \mathbf{Q}^{\mathrm{T}}_\theta(k-1)\begin{bmatrix}z_i(k-1)\\ \mathbf{u}_i(k-1)\end{bmatrix} \qquad (18)$$

where $z_i(k-1)$ is the i^{th} element of vector $\mathbf{z}(k)$

$$z_i(k-1) = -\mathbf{f}^{\mathrm{T}}(k-1)\mathbf{u}_i(k-1)/\gamma(k-1) \qquad (19)$$

and the elements of vector $\mathbf{f}(k-1)$ and $\gamma(k-1)$ are obtained from the rotation matrix $\mathbf{Q}_\theta(k-1)$ as

$$\begin{bmatrix}\gamma(k-1)\\ \mathbf{f}(k-1)\end{bmatrix} = \mathbf{Q}_\theta(k-1)\begin{bmatrix}1\\ \mathbf{0}_{N\times 1}\end{bmatrix} \qquad (20)$$

Eq. (20) needs only to be evaluated once at the beginning of the weight extraction procedure.
Proof of Lemma 2:
The FQRD-RLS algorithm of Table 1 updates $\mathbf{a}(k)$ at every iteration as follows

$$\begin{bmatrix}\frac{e_b(k)}{\lambda^{1/2}\|\mathbf{e}_b(k-1)\|}\\ \mathbf{a}(k)\end{bmatrix} = \tilde{\mathbf{Q}}_{\theta f}(k-1)\begin{bmatrix}\mathbf{a}(k-1)\\ \frac{e_f(k)}{\lambda^{1/2}\|\mathbf{e}_f(k-1)\|}\end{bmatrix} \qquad (21)$$

where $e_b(k)$ and $e_f(k)$ are the backward and the forward prediction error values given by [2]:

$$\begin{aligned} e_f(k) &= x(k) - \mathbf{w}_f^{\mathrm{T}}(k-1)\mathbf{x}(k-1) \\ e_b(k) &= x(k-N-1) - \mathbf{w}_b^{\mathrm{T}}(k)\mathbf{x}(k) \end{aligned} \tag{22}$$

with $\mathbf{w}_f(k) = \mathbf{U}^{-T}(k)\mathbf{d}_{fq2}(k)$ [2] and $\mathbf{w}_b(k)$ denoting the forward and backward prediction weight vectors, respectively. Using Equation (22), the definition of $\mathbf{a}(k) = \lambda^{-1/2}\mathbf{U}^{-\mathrm{T}}(k-1)\mathbf{x}(k)$, and removing scalars and vectors related to input signal $x(k)$, the following relation is obtained from Equation (21)

$$\begin{aligned} &\begin{bmatrix} \frac{-\mathbf{w}_b^{\mathrm{T}}(k)}{\|\mathbf{e}_b(k-1)\|} & \frac{1}{\|\mathbf{e}_b(k-1)\|} \\ \mathbf{U}^{-\mathrm{T}}(k-1) & \mathbf{0} \end{bmatrix} \\ &= \tilde{\mathbf{Q}}_{\theta f}(k-1) \begin{bmatrix} \mathbf{0} & \mathbf{U}^{-\mathrm{T}}(k-2) \\ \frac{1}{\|\mathbf{e}_f(k-1)\|} & \frac{-\mathbf{w}_f^{\mathrm{T}}(k-1)}{\|\mathbf{e}_f(k-1)\|} \end{bmatrix} \end{aligned} \tag{23}$$

Considering the partition of matrix $\mathbf{U}^{-\mathrm{T}}(k-1)$ into its column vectors $\mathbf{u}_i(k-1)$, the column version of (23) becomes

$$\begin{bmatrix} \frac{-w_{b,i}(k-1)}{\|\mathbf{e}_b(k-1)\|} \\ \mathbf{u}_i(k-1) \end{bmatrix} = \tilde{\mathbf{Q}}_{\theta f}(k-1) \begin{bmatrix} \mathbf{u}_{i-2}(k-2) \\ \frac{-w_{f,i-1}(k-1)}{\|\mathbf{e}_f(k-1)\|} \end{bmatrix}, i = 0, \ldots, N-1 \tag{24}$$

where $w_{b,i}(k)$ and $w_{f,i}(k-1)$ are the i^{th} elements of the forward and backward prediction weight vectors, respectively. To account for the first column of (23) we initialize with $\mathbf{u}_{-1}(k-2) = \mathbf{0}_{N\times 1}$ and $w_{f,-1}(k-1) = -1$.

7. REFERENCES

[1] J. Cioffi, "Limited-precision effects in adaptive filtering," *IEEE Trans. on Circuits and Syst.*, vol. 34, no. 7, pp. 821 – 833, July, 1987.

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

[3] F. Ling, "Givens rotation based least squares lattice and related algorithms," *IEEE Trans. Signal Processing*, vol. SP-39, no. 7, pp. 1541–1551, July 1991.

[4] J. A. Apolinário Jr., P. S. R. Diniz, "A new fast QR algorithm based on *a priori* errors," *IEEE Signal Processing Lett.*, vol. 4, no. 11, pp. 307–309, Nov. 1997.

[5] M. G. Bellanger, "The FLS-QR algorithm for adaptive filtering," *Signal Processing*, vol. 17, pp. 291–304, March 1984.

[6] J. A. Apolinário Jr., M. G. Siqueira, P. S. R. Diniz, "The fast QR algorithms based on backward prediction errors: a new implementation and its finite precision performance," *CSSP Birkhuser journal*, Vol. 22, No. 4, pp. 335–349, 2003.

[7] M. D. Miranda, M. Gerken, "A hybrid least squares QR-lattice algorithm using a priori errors," *IEEE Trans. Signal Processing*, vol. 45, no. 12, pp. 2900–2911, Dec. 1997.

[8] A. A. Rontogiannis, S. Theodoridis, "New fast inverse QR least squares adaptive algorithms," in *ICASSP*, Detroit, MI, USA, IEEE, pp. 1412–1415, May, 1995.

[9] C. R. Ward *et al.*, "Application of a systolic array to adaptive beamforming," in *IEE Proceedings*, vol. 131, London, England, pp. 638–645, 1984.

[10] P. A. Regalia, M. G. Bellanger, "On the duality between fast QR methods and lattice methods in least squares adaptive filtering," *IEEE Trans. Signal Processing*, vol. 39, no. 4, pp. 876–891, Apr. 1991.

Table 1: Weight extraction algorithm.

Conventional FQR_PRI_B algorithm
for each k
{ Obtaining $\mathbf{d}_{fq2}(k)$:
$\begin{bmatrix} e_{fq1}(k) \\ \mathbf{d}_{fq2}(k) \end{bmatrix} = \mathbf{Q}_\theta(k-1) \begin{bmatrix} x(k) \\ \lambda^{1/2}\mathbf{d}_{fq2}(k-1) \end{bmatrix}$
Obtaining $\mathbf{a}(k)$:
$\begin{bmatrix} \frac{e_b(k)}{\lambda^{1/2}\|\mathbf{e}_b(k-1)\|} \\ \mathbf{a}(k) \end{bmatrix} = \mathbf{Q}_{\theta f}(k-1) \begin{bmatrix} \mathbf{a}(k-1) \\ \frac{e_f(k)}{\lambda^{1/2}\|\mathbf{e}_f(k-1)\|} \end{bmatrix}$
Obtaining $\|\mathbf{e}_f(k)\|$:
$\|\mathbf{e}_f(k)\| = \sqrt{e_{fq1}^2(k) + \lambda\|\mathbf{e}_f(k-1)\|^2}$
Obtaining $\mathbf{Q}_{\theta f}(k)$:
$\begin{bmatrix} \mathbf{0} \\ \|\mathbf{e}_f^{(0)}(k)\| \end{bmatrix} = \mathbf{Q}_{\theta f}(k) \begin{bmatrix} \mathbf{d}_{fq2}(k) \\ \|\mathbf{e}_f(k)\| \end{bmatrix}$
Obtaining $\mathbf{Q}_\theta(k)$:
$\begin{bmatrix} 1/\gamma(k) \\ \mathbf{0} \end{bmatrix} = \mathbf{Q}_\theta(k) \begin{bmatrix} 1 \\ -\mathbf{a}(k) \end{bmatrix}$
Joint Process Estimation:
$\begin{bmatrix} e_{q1}(k) \\ \mathbf{d}_{q2}(k) \end{bmatrix} = \mathbf{Q}_\theta(k) \begin{bmatrix} d(k) \\ \lambda^{1/2}\mathbf{d}_{q2}(k-1) \end{bmatrix}$
$e(k) = e_{q1}(k)/\gamma(k)$
}
Weight extraction at any chosen time instant k
Initializing $w_{f,-1}(k-1)$:
$w_{f,-1}(k-1) = -1$
Obtaining $\mathbf{f}(k-1)$:
$\begin{bmatrix} \gamma(k-1) \\ \mathbf{f}(k-1) \end{bmatrix} = \mathbf{Q}_\theta(k-1) \begin{bmatrix} 1 \\ \mathbf{0}_{N\times 1} \end{bmatrix}$
for each $i = 0 : N-1$
{ Obtaining $\mathbf{u}_i(k-1)$:
$\begin{bmatrix} \frac{-w_{b,i}(k-1)}{\|\mathbf{e}_b(k-1)\|} \\ \mathbf{u}_i(k-1) \end{bmatrix} = \tilde{\mathbf{Q}}_{\theta f}(k-1) \begin{bmatrix} \mathbf{u}_{i-1}(k-2) \\ \frac{-w_{f,i-1}(k-1)}{\|\mathbf{e}_f(k-1)\|} \end{bmatrix}$
Obtaining $z_i(k-1)$:
$z_i(k-1) = -\mathbf{f}^{\mathrm{T}}(k-1)\mathbf{u}_i(k-1)/\gamma(k-1)$
Obtaining $\mathbf{u}_i(k-2)$:
$\begin{bmatrix} 0 \\ \lambda^{-1/2}\mathbf{u}_i(k-2) \end{bmatrix} = \mathbf{Q}_\theta^{\mathrm{T}}(k-1) \begin{bmatrix} z_i(k-1) \\ \mathbf{u}_i(k-1) \end{bmatrix}$
Obtaining the coefficients $w_{f,i-1}(k-1)$:
$w_{f,i}(k-1) = \mathbf{u}_i^{\mathrm{T}}(k-2)\mathbf{d}_{fq2}(k-1)$
Obtaining the coefficients:
$w_i(k-1) = \mathbf{u}_i^{\mathrm{T}}(k-1)\mathbf{d}_{q2}(k-1)$
}

Table 2: Computational complexity of weight extraction (WE).

$ALG. \times OPER.$	MULT	DIV	SQRT
FQR_PRI_B	$19N + 4$	$4N + 1$	$2N + 1$
WE (per weight i)	$16N - 6 - 14i$	1	0
WE (total)	$7N^2 + N$	1	0
IQRD-RLS	$3N^2 + 2N + 1$	$2N$	N