Multichannel fast QRD-RLS adaptive filtering: block-channel and sequential-channel algorithms based on updating backward prediction errors

A. L. L. Ramos[a], J. A. Apolinário Jr.[a,*], and S. Werner[b]

[a]**Instituto Militar de Engenharia**
Department of Electrical Engineering
Praça General Tibúrcio, 80 – Urca
22290-270 Rio de Janeiro, RJ – Brazil

[b]**Helsinki University of Technology**
Signal Processing Laboratory
P.O. Box 3000
FIN-02015, TKK – Finland

Abstract

Fast QR decomposition recursive least-squares (FQRD-RLS) algorithms are well known for their fast convergence and reduced computational complexity. A considerable research effort has been devoted to the investigation of single-channel versions of the FQRD-RLS algorithms, while the multichannel counterparts have not received the same attention. The goal of this paper is to broaden the study of the efficient and low complexity family of multi-channel RLS adaptive filters, and to offer new algorithm options. We present a generalized approach for block-type multichannel FQRD-RLS (MC-FQRD-RLS) algorithms that include both cases of equal and multiple order. We also introduce new versions for block-channel and sequential-channel processing, details of their derivations, and a comparison in terms of computational complexity. The proposed algorithms are based on the updating of backward *a priori* and *a posteriori* error vectors, which are known to be numerically robust.

Key words: Adaptive systems, Fast algorithms, QR decomposition, Multichannel algorithms, Order recursive algorithms.

* Corresponding author.
Email addresses: `alopesramos@yahoo.com.br` (A. L. L. Ramos), `apolin@ieee.org` (J. A. Apolinário Jr.), `stefan.werner@tkk.fi` (S. Werner).

1 Introduction

Multichannel signal processing can be found in a large variety of applications such as color image processing, multi-spectral remote sensing imagery, biomedicine, channel equalization, stereophonic echo cancellation, multidimensional signal processing, Volterra-type nonlinear system identification, and speech enhancement [1,2].

When considering multichannel fast converging adaptive implementations, it is many times possible to apply standard single-channel algorithms directly to the multichannel problem, e.g., the numerically stable QR decomposition recursive least-squares (QRD-RLS) algorithm. Although such a solution would present fast convergence, it may be computationally prohibitive due a possible large number of coefficients. In order to reduce computational complexity, it is necessary to derive stable algorithms, specially tailored for the multichannel setup. Multichannel adaptive filtering algorithms can be derived using two distinct approaches: 1) *block-type approach* where the channels are processed simultaneously, and; 2) *sequential approach* that processes each channel individually [3].

In this paper, we are concerned with so-called *multichannel fast QR decomposition RLS* (MC-FQRD-RLS) algorithms. The MC-FQRD-RLS algorithms exhibit RLS convergence and numerical robustness at a lower complexity than the QRD-RLS algorithm which can be used in a single-channel adaptive filter, a multi-channel adaptive filter, or an adaptive linear combiner. The main idea of MC-FQRD-RLS algorithms is to exploit the underlying time-shift structure of the input-signal vector of each channel in order to replace matrix update equations with vector update equations. By doing so, the computational complexity can be reduced from $\mathcal{O}(P^2)$ of the standard QRD-RLS implementation to $\mathcal{O}(M^cP)$, where P is the total number of filter coefficients, M is the number of channels, and c is an integer constant that depends on the algorithm approach taken. To be more explicit, the computational complexities associated with the block-channel and the sequential-channel approaches are $\mathcal{O}(M^2P)$ and $\mathcal{O}(MP)$, respectively. That is, taking a sequential-channel approach will render the lowest computational complexity. The main advantage of the block-channel approach is that these algorithms favor parallel processing implementations.

A classification of single-channel FQRD-RLS algorithms is shown in Table 1 (it will be later extended to the multichannel case). The algorithm classification is made according to which error vector is updated (*a priori* or *a posteriori*) and the type of prediction used (forward or backward). A unified framework for fixed-order block-channel MC-FQRD-RLS algorithms was addressed in [4], where two new block-channel versions were introduced. The basic *a posteriori* and *a priori* versions based on updating the backward prediction errors were proposed in [5] and [6], respectively. In [5], a transversal block-channel version and a sequential-channel approach for the same algorithm were presented. In [6], both block-channel and sequential-channel *a priori* versions were treated in detail and three new algorithms were introduced for the cases of equal and unequal channel orders: an order recursive version for the block-channel case, a transversal version, and an order recursive version for the sequential-channel case.

This work introduces new MC-FQRD-RLS algorithms based on the updating of *a posteriori* backward prediction errors, which have a lower computational complexity than their *a priori* counterparts. In particular, we present a general approach for deriving block-channel based multichannel multiple-order fast QRD-RLS algorithms. Adopting the same structure of the input signal vector from [6], two new block-channel algorithms (*a priori* and *a posteriori* versions) are introduced. The former algorithm has, to the authors' knowledge, not been published in literature. The *a posteriori* version bears similarities with the algorithm in [3], however, derived from an input signal matrix with a different structure. The results obtained are then particularized for the equal channel order case which enables us to derive a new order-recursive version based on the updating of the *a posteriori* backward error vector. Finally, we introduce the *a posteriori* counterparts of the *a priori* transversal and order-recursive multiple order sequential-channel versions of [6]. A classification of the MC-FQRD-RLS algorithm is then provided which include all known algorithms.

The paper is organized as follows. The basic equations of the multichannel QRD-RLS algorithms are presented in Section 2. Section 3 provides the details of the alternative input vector used in [6] that facilitates the derivation of algorithms for the case of channels with multiple orders. Section 4 presents the general framework for block-channel algorithms, and two extended block-channel algorithms are introduced in Section 5. The new block-channel order-recursive version for the equal channel order case is derived in Section 6. Section 7 presents the new transversal and order-recursive sequential-channel counterparts of the algorithms presented in [6]. Simulation results and computational complexity issues are discussed in Section 8. Conclusions are presented in Section 9.

Table 1
Classification of single-channel FQRD-RLS algorithms.

Error type	Prediction	
	forward	backward
a posteriori	FQRD_POS_F [7]	**FQRD_POS_B** [8,9]
a priori	FQRD_PRI_F [10]	**FQRD_PRI_B** [11,12]

2 Fundamentals

The multichannel algorithms of the fast QRD-RLS family use the weighted least-squares (WLS) objective function defined as [13]

$$\xi(k) = \sum_{i=0}^{k} \lambda^{k-i}[d(i) - \boldsymbol{x}_P^T(i)\boldsymbol{w}_P(k)]^2 = \boldsymbol{e}^T(k)\boldsymbol{e}(k) \tag{1}$$

where $\boldsymbol{e}(k)$ is the weighted error vector of the form

$$\begin{aligned}\boldsymbol{e}(k) &= \begin{bmatrix} d(k) \\ \lambda^{1/2}d(k-1) \\ \vdots \\ \lambda^{k/2}d(0) \end{bmatrix} - \begin{bmatrix} \boldsymbol{x}_P^T(k) \\ \lambda^{1/2}\boldsymbol{x}_P^T(k-1) \\ \vdots \\ \lambda^{k/2}\boldsymbol{x}_P^T(0) \end{bmatrix} \boldsymbol{w}_P(k) \\ &= \boldsymbol{d}(k) - \boldsymbol{X}_P(k)\boldsymbol{w}_P(k) \end{aligned} \tag{2}$$

where

$$\boldsymbol{x}_P^T(k) = \begin{bmatrix} \boldsymbol{x}_k^T & \boldsymbol{x}_{k-1}^T & \cdots & \boldsymbol{x}_{k-N+1}^T \end{bmatrix} \tag{3}$$

and $\boldsymbol{x}_k^T = [x_1(k) \quad x_2(k) \quad \cdots \quad x_M(k)]$ is the input signal vector at instant k. Note that N is defined as the number of filter coefficients per channel, M is the number of input channels, and $\boldsymbol{w}_P(k)$ is the $P \times 1$ coefficient vector at time instant k, $P = MN$ being the total number of elements for the case of channels with equal orders.

Let the lower triangular matrix $\boldsymbol{U}_P(k)$ denote the Cholesky factor of $\boldsymbol{X}_P^T(k)\boldsymbol{X}_P(k)$ obtained by applying the Givens rotation matrix $\boldsymbol{Q}_P(k)$ onto $\mathbf{X}_P(k)$. The rotated error $\mathbf{e}_q(k)$ can be expressed as follows.

$$\boldsymbol{e}_q(k) = \boldsymbol{Q}_P(k)\boldsymbol{e}(k) = \begin{bmatrix} \boldsymbol{e}_{q1}(k) \\ \boldsymbol{e}_{q2}(k) \end{bmatrix} = \begin{bmatrix} \boldsymbol{d}_{q1}(k) \\ \boldsymbol{d}_{q2}(k) \end{bmatrix} - \begin{bmatrix} \mathbf{0} \\ \boldsymbol{U}_P(k) \end{bmatrix} \boldsymbol{w}_P(k) \tag{4}$$

Due to the unitary property of $\boldsymbol{Q}_P(k)$, minimizing $\|\boldsymbol{e}_q(k)\|^2$ is identical to minimizing the cost function of (1). Therefore, Equation (1) is minimized by choosing $\boldsymbol{w}_P(k)$ in (4) such that $\boldsymbol{d}_{q2}(k) - \boldsymbol{U}_P(k)\boldsymbol{w}_P(k)$ equals zero, i.e.,

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

To allow a compact formulation of the multiple-order case (different channel orders), an alternative structure of the input signal vector will be defined in the next section.

3 Alternative definition of the input vector

For the case of unequal channel orders, let $N_1, N_2, \cdots, N_M$ be the number of *taps* in the *tapped delay–lines* of each of the M channels and, hereafter, $P = \sum_{r=1}^{M} N_r$ the overall number of taps. Without loss of generality, we assume $N_1 \geq N_2 \geq \cdots \geq N_M$.

Fig. 1 shows an example of a multichannel scenario with $M = 3$ channels of unequal orders where $N_1 = 4$, $N_2 = 3$, $N_3 = 2$, i.e., $P = 4 + 3 + 2 = 9$. The following approach to construct the input vector, $\boldsymbol{x}_P(k)$, was considered in [6] [1]: the first $N_1 - N_2$ *samples* from the first channel are chosen to be the leading elements of $\boldsymbol{x}_P(k)$, followed by $N_2 - N_3$ *pairs of samples* from the first and second channels, followed by $N_3 - N_4$ *triples of samples* of the first three channels and so on till the $N_M - N_{M+1}$ *M–tuples of samples* of all channels. It is assumed that $N_{M+1} = 0$.

The procedure detailed above gives rise to two distinct ways of obtaining the expanded input vector, $\boldsymbol{x}_{P+M}(k + 1)$. The first approach is to shift in all the new samples from the different channels at the same time and process all channels simultaneously. The second approach is to shift in a sample from each channel at a time and progress in a recursive manner from the first to the last channel. The first approach leads to block-type multichannel algorithms which are studied in Sections 4–6. The second approach results in various sequential-type multichannel algorithms like the one derived in Section 7. Before deriving new block-channel and sequential-channel algorithms, we give the necessary details related to the input data vector for each case. Next, we consider the block-channel input vector followed by the one used for the sequential-channel approach.

3.1 Input vector for block-type multichannel algorithms

For the case of block-channel multichannel algorithms, the expanded input vector, $\boldsymbol{x}_{P+M}(k + 1)$, is given by

$$\boldsymbol{x}_{P+M}^T(k+1) = \begin{bmatrix} x_1(k+1) & x_2(k+1) & \cdots & x_M(k+1) & \boldsymbol{x}_P^T(k) \end{bmatrix} \boldsymbol{P} \quad (6)$$

[1] It is worth mentioning that another approach dealing with *unequal-number-of-taps* can be found in [14].

where $\boldsymbol{P} = \boldsymbol{P}_M \boldsymbol{P}_{M-1} \cdots \boldsymbol{P}_1$ is a product of M permutation matrices that moves the most recent sample of the ith channel (for $i = 1, 2, \cdots, M$) to position p_i in vector $\boldsymbol{x}_{P+M}(k+1)$, where

$$p_i = \sum_{r=1}^{i-1} r(N_r - N_{r+1}) + i, \;\; i = 1,\ 2,\ \cdots,\ M. \tag{7}$$

After the above process is terminated, we have $\boldsymbol{x}_{P+M}^T(k+1) = [\boldsymbol{x}_P^T(k+1) \;\; x_1(k - N_1 + 1) \cdots \;\; x_M(k - N_M + 1)]$, such that the first P elements of $\boldsymbol{x}_{P+M}^T(k+1)$ provide the input vector for the next iteration. In order to illustrate the role of the permutation matrix $\mathbf{P}$, let us return to the example depicted in Fig. 1. In this example, the expanded input vector $\mathbf{x}_{P+M}(k+1)$ is obtained by inserting the new samples in positions $p_1 = 1$, $p_2 = 3$, and $p_3 = 6$, respectively, i.e.,

$$\boldsymbol{P}^T \begin{bmatrix} x_1(k+1) \\ x_2(k+1) \\ x_3(k+1) \\ \hline \boldsymbol{x}_P(k) \end{bmatrix} = \boldsymbol{P}^T \begin{bmatrix} x_1(k+1) \\ x_2(k+1) \\ x_3(k+1) \\ \hline x_1(k) \\ x_1(k-1) \\ x_2(k) \\ x_1(k-2) \\ x_2(k-1) \\ x_3(k) \\ x_1(k-3) \\ x_2(k-2) \\ x_3(k-1) \end{bmatrix} = \begin{bmatrix} x_1(k+1) \\ x_1(k) \\ x_2(k+1) \\ x_1(k-1) \\ x_2(k) \\ x_3(k+1) \\ x_1(k-2) \\ x_2(k-1) \\ x_3(k) \\ \hline x_1(k-3) \\ x_2(k-2) \\ x_3(k-1) \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}_P(k+1) \\ \hline x_1(k-3) \\ x_2(k-2) \\ x_3(k-1) \end{bmatrix}. \tag{8}$$

3.2 Input vector for sequential-type multichannel algorithms

For the sequential-channel case, the extended input vector, $\boldsymbol{x}_{P+M}(k+1)$, is constructed from $\boldsymbol{x}_P(k)$ in M successive steps as

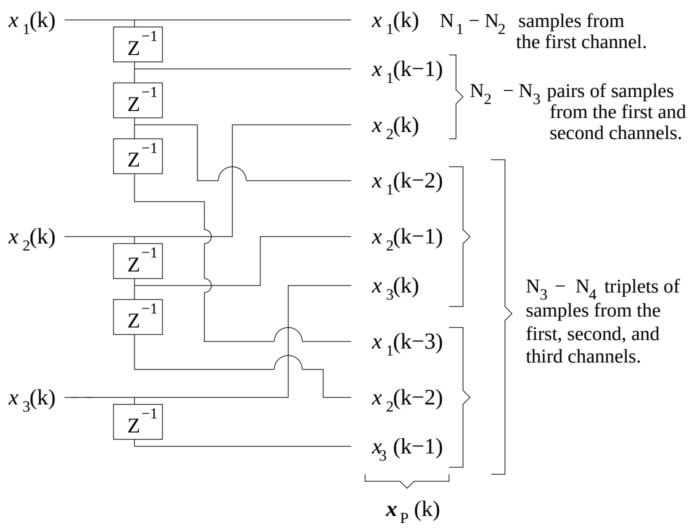
$x_1(k)$
$x_2(k)$
$x_3(k)$
Z^{-1}
$x_1(k)$ $N_1 - N_2$ samples from the first channel.
$x_1(k-1)$
$x_2(k)$
$N_2 - N_3$ pairs of samples from the first and second channels.
$x_1(k-2)$
$x_2(k-1)$
$x_3(k)$
$N_3 - N_4$ triplets of samples from the first, second, and third channels.
$x_1(k-3)$
$x_2(k-2)$
$x_3(k-1)$
$\boldsymbol{x}_P(k)$

Fig. 1. Obtaining the input vector.

$$\boldsymbol{x}_{P+1}^T(k+1) = \begin{bmatrix} x_1(k+1) & \boldsymbol{x}_P^T(k) \end{bmatrix} \tag{9}$$

$$\boldsymbol{x}_{P+i}^T(k+1) = \begin{bmatrix} x_i(k+1) & \boldsymbol{x}_{P+i-1^T(k+1)} \end{bmatrix} \boldsymbol{P}_i \tag{10}$$

where $\boldsymbol{P}_i$ is a permutation matrix which takes the most recent sample $x_i(k+1)$ of the ith channel to position p_i (see Eq. (7)) and left shifts the first $p_i - 1$ elements of $\boldsymbol{x}_{P+i-1}^T(k+1)$. After processing all M channels, the first P elements of the updated extended input vector constitute the input vector of the next iteration, i.e., $\boldsymbol{x}_{P+M}^T(k+1) = [\boldsymbol{x}_P^T(k+1) \quad x_1(k-N_1+1) \quad \cdots \quad x_M(k-N_M+1)]$.

4 Block-type Multichannel Fast QRD-RLS Algorithms

This section presents a general framework for block-type multichannel algorithms using the extended input signal vector $\boldsymbol{x}_{P+M}(k+1)$ defined in Subsection 3.1. Algorithms that can be derived using the formulation provided here are presented in the following sections.

The expanded input data matrix $\bar{\boldsymbol{X}}_{P+M}(k+1)$ can be defined as[2]

$$\bar{\boldsymbol{X}}_{P+M}(k+1) = \begin{bmatrix} \boldsymbol{x}^T_{P+M}(k+1) \\ \lambda^{1/2}\boldsymbol{x}^T_{P+M}(k) \\ \vdots \\ \lambda^{(k+1)/2}\boldsymbol{x}^T_{P+M}(0) \\ \mathbf{0}_{(M-1)\times(P+M)} \end{bmatrix} = \left[\begin{array}{c|c} \boldsymbol{D}_f(k+1) & \begin{array}{c} \boldsymbol{X}_P(k) \\ \mathbf{0}^T \end{array} \\ \hline \multicolumn{2}{c}{\mathbf{0}_{(M-1)\times(P+M)}} \end{array}\right] \boldsymbol{P} \quad (11)$$

In order to triangularize (11) and obtain $\boldsymbol{U}_{P+M}(k+1)$, three sets of Givens rotation matrices $\boldsymbol{Q}(k)$, $\boldsymbol{Q}_f(k+1)$, and $\boldsymbol{Q}'_f(k+1)$ are needed [5,6,15]. The role of each matrix in the triangularization process is illustrated in the following equation.

$$\begin{aligned} &\boldsymbol{Q}'_f(k+1)\boldsymbol{Q}_f(k+1)\boldsymbol{Q}(k)\bar{\boldsymbol{X}}_{P+M}(k+1) = \\ &= \boldsymbol{Q}'_f(k+1)\boldsymbol{Q}_f(k+1) \begin{bmatrix} \boldsymbol{E}_{fq1}(k+1) & \mathbf{0} \\ \boldsymbol{D}_{fq2}(k+1) & \boldsymbol{U}_P(k) \\ \lambda^{(k+1)/2}\boldsymbol{x}^T_0 & \mathbf{0}^T \\ \multicolumn{2}{c}{\mathbf{0}_{(M-1)\times(P+M)}} \end{bmatrix} \mathbf{P} \\ &= \boldsymbol{Q}'_f(k+1) \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \boldsymbol{D}_{fq2}(k+1) & \boldsymbol{U}_P(k) \\ \boldsymbol{E}_f(k+1) & \mathbf{0} \end{bmatrix} \boldsymbol{P} \end{aligned} \quad (12)$$

In (12), $\boldsymbol{Q}(k)$ contains $\mathbf{Q}_P(k)$ as a submatrix which triangulates $\boldsymbol{X}_P(k)$, generating $\boldsymbol{U}_P(k)$. Matrix $\boldsymbol{Q}_f(k+1)$ is responsible for the zeroing of matrix $\mathbf{E}_{fq1}(k+1)$. Note that, when working with fixed-order (or fixed-dimension, as opposed to the ever increasing dimension of $\boldsymbol{Q}_P(k)$, for instance), this is equivalent to annihilating $\boldsymbol{e}^T_{fq1}(k+1)$, the first row of $\boldsymbol{E}_{fq1}(k+1)$, against the diagonal of $\lambda^{1/2}\boldsymbol{E}_f(k)$, generating $\boldsymbol{E}_f(k+1)$. This is shown in (14).

From (12) and using the fixed-order matrices $\boldsymbol{Q}_\theta(k)$ embedded in $\boldsymbol{Q}_P(k)$ and $\overline{\boldsymbol{Q}}_f(k+1)$ embedded in $\boldsymbol{Q}_f(k+1)$, one can, after some algebraic work, obtain

[2] Note that $\bar{\mathbf{X}}_{P+M}(k+1)$ was formed by adding $M-1$ rows of zeros to $\mathbf{X}_{P+M}(k+1)$ such that $\mathbf{U}_{P+M}(k+1)$ has the correct dimension in (16), i.e., $(P+M)\times(P+M)$.

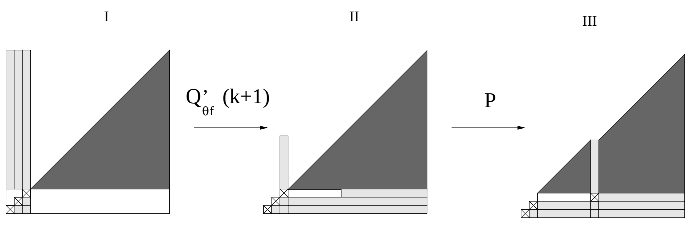

Fig. 2. Obtaining the lower triangular $\boldsymbol{U}_{P+M}(k+1)$.

the following equations.

$$\begin{bmatrix} \boldsymbol{e}_{fq1}^T(k+1) \\ \boldsymbol{D}_{fq2}(k+1) \end{bmatrix} = \boldsymbol{Q}_\theta(k) \begin{bmatrix} \boldsymbol{x}_{k+1}^T \\ \lambda^{1/2}\boldsymbol{D}_{fq2}(k) \end{bmatrix} \tag{13}$$

$$\begin{bmatrix} \boldsymbol{0}^T \\ \boldsymbol{E}_f(k+1) \end{bmatrix} = \overline{\boldsymbol{Q}}_f(k+1) \begin{bmatrix} \boldsymbol{e}_{fq1}^T(k+1) \\ \lambda^{1/2}\boldsymbol{E}_f(k) \end{bmatrix} \tag{14}$$

In (13), $\boldsymbol{x}_{k+1}^T = [x_1(k+1) \quad x_2(k+1) \cdots x_M(k+1)]$ is the *forward reference signal* and $\boldsymbol{e}_{fq1}^T(k+1)$ is the *rotated forward error*; in (14), $\boldsymbol{E}_f(k+1)$ is the $M \times M$ *forward prediction error covariance* (lower-triangular) matrix.

Removing the ever-increasing null section in (12) and using the fixed-order matrix $\boldsymbol{Q}'_{\theta f}(k+1)$ embedded in $\boldsymbol{Q}'_f(k+1)$, we obtain

$$\bar{\boldsymbol{U}}_{P+M}(k+1) = \boldsymbol{Q}'_{\theta f}(k+1) \begin{bmatrix} \boldsymbol{D}_{fq2}(k+1) & \boldsymbol{U}_P(k) \\ \boldsymbol{E}_f(k+1) & \boldsymbol{0} \end{bmatrix} \boldsymbol{P}. \tag{15}$$

Note that the permutation matrix $\boldsymbol{P}$ in (15) prevents a direct annihilation of the first M columns — corresponding to matrix $\boldsymbol{D}_{fq2}(k+1) = [\boldsymbol{d}_{fq2}^{(1)}(k+1) \ \boldsymbol{d}_{fq2}^{(2)}(k+1) \ \cdots \ \boldsymbol{d}_{fq2}^{(M)}(k+1)]$ — against the anti-diagonal of $\boldsymbol{E}_f(k+1)$ using the set of Givens rotations $\boldsymbol{Q}'_{\theta f}(k+1) = \boldsymbol{Q}'_{\theta f}{}^{(M)}(k+1) \cdots \boldsymbol{Q}'_{\theta f}{}^{(2)}(k+1)\boldsymbol{Q}'_{\theta f}{}^{(1)}(k+1)$. Also from (15) it can be seen that this permutation factor, $\boldsymbol{P} = \boldsymbol{P}_M\boldsymbol{P}_{M-1}\cdots\boldsymbol{P}_1$, will right-shift the first M columns to position p_i, for $i = M$ to 1, in this order. Thus, only the first $P + i - p_i$ elements of each $\boldsymbol{d}_{fq2}^{(i)}(k+1)$ will be rotated against the anti-diagonal of $\boldsymbol{E}_f(k+1)$ using the set of Givens rotations in $\boldsymbol{Q}'_{\theta f}(k+1)$. It is straightforward to see that when the position $p_i = i$, the corresponding permutation factor $\boldsymbol{P}_i$ degenerates to an identity matrix. If this is true for all M channels, this formulation leads to the equal-order algorithms of [5,6,4,15].

The overall process is illustrated in Fig. 2 for a three-channel case with the first two channels having equal length, i.e., $p_1 = 1$ and $p_2 = 2$; consequently, $\boldsymbol{P}_1 = \boldsymbol{P}_2 = \boldsymbol{I}$. Part one of this figure shows the initial state as in (15) but with reduced dimension, and the operations involving matrices $\boldsymbol{Q}'_{\theta f}(k+1)$ and $\boldsymbol{P}$ are illustrated in parts two and three, respectively. As we can see, the resulting matrix $\bar{\boldsymbol{U}}_{P+M}(k+1)$ in (15) does not have the desired lower triangular shape. Hence, another permutation factor to up-shift the $(P+M-i+1)$th row to the $(P+M-p_i+1)$th position is needed leading to

$$\boldsymbol{U}_{P+M}(k+1) = \overline{\boldsymbol{P}}\boldsymbol{Q}'_{\theta f}(k+1)\begin{bmatrix} \boldsymbol{D}_{fq2}(k+1) & \boldsymbol{U}_P(k) \\ \boldsymbol{E}_f(k+1) & \boldsymbol{0} \end{bmatrix}\boldsymbol{P} \tag{16}$$

where the permutation matrix $\overline{\boldsymbol{P}} = \boldsymbol{P}_1\boldsymbol{P}_2\cdots\boldsymbol{P}_M$, $\boldsymbol{P}_i$ is responsible for up-shifting the $P+M-i+1$ row to the $P+M-p_i+1$ position.

From (16), it is possible to obtain

$$\begin{aligned}[\boldsymbol{U}_{P+M}(k+1)]^{-1} &= \boldsymbol{P}^T \\ &\times \begin{bmatrix} \boldsymbol{0} & \boldsymbol{E}_f^{-1}(k+1) \\ \boldsymbol{U}_P^{-1}(k) & -\boldsymbol{U}_P^{-1}(k)\boldsymbol{D}_{fq2}(k+1)\boldsymbol{E}_f^{-1}(k+1) \end{bmatrix}\boldsymbol{Q}'^{T}_{\theta f}(k+1)\overline{\boldsymbol{P}}^T\end{aligned} \tag{17}$$

which will be used in the next section to derive the *a priori* and the *a posteriori* versions of the algorithm.

Also from (16), we can write

$$\begin{bmatrix} \boldsymbol{0} \\ * \\ \boldsymbol{E}_f^0(k+1) \end{bmatrix} = \boldsymbol{Q}'_{\theta f}(k+1)\begin{bmatrix} \boldsymbol{D}_{fq2}(k+1) \\ \boldsymbol{E}_f(k+1) \end{bmatrix} \tag{18}$$

where $\boldsymbol{E}_f^0(k+1)$ is the zero-order error covariance matrix[3]. The *asterisk* $*$ is used to denote possible non-zero elements according to the process explained above.

[3] The term was coined due to the fact that, in the single channel case, the corresponding scalar $\|\mathbf{e}_f^{(0)}(k+1)\| = \sum_{i=0}^{k+1}\lambda^{(k+1-i)}x^2(i)$ is the norm of the zero-order forward prediction error which is an estimate (albeit biased) of the input variance. Also note that, for zero-order prediction, the forward prediction error vector equals its backward counterpart, and $\|\mathbf{e}_f^{(0)}(k)\| = \|\mathbf{e}_b^{(0)}(k)\|$.

5 *A PRIORI* and *A POSTERIORI* Versions

The *a priori* and the *a posteriori* versions of the block-channel algorithm in Section 4 are based on updating expanded vectors $\boldsymbol{a}_{P+M}(k+1)$ or $\boldsymbol{f}_{P+M}(k+1)$ given by

$$\boldsymbol{a}_{P+M}(k+1) = \lambda^{-1/2}\boldsymbol{U}_{P+M}^{-T}(k)\boldsymbol{x}_{P+M}(k+1), \quad \text{and} \tag{19}$$
$$\boldsymbol{f}_{P+M}(k+1) = \boldsymbol{U}_{P+M}^{-T}(k+1)\boldsymbol{x}_{P+M}(k+1). \tag{20}$$

Vectors $\boldsymbol{a}_P(k+1)$ and $\boldsymbol{f}_P(k+1)$ are contained within the matrix $\mathbf{Q}_\theta(k+1)$ and are also known as the *a priori* and the *a posteriori* backward error vectors, respectively [6,5].

From Equations (6), (17), and (19), we can write

$$\boldsymbol{a}_{P+M}(k+1) = \overline{\boldsymbol{P}}\lambda^{-1/2}\boldsymbol{Q}'_{\theta f}(k)\begin{bmatrix}\boldsymbol{a}_P(k)\\ \boldsymbol{r}(k+1)\end{bmatrix} \tag{21}$$

where

$$\begin{aligned}\boldsymbol{r}(k+1) &= \lambda^{-1/2}\boldsymbol{E}_f^{-T}(k)\left[\boldsymbol{x}_{k+1} - \boldsymbol{W}_f^T(k)\boldsymbol{x}_P(k)\right]\\ &= \lambda^{-1/2}\boldsymbol{E}_f^{-T}(k)\boldsymbol{e}'_f(k+1)\end{aligned} \tag{22}$$

with $\boldsymbol{e}'_f(k+1)$ being the *a priori* forward error vector and

$$\boldsymbol{W}_f(k) = \boldsymbol{U}_P^{-1}(k-1)\boldsymbol{D}_{fq2}(k) \tag{23}$$

is a matrix containing the coefficient vectors of the forward prediction problem.

Likewise, combining Equations (6), (17), and (20), we have

$$\boldsymbol{f}_{P+M}(k+1) = \overline{\boldsymbol{P}}\boldsymbol{Q}'_{\theta f}(k+1)\begin{bmatrix}\boldsymbol{f}_P(k)\\ \boldsymbol{p}(k+1)\end{bmatrix} \tag{24}$$

where

$$\begin{aligned}\boldsymbol{p}(k+1) &= \boldsymbol{E}_f^{-T}(k+1)\left[\boldsymbol{x}_{k+1} - \boldsymbol{W}_f^T(k+1)\boldsymbol{x}_P(k)\right]\\ &= \boldsymbol{E}_f^{-T}(k+1)\boldsymbol{e}_f(k+1)\end{aligned} \tag{25}$$

with $\boldsymbol{e}_f(k+1)$ being the *a posteriori* forward error vector.

The matrix inversion operation in (22) can be avoided using the recursive solution presented in [6] (see also Table 3). To solve for $\boldsymbol{p}(k+1)$ in Equation (25), we can use the the following equation (see Appendix A for the proof).

$$\overline{\boldsymbol{Q}}_f(k+1)\begin{bmatrix}\gamma(k)\\ \mathbf{0}\end{bmatrix}=\begin{bmatrix}*\\ \boldsymbol{p}(k+1)\end{bmatrix} \tag{26}$$

The rotation angles in matrix $\boldsymbol{Q}_\theta(k)$ are obtained using

$$\boldsymbol{Q}_\theta(k+1)\begin{bmatrix}1\\ \mathbf{0}\end{bmatrix}=\begin{bmatrix}\gamma(k+1)\\ \boldsymbol{f}_P(k+1)\end{bmatrix} \tag{27}$$

for the *a posteriori* case, and

$$\begin{bmatrix}1/\gamma(k+1)\\ \mathbf{0}\end{bmatrix}=\boldsymbol{Q}_\theta(k+1)\begin{bmatrix}1\\ -\boldsymbol{a}_P(k+1)\end{bmatrix} \tag{28}$$

for the *a priori* case.

Finally, the joint process estimation is performed as

$$\begin{bmatrix}e_{q1}(k+1)\\ \boldsymbol{d}_{q2}(k+1)\end{bmatrix}=\boldsymbol{Q}_\theta(k+1)\begin{bmatrix}d(k+1)\\ \lambda^{1/2}\boldsymbol{d}_{q2}(k)\end{bmatrix} \tag{29}$$

and the *a priori error* is given by [5,6]

$$\varepsilon(k+1)=e_{q1}(k+1)/\gamma(k+1). \tag{30}$$

The *a posteriori* and *a priori* algorithms are summarized in Tables 2 and 3, respectively.

Remark 1 *In [4], two* a priori *and two* a posteriori *block-channel MC-FQRD-RLS algorithms were presented for the special case of M channels with equal orders, i.e., $N_i = N$ and $P = MN$. For this particular case, the last M elements of vectors $\boldsymbol{a}_{P+M}(k+1)$ and $\boldsymbol{f}_{P+M}(k+1)$ are known prior to their updating through Equations (19) and (20), respectively. However, this prior knowledge is no longer available for the general case of multiple order channels. If the algorithms of Tables 2 and 3 are constrained to have equal channel orders, matrices $\boldsymbol{P}$ and $\overline{\boldsymbol{P}}$ would be identities. Then, the* a priori *and* a posteriori *algorithms presented in this section would reduce to those of [4].*

Table 2
The MCFQRD_POS_B Equations [4,5,16].

```
For each $k$, do
{ 1. Obtaining $\boldsymbol{D}_{fq2}(k+1)$ and $\boldsymbol{e}_{fq1}(k+1)$
    $\begin{bmatrix} \boldsymbol{e}^T_{fq1}(k+1) \\ \boldsymbol{D}_{fq2}(k+1) \end{bmatrix} = \boldsymbol{Q}_\theta(k) \begin{bmatrix} \boldsymbol{x}^T_{k+1} \\ \lambda^{1/2}\boldsymbol{D}_{fq2}(k) \end{bmatrix}$    (13)
  2. Obtaining $\boldsymbol{E}_f(k+1)$ and $\overline{\boldsymbol{Q}}_f(k+1)$
    $\begin{bmatrix} \boldsymbol{0}^T \\ \boldsymbol{E}_f(k+1) \end{bmatrix} = \overline{\boldsymbol{Q}}_f(k+1) \begin{bmatrix} \boldsymbol{e}^T_{fq1}(k+1) \\ \lambda^{1/2}\boldsymbol{E}_f(k) \end{bmatrix}$    (14)
  3. Obtaining $\boldsymbol{p}(k+1)$
    $\begin{bmatrix} * \\ \boldsymbol{p}(k+1) \end{bmatrix} = \overline{\boldsymbol{Q}}_f(k+1) \begin{bmatrix} \gamma(k) \\ \boldsymbol{0} \end{bmatrix}$    implements (25)
  4. Obtaining $\boldsymbol{Q}'_{\theta f}(k+1)$
    $\begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{*} \\ \boldsymbol{E}^0_f(k+1) \end{bmatrix} = \boldsymbol{Q}'_{\theta f}(k+1) \begin{bmatrix} \boldsymbol{D}_{fq2}(k+1) \\ \boldsymbol{E}_f(k+1) \end{bmatrix}$    (18)
  5. Obtaining $\boldsymbol{f}_P(k+1)$
    $\boldsymbol{f}_{P+M}(k+1) = \overline{\boldsymbol{P}}\boldsymbol{Q}'_{\theta f}(k+1) \begin{bmatrix} \boldsymbol{f}_P(k) \\ \boldsymbol{p}(k+1) \end{bmatrix}$    (24)
  6. Obtaining $\boldsymbol{Q}_\theta(k+1)$ and $\gamma(k+1)$
    $\boldsymbol{Q}_\theta(k+1) \begin{bmatrix} 1 \\ \boldsymbol{0} \end{bmatrix} = \begin{bmatrix} \gamma(k+1) \\ \boldsymbol{f}_P(k+1) \end{bmatrix}$    (27)
  7. Joint Estimation
    $\begin{bmatrix} e_{q1}(k+1) \\ \boldsymbol{d}_{q2}(k+1) \end{bmatrix} = \boldsymbol{Q}_\theta(k+1) \begin{bmatrix} d(k+1) \\ \lambda^{1/2}\boldsymbol{d}_{q2}(k) \end{bmatrix}$    (29)
  8. Obtaining the a priori error
    $\varepsilon(k+1) = e_{q1}(k+1)/\gamma(k+1)$    (30)
}
```

6 Order Recursive Block-type Multichannel Fast QRD-RLS Algorithm

In this section, we derive an order recursive version of the *a posteriori* algorithm of Table 2 for the special case when all M channels have equal orders,

Table 3
The MCFQRD_PRI_B Equations [4,6,16].

```
For each $k$, do
{ 1. Obtaining $\boldsymbol{D}_{fq2}(k+1)$ and $\boldsymbol{e}_{fq1}(k+1)$
    $\begin{bmatrix} \boldsymbol{e}_{fq1}^T(k+1) \\ \boldsymbol{D}_{fq2}(k+1) \end{bmatrix} = \boldsymbol{Q}_\theta(k) \begin{bmatrix} \boldsymbol{x}_{k+1}^T \\ \lambda^{1/2}\boldsymbol{D}_{fq2}(k) \end{bmatrix}$    (13)
  2. Obtaining $\boldsymbol{E}_f(k+1)$ and $\overline{\boldsymbol{Q}}_f(k+1)$
    $\begin{bmatrix} \boldsymbol{0}^T \\ \boldsymbol{E}_f(k+1) \end{bmatrix} = \overline{\boldsymbol{Q}}_f(k+1) \begin{bmatrix} \boldsymbol{e}_{fq1}^T(k+1) \\ \lambda^{1/2}\boldsymbol{E}_f(k) \end{bmatrix}$    (14)
  3. Obtaining $\boldsymbol{r}(k+1)$
    $\begin{bmatrix} * \\ \boldsymbol{0} \end{bmatrix} = \overline{\boldsymbol{Q}}_f(k+1) \begin{bmatrix} 1/\gamma(k) \\ -\boldsymbol{r}(k+1) \end{bmatrix}$    implements (22)
  4. Obtaining $\boldsymbol{a}_P(k+1)$
    $\boldsymbol{a}_{P+M}(k+1) = \overline{\boldsymbol{P}}\boldsymbol{Q}'_{\theta f}(k+1) \begin{bmatrix} \boldsymbol{a}_P(k) \\ \boldsymbol{r}(k+1) \end{bmatrix}$    (21)
  5. Obtaining $\boldsymbol{Q}'_{\theta f}(k+1)$
    $\begin{bmatrix} \boldsymbol{0} \\ * \\ \boldsymbol{E}_f^0(k+1) \end{bmatrix} = \boldsymbol{Q}'_{\theta f}(k+1) \begin{bmatrix} \boldsymbol{D}_{fq2}(k+1) \\ \boldsymbol{E}_f(k+1) \end{bmatrix}$    (18)
  6. Obtaining $\boldsymbol{Q}_\theta(k+1)$ and $\gamma(k+1)$
    $\begin{bmatrix} 1/\gamma(k+1) \\ \boldsymbol{0} \end{bmatrix} = \boldsymbol{Q}_\theta(k+1) \begin{bmatrix} 1 \\ -\boldsymbol{a}_P(k+1) \end{bmatrix}$    (28)
  7. Joint Estimation
    $\begin{bmatrix} e_{q1}(k+1) \\ \boldsymbol{d}_{q2}(k+1) \end{bmatrix} = \boldsymbol{Q}_\theta(k+1) \begin{bmatrix} d(k+1) \\ \lambda^{1/2}\boldsymbol{d}_{q2}(k) \end{bmatrix}$    (29)
  8. Obtaining the a priori error
    $\varepsilon(k+1) = e_{q1}(k+1)/\gamma(k+1)$    (30)
}
```

i.e., $N_i = N$. The *a priori* counterpart of the algorithm presented here can be found in [6].

For the order recursive version, the quantities $\boldsymbol{D}_{fq2}(k)$, $\boldsymbol{d}_{q2}(k)$, and $\boldsymbol{f}_P(k)$ are split up into N blocks. For matrix $\boldsymbol{D}_{fq2}(k)$, we have

$$\boldsymbol{D}_{fq2}(k) = \begin{bmatrix} \boldsymbol{D}^{(1)}_{fq2}(k) \\ \vdots \\ \boldsymbol{D}^{(N)}_{fq2}(k) \end{bmatrix} \tag{31}$$

where $\boldsymbol{D}^{(i)}_{fq2}(k)$ has dimensions $M \times M$.

Taking into consideration the block-channel structure of $\boldsymbol{D}^{(i)}_{fq2}(k)$ and that $N_i = N$ (equal orders), Equation (18) can be rewritten as

$$\begin{bmatrix} \mathbf{0}_{M(N-i-1)\times M} \\ \mathbf{0}_{M(i-1)\times M} \\ \boldsymbol{E}^{(i-1)}_{f}(k+1) \end{bmatrix} = \boldsymbol{Q}'_{\theta f}{}^{(N-i+1)}(k+1) \begin{bmatrix} \mathbf{0}_{M(N-i)\times M} \\ \boldsymbol{D}^{(N-i+1)}_{fq2}(k) \\ \mathbf{0}_{M(i-1)\times M} \\ \boldsymbol{E}^{(i)}_{f}(k+1) \end{bmatrix} \tag{32}$$

for $i = N, N-1, \cdots, 1$.

With the order-recursive formulation in Equation (32), matrix $\boldsymbol{Q}'_{\theta f}(k+1)$ in Equation (18) will be equal to

$$\boldsymbol{Q}'_{\theta f}(k+1) = \boldsymbol{Q}'_{\theta f}{}^{(N)}(k+1)\boldsymbol{Q}'_{\theta f}{}^{(N-1)}(k+1) \;\cdots\; \boldsymbol{Q}'_{\theta f}{}^{(1)}(k+1). \tag{33}$$

Equation (32) can also be performed in a forward manner, i.e., for $i = 1, 2, \cdots, N$. This property is the key to derive the lattice version of the algorithm. Recalling that $\boldsymbol{Q}'_{\theta f}(k)$ is used to update $\boldsymbol{f}_P(k)$, we can rewrite Equation (24) as

$$\begin{bmatrix} \mathbf{0}_{M(N-i)} \\ \boldsymbol{f}^{(N-i+1)}(k+1) \\ \mathbf{0}_{M(i-1)} \\ \boldsymbol{p}_{i-1}(k+1) \end{bmatrix} = \boldsymbol{Q}'_{\theta f}{}^{(N-i+1)}(k+1) \begin{bmatrix} \mathbf{0}_{M(N-i)} \\ \boldsymbol{f}^{(N-i+2)}(k) \\ \mathbf{0}_{M(i-1)} \\ \boldsymbol{p}_{i}(k+1) \end{bmatrix} \tag{34}$$

for $i = 1, 2, \cdots, N$.

Using the last two equations, steps 4 and 5 of the algorithm in Table 2 can now be carried out in a forward manner. The rotation angles $\boldsymbol{Q}^{(i)}_{\theta}(k+1)$ are obtained through

$$\boldsymbol{Q}^{(i)}_{\theta}(k+1) \begin{bmatrix} \gamma_{i-1}(k+1) \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \gamma_{i}(k+1) \\ \boldsymbol{f}^{(N-i+2)}(k+1) \end{bmatrix} \tag{35}$$

and the joint estimation is performed according to

$$\begin{bmatrix} \boldsymbol{e}_{q1}^{(i)}(k+1) \\ \boldsymbol{d}_{q2}^{(N-i+1)}(k+1) \end{bmatrix} = \boldsymbol{Q}_{\theta}^{(N-i+1)}(k+1) \begin{bmatrix} \boldsymbol{e}_{q1}^{(i)}(k+1) \\ \lambda^{1/2}\boldsymbol{d}_{q2}^{(N-i+1)}(k) \end{bmatrix}. \tag{36}$$

In order to adjust the equations of steps 1 to 3 of the algorithm in Table 2 to this formulation, it suffices to observe that they can be split up into $M \times M$ blocks that will be processed in an order-recursive way. Table 4 presents details of the lattice (or order recursive) version of the Block-type Multichannel Fast QRD-RLS algorithm based on *a posteriori* backward prediction errors [15].

Table 4
The Lattice Block-channel MCFQRD_POS_B Algorithm [15].

```
*Initializations:*
$\boldsymbol{f}_P(0) = 0$; $\boldsymbol{D}_{fq2}(0) = 0$; $\gamma_0(0) = 1$; $\boldsymbol{d}_{q2}(0) = 0$; $\boldsymbol{E}_f^i(0) = \mu \boldsymbol{I}$,
$\mu = small\ \ number$, all $cosines = 1$, and all $sines = 0$;
For each $k$, do
{ $\widetilde{\boldsymbol{e}}_{fq1}^{(0)\ T}(k+1) = \boldsymbol{x}_{k+1}^T$
  A. Obtaining $\boldsymbol{E}_f^{(0)}(k+1)$ and $\overline{\boldsymbol{Q}}_f^{(0)}(k+1)$
  $\begin{bmatrix} \mathbf{0}^T \\ \boldsymbol{E}_f^{(0)}(k+1) \end{bmatrix} = \overline{\boldsymbol{Q}}_f^{(0)}(k+1) \begin{bmatrix} \widetilde{\boldsymbol{e}}_{fq1}^{(0)\ T}(k+1) \\ \lambda^{1/2}\boldsymbol{E}_f^{(0)}(k) \end{bmatrix}$
  B. Obtaining $\boldsymbol{p}_0(k+1)$
  $\begin{bmatrix} * \\ \boldsymbol{p}_0(k+1) \end{bmatrix} = \overline{\boldsymbol{Q}}_f^{(0)}(k+1) \begin{bmatrix} \gamma_0(k) \\ \mathbf{0} \end{bmatrix}$
  $\boldsymbol{f}^{(N+1)}(k+1) = \boldsymbol{p}_0(k+1)$;  $\gamma_0(k+1) = 1$;
  $\boldsymbol{e}_{q1}(k+1) = d(k+1)$
  for $i = 1 : N$
   { 1. Obtaining $\boldsymbol{D}_{fq2}^{(N-i+1)}(k+1)$ and $\boldsymbol{e}_{fq1}^{(i)}(k+1)$
   $\begin{bmatrix} \widetilde{\boldsymbol{e}}_{fq1}^{(i)\ T}(k+1) \\ \boldsymbol{D}_{fq2}^{(N-i+1)}(k+1) \end{bmatrix} = \boldsymbol{Q}_\theta^{(i)}(k) \begin{bmatrix} \widetilde{\boldsymbol{e}}_{fq1}^{(i-1)\ T}(k+1) \\ \lambda^{1/2}\boldsymbol{D}_{fq2}^{(N-i+1)}(k) \end{bmatrix}$
   2. Obtaining $\boldsymbol{E}_f^{(i)}(k+1)$
   $\begin{bmatrix} \mathbf{0}^T \\ \boldsymbol{E}_f^{(i)}(k+1) \end{bmatrix} = \overline{\boldsymbol{Q}}_f^{(i)}(k+1) \begin{bmatrix} \widetilde{\boldsymbol{e}}_{fq1}^{(i)\ T}(k+1) \\ \lambda^{1/2}\boldsymbol{E}_f^{(i)}(k) \end{bmatrix}$
   3. Obtaining $\boldsymbol{p}_i(k+1)$
   $\begin{bmatrix} * \\ \boldsymbol{p}_i(k+1) \end{bmatrix} = \overline{\boldsymbol{Q}}_f^{(i)}(k+1) \begin{bmatrix} \gamma_i(k) \\ \mathbf{0} \end{bmatrix}$
   4. Obtaining $\boldsymbol{Q}'_{\theta f}{}^{(N-i+1)}(k+1)$
   $\begin{bmatrix} \mathbf{0}_{M(N-i-1)\times M} \\ \mathbf{0}_{M(i-1)\times M} \\ \boldsymbol{E}_f^{(i-1)}(k+1) \end{bmatrix} = \boldsymbol{Q}'_{\theta f}{}^{(N-i+1)}(k+1) \begin{bmatrix} \mathbf{0}_{M(N-i)\times M} \\ \boldsymbol{D}_{fq2}^{(N-i+1)}(k) \\ \mathbf{0}_{M(i-1)\times M} \\ \boldsymbol{E}_f^{(i)}(k+1) \end{bmatrix}$
   5. Obtaining $\boldsymbol{f}^{(N-i+1)}(k+1)$
   $\begin{bmatrix} \mathbf{0}_{M(N-i)} \\ \boldsymbol{f}^{(N-i+1)}(k+1) \\ \mathbf{0}_{M(i-1)} \\ \boldsymbol{p}_{i-1}(k+1) \end{bmatrix} = \boldsymbol{Q}'_{\theta f}{}^{(N-i+1)}(k+1) \begin{bmatrix} \mathbf{0}_{M(N-i)} \\ \boldsymbol{f}^{(N-i+2)}(k) \\ \mathbf{0}_{M(i-1)} \\ \boldsymbol{p}_i(k+1) \end{bmatrix}$
   6. Obtaining $\boldsymbol{Q}_\theta^{(i)}(k+1)$ and $\gamma_i(k+1)$
   $\boldsymbol{Q}_\theta^{(i)}(k+1) \begin{bmatrix} \gamma_{i-1}(k+1) \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \gamma_i(k+1) \\ \boldsymbol{f}^{(N-i+2)}(k+1) \end{bmatrix}$
   7. Joint Estimation
   $\begin{bmatrix} \boldsymbol{e}_{q1}^{(i)}(k+1) \\ \boldsymbol{d}_{q2}^{(N-i+1)}(k+1) \end{bmatrix} = \boldsymbol{Q}_\theta^{(i)}(k+1) \begin{bmatrix} \boldsymbol{e}_{q1}^{(i-1)}(k+1) \\ \lambda^{1/2}\boldsymbol{d}_{q2}^{(N-i+1)}(k) \end{bmatrix}$
   }
  8. Obtaining the *a priori* error
  $\varepsilon(k+1) = e_{q1}(k+1)/\gamma(k+1)$
}
```

7 Sequential-type Multichannel Fast QRD-RLS Algorithms

So far we have been concerned with the derivation of block-type MC-FQRD-RLS algorithms. This section considers algorithms that process the channels sequentially. In the following, we derive the *a posteriori* counterpart of the transversal and order recursive algorithms presented in [6].

The input data matrices used in sequential-channel algorithms are defined as

$$\boldsymbol{X}_{P+i}(k) = \begin{bmatrix} \boldsymbol{x}^T_{P+i}(k) \\ \lambda^{1/2}\boldsymbol{x}^T_{P+i}(k-1) \\ \vdots \\ \lambda^{k/2}\boldsymbol{x}^T_{P+i}(0) \end{bmatrix}, \quad i = 1, 2, \cdots, M. \tag{37}$$

where vector $\boldsymbol{x}_{P+i}(k)$ is the extended input vector defined in Equation (10).

If matrix $\boldsymbol{U}_{P+i}(k)$ is used to denote the Cholesky factor of $\boldsymbol{X}^T_{P+i}(k)\boldsymbol{X}_{P+i}(k)$, we can, in a similar manner as for the block-channel algorithms, define the *a posteriori* backward error vector, $\boldsymbol{f}_{P+i}(k+1)$, as

$$\boldsymbol{f}_{P+i}(k+1) = \boldsymbol{U}^{-T}_{P+i}(k+1)\boldsymbol{x}_{P+i}(k+1), \;\; for \;\; i = 1, 2, \cdots, M. \tag{38}$$

From (38) and the definition of the input vector $\boldsymbol{x}_{P+i}$ in Subsection 3.2, we can write

$$\boldsymbol{f}_{P+M}(k+1) = \begin{bmatrix} \boldsymbol{f}^{(M)}(k+1) \\ \boldsymbol{f}_P(k+1) \end{bmatrix} \tag{39}$$

where $\boldsymbol{f}^{(M)}(k+1)$ is a vector containing the first M elements of $\boldsymbol{f}_{P+M}(k+1)$.

The updating of $\boldsymbol{f}_{P+i}(k+1)$ is accomplished in M forward steps at each instant k:

$$\boldsymbol{f}_P(k) \rightarrow \boldsymbol{f}_{P+1}(k+1) \rightarrow \;\; \cdots \;\; \rightarrow \boldsymbol{f}_{P+M}(k+1)$$

Equation (37) suggests that the updating of the information matrix is performed in M forward steps for each iteration.

7.1.1 First step $(i = 1)$

$\boldsymbol{X}_{P+1}(k)$ can be defined as

$$\boldsymbol{X}_{P+1}(k) = \left[\begin{array}{c|c} \boldsymbol{d}_f^{(1)}(k) & \begin{array}{c} \boldsymbol{X}_P(k-1) \\ \boldsymbol{0}^T \end{array} \end{array} \right] \tag{40}$$

where $\boldsymbol{d}_{f1}^{(1)}(k) = [x_1(k) \quad \lambda^{1/2} x_1(k-1) \quad \cdots \quad \lambda^{k/2} x_1(0)]$.

Let $\boldsymbol{Q}_P^{(1)}(k)$ be the orthogonal matrix associated with the Cholesky factor $\boldsymbol{U}_P(k-1)$ of matrix $\boldsymbol{X}_P^T(k-1)\boldsymbol{X}_P(k-1)$. Then, from (40), we can write

$$\begin{bmatrix} \boldsymbol{Q}_P^{(1)}(k) & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I}_{1\times 1} \end{bmatrix} \left[\begin{array}{c|c} \boldsymbol{d}_f^{(1)}(k) & \begin{array}{c} \boldsymbol{X}_P(k-1) \\ \boldsymbol{0}^T \end{array} \end{array} \right] = \begin{bmatrix} \boldsymbol{e}_{fq1}^{(1)}(k) & \boldsymbol{0} \\ \boldsymbol{d}_{fq2}^{(1)}(k) & \boldsymbol{U}_P(k-1) \\ \lambda^{k/2} x_1(0) & \boldsymbol{0}^T \end{bmatrix} \tag{41}$$

To complete the triangularization process of $\boldsymbol{X}_{P+1}(k)$ leading to $\boldsymbol{U}_{P+1}(k)$, we premultiply (41) by two other Givens rotation matrices as follows

$$\begin{aligned} \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{U}_{P+1}(k) \end{bmatrix} &= \boldsymbol{Q}'^{(1)}_f(k) \boldsymbol{Q}_f^{(1)}(k) \begin{bmatrix} \boldsymbol{e}_{fq1}^{(1)}(k) & \boldsymbol{0} \\ \boldsymbol{d}_{fq2}^{(1)}(k) & \boldsymbol{U}_P(k-1) \\ \lambda^{k/2} x_1(0) & \boldsymbol{0}^T \end{bmatrix} \\ &= \boldsymbol{Q}'^{(1)}_f(k) \begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{d}_{fq2}^{(1)}(k) & \boldsymbol{U}_P(k-1) \\ e_{fP}^{(1)}(k) & \boldsymbol{0}^T \end{bmatrix} \end{aligned} \tag{42}$$

In the previous equation, $\boldsymbol{Q}_f^{(1)}(k)$ is the orthogonal matrix zeroing $\boldsymbol{e}_{fq1}^{(1)}(k)$ generating $e_{fP}^{(1)}(k)$. Matrix $\boldsymbol{Q}'^{(1)}_f(k)$ completes the triangularization process by zeroing $\boldsymbol{d}_{fq2}^{(1)}(k)$ from (42) in a top down procedure against $e_{fP}^{(1)}(k)$. Removing the resulting null section in the upper part of (42) gives

$$\boldsymbol{U}_{P+1}(k) = \boldsymbol{Q}'_{\theta f}{}^{(1)}(k) \begin{bmatrix} \boldsymbol{d}_{fq2}^{(1)}(k) & \boldsymbol{U}_P(k-1) \\ e_{fP}^{(1)}(k) & \mathbf{0}^T \end{bmatrix}. \tag{43}$$

From (43), we get the following relation that is useful for the updating of $\boldsymbol{f}_P(k)$:

$$[\boldsymbol{U}_{P+1}(k+1)]^{-1} = \begin{bmatrix} \mathbf{0}^T & \frac{1}{e_{fP}^{(1)}(k+1)} \\ \boldsymbol{U}_P^{-1}(k) & -\frac{1}{e_{fP}^{(1)}(k+1)}\boldsymbol{U}_P^{-1}(k)\boldsymbol{d}_{fq2}^{(1)}(k+1) \end{bmatrix} \left[\boldsymbol{Q}'_{\theta f}{}^{(1)}(k+1)\right]^T \tag{44}$$

Also from (43), we see that $\boldsymbol{Q}'_{\theta f}{}^{(1)}(k)$ is the Givens rotation matrix responsible for zeroing $\boldsymbol{d}_{fq2}^{(1)}(k)$ against $e_{fP}^{(1)}(k)$, i.e., we have

$$\begin{bmatrix} \mathbf{0} \\ e_{f0}^{(1)}(k+1) \end{bmatrix} = \boldsymbol{Q}'_{\theta f}{}^{(1)}(k+1) \begin{bmatrix} \boldsymbol{d}_{fq2}^{(1)}(k+1) \\ e_{fP}^{(1)}(k+1) \end{bmatrix}. \tag{45}$$

A recursive expression for $\boldsymbol{f}_{P+1}(k+1)$ is obtained by using (44) and (9) in (20):

$$\boldsymbol{f}_{P+1}(k+1) = \boldsymbol{Q}'_{\theta f}{}^{(1)}(k+1) \begin{bmatrix} \boldsymbol{f}_P(k) \\ p^{(1)}(k+1) \end{bmatrix} \tag{46}$$

where

$$p^{(1)}(k+1) = \frac{e_P{}^{(1)}(k+1)}{|e_{fP}^{(1)}(k+1)|} \tag{47}$$

with $e_P{}^{(1)}(k+1)$ denoting the *a posteriori* error of the forward prediction of the first channel, and $|e_{f_P}^{(1)}(k+1)|$ is given by

$$|e_{f_P}^{(1)}(k+1)| = \sqrt{\left(\lambda^{1/2}|e_{f_P}^{(1)}(k)|\right)^2 + |(e_{fq1_P}^{(i)}(k+1)|^2}. \tag{48}$$

The updating of $\boldsymbol{d}_{fq2}^{(1)}(k)$ is performed according to

$$\begin{bmatrix} \tilde{e}_{fq1}^{(1)}(k+1) \\ \boldsymbol{d}_{fq2}^{(1)}(k+1) \end{bmatrix} = \boldsymbol{Q}_{\theta_P}^{(0)}(k) \begin{bmatrix} x_1(k+1) \\ \lambda^{1/2}\boldsymbol{d}_{fq2}^{(1)}(k) \end{bmatrix}. \tag{49}$$

Matrix $\boldsymbol{Q}_{\theta_{P+1}}^{(1)}(k+1)$, needed in the next steps, is obtained from

$$\boldsymbol{Q}_{\theta_{P+1}}^{(1)}(k+1) \begin{bmatrix} 1 \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \gamma_{P+1}^{(i)}(k+1) \\ \boldsymbol{f}_{P+1}(k+1) \end{bmatrix}. \tag{50}$$

7.1.2 Following steps ($i > 1$)

The input information matrix $\boldsymbol{X}_{P+i}(k)$ is related to $\boldsymbol{X}_{P+i-1}(k)$ according to

$$\boldsymbol{X}_{P+i}(k) = \begin{bmatrix} x_i(k) & \\ \lambda^{1/2}x_i(k-1) & \boldsymbol{X}_{P+i-1}(k) \\ \vdots & \\ \lambda^{k/2}x_i(0) & \end{bmatrix} \boldsymbol{P}_i. \tag{51}$$

As in the first step, matrix $\boldsymbol{X}_{P+i}(k)$ must be triangularized to obtain $\boldsymbol{U}_{P+i}(k)$ (Cholesky factor of $\boldsymbol{X}_{P+i}^T(k)\boldsymbol{X}_{P+i}(k)$). This process is detailed in the following. Let $\boldsymbol{Q}_{\theta_{P+i-1}}(k)$ denote the orthogonal matrix associated with the QR decomposition of $\boldsymbol{X}_{P+i-1}(k)$. From (51), we can write

$$\begin{aligned} &\mathbf{Q}_f^{(i)}(k) \begin{bmatrix} \boldsymbol{Q}_{P+i-1}(k) & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{X}_{P+i}(k) \\ \mathbf{0}^T \end{bmatrix} = \\ &\mathbf{Q}_f^{(i)}(k) \begin{bmatrix} \boldsymbol{e}_{fq1_{P+i-1}}^{(i)}(k) & \mathbf{0} \\ \boldsymbol{d}_{fq2}^{(i)}(k) & \boldsymbol{U}_{P+i-1}(k) \\ \lambda^{k/2}\boldsymbol{x}_i(0) & \mathbf{0}^T \end{bmatrix} \boldsymbol{P}_i = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \boldsymbol{d}_{fq2}^{(i)}(k) & \boldsymbol{U}_{P+i-1}(k) \\ e_{f_{P+i-1}}^{(i)}(k) & \mathbf{0}^T \end{bmatrix} \boldsymbol{P}_i. \end{aligned} \tag{52}$$

Equation (52) is obtained by annihilating $\boldsymbol{e}_{fq1_{P+i-1}}^{(i)}(k)$ into the first element of the last row of the matrix using an appropriate orthogonal matrix, $\mathbf{Q}_f^{(i)}(k)$, and thereafter removing the resulting null section.

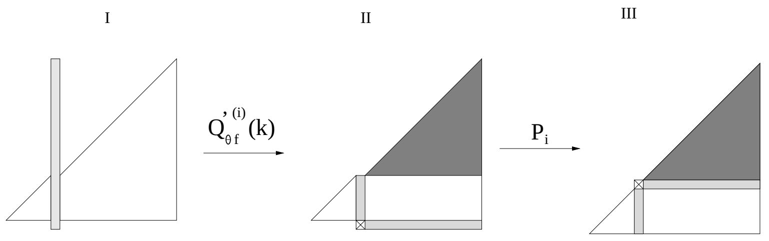

Fig. 3. Obtaining the lower triangular factor $\boldsymbol{U}_{P+i}(k)$.

As before, the existence of the permutation matrix $\boldsymbol{P}_i$ in (52) prevents us from directly annihilating $\boldsymbol{d}^{(i)}_{fq2}(k)$ into $e^{(i)}_{f_{P+i-1}}(k)$ to complete the triangularization of matrix $\boldsymbol{X}_{P+i}(k)$ (i.e., generating $\boldsymbol{U}_{P+i}(k)$). Fig. 3 illustrates the application of the Givens rotations under these circumstances. This process can be summarized as follows. The permutation factor, $\boldsymbol{P}_i$, right shifts $\boldsymbol{d}^{(i)}_{fq2}(k)$ to the ith position as shown in the first part of the figure. Then, a set of $P+i-p_i$ Given rotation matrices, ${\boldsymbol{Q}'_{\theta f}}^{(i)}$, are used to nullify the first $P+i-p_i$ elements of $\boldsymbol{d}^{(i)}_{fq2}(k)$ against $e^{(i)}_{f_{P+i-1}}(k)$ in a top down procedure. To obtain the desired triangular structure, we need another permutation factor that moves the last row of the matrix to the $P-p_i+1$ position, after downshifting the previous $P-p_i$ rows. This permutation factor coincides with $\boldsymbol{P}_i$.

The lower triangular matrix $\boldsymbol{U}_{P+i}(k)$, obtained as described above, is guaranteed to be positive definite if its diagonal elements and $e^{(i)}_{f_{P+i-1}}(k)$ are positive. Recalling that $e^{(i)}_{f_{P+i-1}}(k)$ is the absolute value of the forward error, $\boldsymbol{U}_{P+i}(k)$ will be positive definite if it is initialized properly.

The procedure above can be written in a more compact form as

$$\boldsymbol{U}_{P+i}(k) = \boldsymbol{P}_i {\boldsymbol{Q}'_{\theta f}}^{(i)}(k) \begin{bmatrix} \boldsymbol{d}^{(i)}_{fq2}(k) & \boldsymbol{U}_{P+i-1}(k) \\ e^{(i)}_{f_{P+i-1}}(k) & \boldsymbol{0}^T \end{bmatrix} \boldsymbol{P}_i. \tag{53}$$

From (53), the following relation can be derived

$$[\boldsymbol{U}_{P+i}(k+1)]^{-1} = \boldsymbol{P}_i^T \times \begin{bmatrix} \boldsymbol{0}^T & \frac{1}{e^{(i)}_{f_{P+i-1}}(k+1)} \\ \boldsymbol{U}^{-1}_{P+i-1}(k+1) & -\frac{\boldsymbol{U}^{-1}_{P+i-1}(k+1)\boldsymbol{d}^{(i)}_{fq2}(k+1)}{e^{(i)}_{f_{P+i-1}}(k+1)} \end{bmatrix} \times {\boldsymbol{Q}'^T_{\theta f}}^{(i)}(k+1)\boldsymbol{P}_i^T \tag{54}$$

From (54), (10), and (38), we get the following recursive expression for $\boldsymbol{f}_{P+i}(k+1)$:

$$\boldsymbol{f}_{P+i}(k+1) = \boldsymbol{P}_i \boldsymbol{Q'}_{\theta f}^{(i)}(k+1) \begin{bmatrix} \boldsymbol{f}_{P+i-1}(k+1) \\ p_{P+i-1}^{(i)}(k+1) \end{bmatrix} \tag{55}$$

where

$$p_{P+i-1}^{(i)}(k+1) = \frac{e_{P+i-1}^{(i)}(k+1)}{|e_{f_{P+i-1}}^{(i)}(k+1)|}. \tag{56}$$

The scalar quantity $e_{P+i-1}^{(i)}(k+1)$ is the *a posteriori* forward prediction error for the *i*th channel, and $|e_{f_{P+i-1}}^{(i)}(k+1)|$ is given by

$$|e_{f_{P+i-1}}^{(i)}(k+1)| = \sqrt{\left(\lambda^{1/2}|e_{f_{P+i-1}}^{(i)}(k)|\right)^2 + |e_{fq1_{P+i-1}}^{(i)}(k+1)|^2} \tag{57}$$

By carefully examining (55) and recalling the definitions of $\boldsymbol{Q'}_{\theta f}^{(i)}(k+1)$ and $\boldsymbol{P}_i$, we can can conclude that the last $p_i - 1$ elements of $\boldsymbol{f}_{P+i}(k+1)$ and $\boldsymbol{f}_{P+i-1}(k+1)$ are identical. To see this, one just needs to remember that the set of Givens rotations in $\boldsymbol{Q'}_{\theta f}^{(i)}(k+1)$ are such that the next rotation matrix always acts in a smaller portion of vector $\boldsymbol{f}_{P+i}(k+1)$ than the previous; then $\boldsymbol{P}_i$ shifts down the unchanged elements which will remain unchanged (after the next channel is processed) and so on. This fact helps reducing the computational burden on the updating process of this vector.

The updating of $\boldsymbol{d}_{fq2}^{(i)}(k)$ is performed according to

$$\begin{bmatrix} \tilde{e}_{fq1}^{(i)}(k+1) \\ \boldsymbol{d}_{fq2}^{(i)}(k+1) \end{bmatrix} = \boldsymbol{Q}_{\theta_{P+i-1}}^{(i-1)}(k+1) \begin{bmatrix} x_i(k+1) \\ \lambda^{1/2}\boldsymbol{d}_{fq2}^{(i)}(k) \end{bmatrix} \tag{58}$$

and the Givens rotations matrices $\boldsymbol{Q}_{\theta_{P+i}}(k+1)$ needed in the next forward step are obtained as follows.

$$\boldsymbol{Q}_{\theta_{P+i}}^{(i)}(k+1) \begin{bmatrix} 1 \\ \boldsymbol{0} \end{bmatrix} = \begin{bmatrix} \gamma_{P+i}^{(i)}(k+1) \\ \boldsymbol{f}_{P+i}(k+1) \end{bmatrix} \tag{59}$$

After the *M-th* channel is processed, the joint process estimation is performed according to

$$\begin{bmatrix} e_{q1}(k+1) \\ \boldsymbol{d}_{q2}(k+1) \end{bmatrix} = \boldsymbol{Q}_{\theta}^{(0)}(k+1) \begin{bmatrix} e_{q1}(k) \\ \boldsymbol{d}_{q2}(k) \end{bmatrix}. \tag{60}$$

Finally, emerging as a direct consequence of what was discussed so far, variables $|e_{f_j}^{(i)}(k+1)|$ and $p_j^{(i)}(k+1)$, in order to attain an order recursive structure for the proposed algorithm, can be expressed as follows.

$$|e_{f_j}^{(i)}(k+1)| = \sqrt{\left(\lambda^{1/2}|e_{f_j}^{(i)}(k)|\right)^2 + \left(e_{fq1_j}^{(i)}(k+1)\right)^2}, \quad \begin{cases} i = 1, 2, \cdots, M \\ j = p_i, \cdots, P \end{cases} \tag{61}$$

and

$$p_j^{(i)}(k+1) = \frac{\gamma_j^{(i-1)}(k)e_{fq1_j}^{(i)}(k+1)}{|e_{f_j}^{(i)}(k+1)|}, \quad \begin{cases} j = p_i, \cdots, P \\ i = 1, 2, \cdots, M \end{cases} \tag{62}$$

The transversal and order recursive algorithms are summarized in Appendix B, Tables 7 and 8, where complex implementations were considered.

8 Simulation results and computational complexity

In this section, the performances of the proposed algorithm are tested in a Volterra system identification setup. Thereafter, a discusssion of the computational complexity is provided together with a classification of all known algorithms based on the updating of backward prediction errors.

8.1 The computer experiment

The performances of the MC-FQRD-RLS algorithms are evaluated in a nonlinear system identification setup. This environment was chosen due to the very high correlation among the element of the input signal vector of the adaptive filter. The plant is a truncated second-order Volterra system [2] which can be described as

$$d(k) = \sum_{n_1=0}^{L-1} w_{n_1}x(k-n_1) + \sum_{n_1=0}^{L-1}\sum_{n_2=0}^{L-1} w_{n_1,n_2}x(k-n_1)x(k-n_2) + \rho(k) \tag{63}$$

Equation (63) can be easily reformulated as a multichannel problem with

$M = L + 1$ channels, where the most recent sample of the ith channel is

$$x_i(k) = \begin{cases} x(k), & i = 1 \\ x(k)x(k-i+2), & i = 2, \cdots, L+1 \end{cases}$$

and the i-th channel order is

$$N_i = \begin{cases} L, & i = 1, 2 \\ L - i + 2, & i = 3, \cdots, L+1. \end{cases}$$

In our experiment, we used $L = 4$, and the power of the observation noise $\rho(k)$ was set such that the signal-to-noise ratio (SNR) was 60 dB. The input signal $x(k)$ applied to the Volterra structure is a white Gaussian noise sequence colored by an AR filter with a single pole at 0.9. The conditioning number (eigenvalue spread) associated with the input vector presented to the multichannel adaptive filter is above 3000 (highly correlated). The value of the forgetting factor was $\lambda = 0.98$. Fig. 4 shows the learning curves (MSE estimated as an average of 5000 independent runs) of the multiple order MC-FQRD-RLS algorithms. The learning curve of the Normalized LMS (NLMS) algorithm [13], with step-size $\mu = 0.5$, was included for comparison. Although not appearing in the figure, the convergence of the NLMS algorithms is reached around $k = 4500$ (around $k = 2500$ in case $\mu = 1$, with a higher misadjustment). All QRD-RLS algorithms (including the multichannel cases), provided they have equivalent initialization, present identical learning curves when compared to the RLS algorithm; this is so because they minimize exactly the same cost function.

Also an average of 100 independent runs of 2×10^4 samples each was carried out. Along with the multiple order MCFQRD-RLS algorithms described in this paper, the RLS [13] and the Inverse QRD-RLS [17,18] algorithms were also used in this same experiment and, as expected, all algorithms presented identical learning curves as in Fig. 4. Nevertheless, the RLS algorithm diverged after approximately 1300 samples. Conversely, the proposed algorithms showed no sign of divergence.

8.2 Computational complexity issues

In order to present the computational complexity of the multichannel algorithms discussed in this work, let us first classify them in a comprehensive way. Table 5 shows the classification of the multichannel algorithms based on backward prediction errors: *a posteriori* or *a priori*, block-channel or sequential-channel, equal or multiple order, and order recursive (lattice) or transversal.

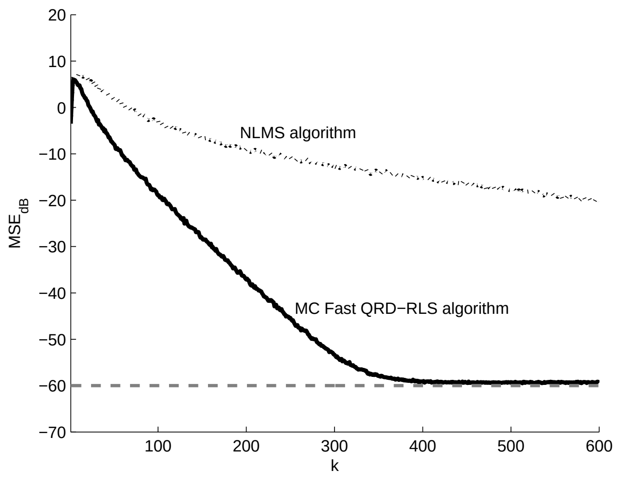

Fig. 4. Learning curves.

Table 5
Classification of the MCFQRD-RLS algorithms.

Error Type	Approach and Order		Structure	References	Algorithm
MCFQR POS_B	**BLOCK-CHANNEL**	*Equal Order*	Lattice	[15]	**1**†
			Transversal	[5,4,15]	**2**†
		Multiple Order	Lattice	—	**3**
			Transversal	[16,3]	**4**†
	SEQUENTIAL-CHANNEL	*Equal Order*	Lattice	Implicit in **7** [19]	**5**†
			Transversal	Sugested in [5]	**6**†
		Multiple Order	Lattice	[19]	**7**†
			Transversal	[20,3]	**8**†
MCFQR PRI_B	**BLOCK-CHANNEL**	*Equal Order*	Lattice	[6]	**9**
			Transversal	[6,4,15]	**10**†
		Multiple Order	Lattice	—	**11**
			Transversal	[16]	**12**†
	SEQUENTIAL-CHANNEL	*Equal Order*	Lattice	Implicit in **15** [6]	**13**
			Transversal	Implicit in **16** [6]	**14**
		Multiple Order	Lattice	[6]	**15**
			Transversal	[6]	**16**

† algorithms described in this work.

Following the classification on Table 5, Table 6 presents the computational complexity in terms of number of multiplications, divisions, and square-roots per sample for each algorithm. Note that for the case of single channel algorithms, the computational complexities of Fast QRD-RLS algorithms are lower by one order, $\mathcal{O}(P)$, than those of the conventional QRD-RLS and the Inverse QRD-RLS algorithms, $\mathcal{O}(P^2)$.

From Table 6 we can observe the following: the sequential-channel lattice

Table 6
Computational complexity of Multichannel Fast QRD-RLS algorithms, according to Table 5.

ALGORITHM	MULTIPLICATIONS	DIVISIONS	SQUARED ROOTS
Algs. **2, 4** [5,4,16], summarized in Table 2	$4NM^2+11NM+$ $5M^2+6M+7N-$ $(4M^2+6M)\sum_{i=1}^{M}(p_i-i)$	$2NM+2M+N-$ $2M\sum_{i=1}^{M}(p_i-i)$	$2NM+M+N-$ $2M\sum_{i=1}^{M}(p_i-i)$
Algs. **10, 12** [6,4,16], summarized in Table 3	$4NM^2+11NM+$ $5M^2+6M+9N-$ $(4M^2+6M)\sum_{i=1}^{M}(p_i-i)$	$2NM+3M+2N-$ $2M\sum_{i=1}^{M}(p_i-i)+2$	$2NM+M+N-$ $2M\sum_{i=1}^{M}(p_i-i)$
Alg. **1** [15], summarized in Tab 4	$4M^3N+17M^2N+$ $12MN+5M^2+5M$	$2M^2N+3MN+2M$	$M^2N+2MN+M$
Alg. **9** [6]	$4M^3N+17M^2N+$ $14MN+5M^2+6M$	$2M^2N+5MN+3M$	$M^2N+2MN+M$
Algs. **6, 8** [20], summarized in Tab 7	$14NM+13M+$ $5N-9\sum_{i=1}^{M}p_i$	$3NM+4M-$ $3\sum_{i=1}^{M}p_i$	$2NM+3M-$ $2\sum_{i=1}^{M}p_i$
Algs. **14, 16** [6]	$15NM+14M+$ $5N-10\sum_{i=1}^{M}p_i$	$4NM+5M-$ $4\sum_{i=1}^{M}p_i$	$2NM+3M-$ $2\sum_{i=1}^{M}p_i$
Algs. **5, 7** [19], summarized in Tab 8	$14NM+13M+$ $5N-9\sum_{i=1}^{M}p_i$	$4NM+5M-$ $4\sum_{i=1}^{M}p_i$	$2NM+3M-$ $2\sum_{i=1}^{M}p_i$
Algs. **13, 15** [6]	$15NM+14M+$ $5N-10\sum_{i=1}^{M}p_i$	$5NM+6M-$ $5\sum_{i=1}^{M}p_i$	$2NM+3M-$ $2\sum_{i=1}^{M}p_i$

algorithms proposed in [19] (Algorithm 5 and Algorithm 7) have a lower computational complexity (multiplications and divisions) than those based on *a priori* backward prediction errors updating proposed in [6] (*direct* and *recursive forms*, Algorithm 13 and Algorithm 15). It can also be seen that their order recursiveness has a cost in terms of the computational complexity when compared to its transversal (or direct form) counterpart in [20]. The most attractive algorithms, from a computational complexity point of view, are Algorithm 6 (equal order) and Algorithm 8 (multiple order) [20]. Considering Algorithm 8 [20] for channels of equal orders, i.e., $N_1 = N_2 = \cdots = N_M = K$ and $P = KM$, it can be observed from Table 6 that this algorithm is of $O(M^2)$ computational complexity, lower by one order of magnitude when compared to the $O(M^3)$ block-type multichannel algorithms of [6], [4] and [15].

Finally, it can be added that all new algorithms, in the simulations carried out in this work, have shown no signs of divergence. This was somehow expected for algorithms employing numerically stable Givens rotations to perform QR decomposition when updating backward prediction errors.

9 Conclusions

The study of multichannel fast QRD-RLS (MC-FQRD-RLS) algorithms may be difficult not only because of the complex equations employed, but also due to vast notations used by different authors. In an attempt to clarify the differences among the many versions available, we introduced a classification of these algorithms. A total of nine algorithms are introduced or described in this paper from which seven are based on *a posteriori* error vector updating, three block-channel and four sequential-channel approaches, and two are block-channel algorithms based on the updating of *a priori* error vector. The classification also revealed the possibility of two unpublished algorithms belonging to this family.

In particular, a general formulation for block-channel multiple-order multichannel fast QRD-RLS algorithms was introduced. The new formulation provides block-type multichannel algorithms that are capable of processing all channels simultaneously facilitating parallel implementations. Both *a posteriori* and *a priori* versions were derived. Then an order recursive MC-FQRD-RLS algorithm based on *a posteriori* error updating was presented. This new algorithm exhibits the lowest complexity among known order recursive MC-FQRD-RLS algorithms, while keeping all desirable numerical properties of its family. We also addressed the lattice version of the MC-FQRD-RLS algorithm based on the *a posteriori* backward error updating. Its order recursiveness and stability are interesting features and this algorithm can be used in a wide range of applications, many of them in the field of telecommunications. The new lattice algorithm presents the same converge properties as the one of [6], and also saves computational load which makes it attractive as the numbers of channels and coefficients per channel increase.

Finally, a multiple-order multichannel *a posteriori* sequential-channel algorithm was introduced which has lower computational complexity when compared to its *a priori* counterpart.

Appendix A: Proof of Equation (26)

Proof 1 *From* (14)*, it is clear that* $\boldsymbol{E}_f(k+1)$ *is the Cholesky factor of* $[\tilde{\boldsymbol{e}}_{fq1} \quad \lambda^{1/2}\boldsymbol{E}_f^T(k)]^T$ *[21]. Consequently, we can write*

$$\boldsymbol{E}_f^T(k+1)\boldsymbol{E}_f(k+1) = \begin{bmatrix} \tilde{\boldsymbol{e}}_{fq1}^T(k+1) \\ \lambda^{1/2}\boldsymbol{E}_f(k) \end{bmatrix}^T \times \begin{bmatrix} \tilde{\boldsymbol{e}}_{fq1}^T(k+1) \\ \lambda^{1/2}\boldsymbol{E}_f(k) \end{bmatrix}$$
$$= \tilde{\boldsymbol{e}}_{fq1}(k+1)\tilde{\boldsymbol{e}}_{fq1}^T(k+1) + \lambda\boldsymbol{E}_f^T(k)\boldsymbol{E}_f(k) \qquad (64)$$

*The above equation is the product form of (*14*). Premultiply and post multiply (*64*) by* $\boldsymbol{E}_f^{-T}(k+1)\gamma^2(k)$ *and* $\boldsymbol{E}_f^{-1}(k+1)$*, respectively. Then, after some algebraic manipulations, we have*

$$\gamma^2(k)\boldsymbol{I} = \boldsymbol{p}(k+1)\boldsymbol{p}^T(k+1) + \boldsymbol{\Psi} \qquad (65)$$

where $\boldsymbol{\Psi} = \lambda\gamma^2(k)\boldsymbol{E}_f^{-T}(k+1)\boldsymbol{E}_f^T(k)\boldsymbol{E}_f(k)\boldsymbol{E}_f^{-1}(k+1)$.

*Finally, after premultiplying and post multiplying (*65*) by* $\boldsymbol{p}^T(k+1)$ *and* $\boldsymbol{p}(k+1)$*, respectively, we obtain*

$$\begin{aligned}\gamma^2(k) &= \boldsymbol{p}^T(k+1)\boldsymbol{p}(k+1) + \frac{\boldsymbol{p}^T(k+1)\boldsymbol{\Psi}\boldsymbol{p}(k+1)}{\boldsymbol{p}^T(k+1)\boldsymbol{p}(k+1)} \\ &= \boldsymbol{p}^T(k+1)\boldsymbol{p}(k+1) + *^2\end{aligned} \qquad (66)$$

*The expression in (*66*) can be regarded as a Cholesky product. Hence, it can be factored as*

$$\begin{bmatrix} \gamma(k) \\ \boldsymbol{0} \end{bmatrix} = \boldsymbol{Q} \begin{bmatrix} * \\ \boldsymbol{p}(k+1) \end{bmatrix} \qquad (67)$$

where $\boldsymbol{Q}$ *is an orthogonal matrix.*

*If we recall our starting point in (*14*), we can see that* $\boldsymbol{Q}$ *is related to* $\overline{\boldsymbol{Q}}_f(k+1)$*. Moreover, from the knowledge of the internal structure of* $\overline{\boldsymbol{Q}}_f(k+1)$*, we can conclude that* $\boldsymbol{Q} = \overline{\boldsymbol{Q}}_f^T(k+1)$ *satisfies (*67*) leading to (*26*). This concludes the proof. Vector* $\boldsymbol{p}(k+1)$ *can be easily obtained from (*26*) because* $\gamma(k)$ *and* $\overline{\boldsymbol{Q}}_f(k+1)$ *are known quantities.*

□

Appendix B: The proposed algorithms

Tables 7 and 8 show the complex-valued versions of the Multiple Order Sequential-type MCFQRD_POS_B Algorithm [20] and the Lattice Multiple Order Sequential MCFQRD_POS_B Algorithm [19], respectively.

Table 7
The Multiple Order Sequential-type MCFQRD_POS_B Algorithm [20].

```
Initializations:
$\boldsymbol{d}_{fq2}^{(i)} = zeros(P,1)$;   $\boldsymbol{f}^{(M)}(0) = \boldsymbol{0}$;   $\boldsymbol{d}_{q2} = \boldsymbol{0}$;   $\gamma_P^{(0)}(0) = 1$;
$e_{f_P}^{(i)}(0) = \mu$; $i = 1, 2, \cdots, M$,  all cosines = 1,   and all sines = 0.
for $k = 1, 2, \cdots$
{ $\gamma_0^{(1)} = 1$;   $e_{q1}^{(0)}(k+1) = d(k+1)$;
  for $i = 1 : M$,
  { $e_{fq1_0}^{(i)}(k+1) = x_i(k+1)$;
    for $j = 1 : P$, % Obtaining $e_{fq1}^{(i)}(k+1)$ and $\boldsymbol{d}_{fq2}^{(i)}(k+1)$:
    {$e_{fq1_j}^{(i)}(k+1) = \cos\left[\theta_j^{(i-1)}(k)\right] e_{fq1_{j-1}}^{(i)}(k+1) + \lambda^{1/2}\sin\left[\theta_j^{(i-1)}(k)\right] \boldsymbol{d}_{fq2_{P-j+1}}^{(i)}(k)$;
      $\boldsymbol{d}_{fq2_{P-j+1}}^{(i)}(k+1) = \lambda^{1/2}\cos\left[\theta_j^{(i-1)}(k)\right] \boldsymbol{d}_{fq2_{P-j+1}}^{(i)}(k) - \sin^*\left[\theta_j^{(i-1)}(k)\right] e_{fq1_{j-1}}^{(i)}(k+1)$;
    }
    $\|e_{f_P}^{(i)}(k+1)\| = \sqrt{\left(\lambda^{1/2}\|e_{f_P}^{(i)}(k)\|\right)^2 + \|e_{fq1_P}^{(i)}(k+1)\|^2}$;
    for $j = P : -1 : p_i$, % Obtaining $\boldsymbol{Q'}_{\theta f}^{(i)}(k+1)$:
    {$e_{f_{j-1}}^{(i)}(k+1) = \sqrt{\|e_{f_j}^{(i)}(k+1)\|^2 + \|\boldsymbol{d}_{fq2_{P-j+1}}^{(i)}(k+1)\|^2}$;
      $\cos\theta'^{(i)}_{f_j}(k+1) = \|e_{f_j}^{(i)}(k+1)/e_{f_{j-1}}^{(i)}(k+1)\|$;
      $\sin\theta'^{(i)}_{f_j}(k+1) = \left[\cos\theta'^{(i)}_{f_j}(k+1)\ \boldsymbol{d}_{fq2_{P-j+1}}^{(i)}(k+1)/e_{f_j}^{(i)}(k+1)\right]^*$;
    }
    $p_P^{(i)}(k+1) = \gamma_P^{(i-1)}(k)\left[e_{fq1_P}^{(i)}(k+1)\right]^* / \|e_{f_P}^{(i)}(k+1)\|$;
    for $j = P : -1 : p_i$, % Obtaining $\boldsymbol{f}^{(i)}(k+1)$:
    {$\boldsymbol{f}_{P-j+1}^{(i)}(k+1) = \cos\theta'^{(i)}_{f_j}(k+1)\boldsymbol{f}_{P-j+2}^{(i-1)}(k+1) - \left[\sin\theta'^{(i)}_{f_j}(k+1)\right]^* p_j^{(i)}(k+1)$;
      $p_{j-1}^{(i)}(k+1) = \sin\theta'^{(i)}_{f_j}(k+1)\boldsymbol{f}_{P-j+2}^{(i-1)}(k+1) + \cos\theta'^{(i)}_{f_j}(k+1)p_j^{(i)}(k+1)$;
    }
    $\boldsymbol{f}_{P+1-p_i+1}^{(i)}(k+1) = p_{p_i-1}^{(i)}(k+1)$;
    for $j = p_i : P$, % Obtaining $\boldsymbol{Q}_\theta^{(i)}(k)$:
    {$\sin\theta_j^{(i)}(k) = -\left[\boldsymbol{f}_{P-j+2}^{(i)}(k+1)\right]^* / \gamma_{j-1}^{(i)}$;

      $\cos\theta_j^{(i)}(k) = \sqrt{1 - \|\sin\theta_j^{(i)}(k)\|^2}$;
      $\gamma_j^{(i)}(k) = \cos\theta_j^{(i)}(k)\gamma_{j-1}^{(i)}(k+1)$;
    }
  } for $i$
  for $j = 1 : P$, % Joint process estimation:
  {$e_{q1}^{(j)}(k+1) = \cos\theta_j^{(0)}(k+1)\boldsymbol{e}_{q1}^{(j-1)}(k+1) + \lambda^{1/2}\sin\theta_j^{(0)}(k+1)\boldsymbol{d}_{q2}^{(P-j+1)}(k)$;
    $\boldsymbol{d}_{q2}^{(P-j+1)}(k+1) = \lambda^{1/2}\cos\theta_j^{(0)}(k+1)\boldsymbol{d}_{q2}^{(P-j+1)}(k) - \left[\sin\theta_j^{(0)}(k+1)\right]^* e_{q1}^{(j-1)}(k+1)$;
  }
  $\varepsilon(k+1) = \left[e_{q1}^{(P)}(k+1)\right]^* / \gamma_P^{(0)}(k+1)$;
} for $k$
```

Obs.: $\theta_j^{(M)}(k) = \theta_j^{(0)}(k+1)$ and $\boldsymbol{f}_{P-j+2}^{(M)}(k) = \boldsymbol{f}_{P-j+2}^{(0)}(k+1)$.

The *asterisk* $(*)$ denotes complex conjugation.

Table 8
The Lattice Multiple Order Sequential
MCFQRD_POS_B Algorithm [19].

```
Initializations:
$\boldsymbol{d}_{fq2}^{(i)} = zeros(P,1)$;   $\boldsymbol{f}^{(M)}(0) = \boldsymbol{0}$;   $\boldsymbol{d}_{q2} = \boldsymbol{0}$;   $\gamma_P^{(0)}(0) = 1$;
$e_{f_P}^{(i)}(0) = \mu$; $i = 1, 2, \cdots, M$,  all cosines = 1,   and all sines = 0.
for $k = 1, 2, \cdots$
{ $\gamma_0^{(1)} = 1$;   $e_{q1}^{(0)}(k+1) = d^*(k+1)$;
  $|e_0^{(1)}(k+1)| = \sqrt{\left(\lambda^{1/2}|e_0^{(1)}(k)|\right)^2 + |x_1(k+1)|^2}$;
  $\boldsymbol{f}_{P+1}^{(1)}(k+1) = [x_1(k+1)]^* / |e_0^{(1)}(k+1)|$;
  for $i = 1 : M$,
   { $e_{fq1_0}^{(i)}(k+1) = x_i(k+1)$
     for $j = 1 : P$,
     { $e_{fq1_j}^{(i)}(k+1) = \cos\left[\theta_j^{(i-1)}(k)\right] e_{fq1_{j-1}}^{(i)}(k+1) + \lambda^{1/2}\sin\left[\theta_j^{(i-1)}(k)\right] \boldsymbol{d}_{fq2_{P-j+1}}^{(i)}(k)$;
       $\boldsymbol{d}_{fq2_{P-j+1}}^{(i)}(k) = \lambda^{1/2}\cos\left[\theta_j^{(i-1)}(k)\right] \boldsymbol{d}_{fq2_{P-j+1}}^{(i)}(k) - \sin\left[\theta_j^{(i-1)}(k)\right]^* e_{fq1_{j-1}}^{(i)}(k+1)$;
       if $j \geq p_i - 1$,
         $|e_{f_j}^{(i)}(k+1)| = \sqrt{\left(\lambda^{1/2}|e_{f_j}^{(i)}(k)|\right)^2 + |e_{fq1_P}^{(i)}(k+1)|^2}$;
         $p_j^{(i)}(k+1) = \frac{\gamma_j^{(i-1)}(k)\left[e_{fq1_j}^{(i)}(k+1)\right]^*}{|e_{f_j}^{(i)}(k+1)|}$;
         if $j = p_i - 1$,
          $\boldsymbol{f}_{P+1-j+1}^{(i)}(k+1) = p_j^{(i)}(k+1)$;
       if $j > p_i - 1$,
         $\cos\theta'^{(i)}_{f_j}(k+1) = |e_{f_j}^{(i)}(k+1)| / |e_{f_{j-1}}^{(i)}(k+1)|$;
         $\sin\theta'^{(i)}_{f_j}(k+1) = \left[\cos\theta'^{(i)}_{f_j}(k+1)\boldsymbol{d}_{fq2_{P-j+1}}^{(i)}(k+1) / e_{f_j}^{(i)}(k+1)\right]^*$;
         $\boldsymbol{f}_{P-j+1}^{(i)}(k+1) = \cos\theta'^{(i)}_{f_j}(k+1)\boldsymbol{f}_{P-j+2}^{(i-1)}(k+1) - \sin\left[\theta'^{(i)}_{f_j}(k+1)\right]^* p_j^{(i)}(k+1)$;
         $\sin\theta_j^{(i)}(k) = -\left[\boldsymbol{f}_{P-j+2}^{(i)}(k+1)\right]^* / \gamma_{j-1}^{(i)}$;
         $\cos\theta_j^{(i)}(k) = \sqrt{1 - |\sin\theta_j^{(i)}(k)|^2}$;
         $\gamma_j^{(i)}(k) = \cos\theta_j^{(i)}(k)\gamma_{j-1}^{(i)}(k+1)$;
     } for j
  } for $i$
  for $j = 1 : P$ % Joint process estimation:
  { $e_{q1}^{(j)}(k+1) = \cos\theta_j^{(0)}(k+1)\boldsymbol{e}_{q1}^{(j-1)}(k+1) + \lambda^{1/2}\sin\theta_j^{(0)}(k+1)\boldsymbol{d}_{q2}^{(P-j+1)}(k)$;
     $\boldsymbol{d}_{q2}^{(P-j+1)}(k+1) = \lambda^{1/2}\cos\theta_j^{(0)}(k+1)\boldsymbol{d}_{q2}^{(P-j+1)}(k) - \sin\left[\theta_j^{(0)}(k+1)\right]^* e_{q1}^{(j-1)}(k+1)$;
  }
  $\varepsilon(k+1) = \left[e_{q1}^{(P)}(k+1)\right]^* / \gamma_P^{(0)}(k+1)$;  % the *a priori* error
} for $k$
```

Obs.: The *asterisc* $(*)$ denotes complex conjugation.
$\theta_j^{(M)}(k) = \theta_j^{(0)}(k+1)$ and $\boldsymbol{f}_{P-j+2}^{(M)}(k) = \boldsymbol{f}_{P-j+2}^{(0)}(k+1)$.

Acknowledgments

The authors would like to thank CAPES, CNPq, FAPERJ (Brazil), and Academy of Finland, Smart and Novel Radios (SMARAD) Center of Excellence (Finland), for partial support of this work. António Ramos thanks Mr. Emitério Ramos for his kind support. We also would like to thank Mr. Mobien Shoaib for spending time on reviewing and commenting the paper while in phase of preparation.

References

[1] N. Kalouptsidis, S. Theodoridis, Adaptive Systems Identification and Signal Processing Algorithms, Prentice Hall, Upper Saddle River, USA, 1993.

[2] V. J. Mathews, G. L. Sicuranza, Polynomial Signal Processing, Wiley–Intercience: John Wiley and Sons, New York, USA, 2000.

[3] M. A. Syed, V. J. Mathews, QR-decomposition based algorithms for adaptive Volterra filtering, IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications 40 (6) (1993) 372–382.

[4] C. A. Medina S., J. A. Apolinário Jr., M. G. Siqueira, A unified framework for multichannel fast QRD-LS adaptive filters based on backward prediction errors, in: Proc. 45th Midwest Symposium on Circuits and Systems (MWSCAS'O2), Tulsa, USA, Vol. 3, 2002, pp. 668–671.

[5] M. G. Bellanger, P. A. Regalia, The FLS-QR algorithm for adaptive filtering: the case of multichannel signals, (EURASIP) Signal Processing 22 (2) (1991) 115–126.

[6] A. A. Rontogiannis, S. Theodoridis, Multichannel fast QRD-LS adaptive filtering: New technique and algorithms, IEEE Transactions on Signal Processing 46 (11) (1998) 2862–2876.

[7] J. M. Cioffi, The fast adaptive ROTOR's RLS algorithm, IEEE Transactions on Acoustics, Speech, and Signal Processing 38 (4) (1990) 631–653.

[8] P. A. Regalia, M. Bellanger, On the duality between fast QR methods and lattice methods in least squares adaptive filtering, IEEE Transactions on Signal Processing 39 (4) (1991) 879–891.

[9] J. A. Apolinário Jr., M. G. Siqueira, P. S. R. Diniz, Fast QR algorithms based on backward prediction errors: a new implementation and its finite precision performance, Birkhäuser, Circuits, Systems, and Signal Processing 22 (4) (2003) 335–349.

[10] J. A. Apolinário Jr., P. S. R. Diniz, A new fast QR algorithm based on *a priori* errors, IEEE Signal Processing Letters 4 (11) (1997) 307–309.

[11] M. D. Miranda, M. Gerken, An hybrid QR-lattice least squares algorithm using *a priori* errors, in: Proc. 38th Midwest Symposium on Circuits and Systems (MWSCAS'95), Rio de Janeiro, Brazil, Vol. 2, 1995, pp. 983–986.

[12] A. A. Rontogiannis, S. Theodoridis, New fast inverse QR least squares adaptive algorithms, in: Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'95), Detroit, USA, Vol. 2, 1995, pp. 1412–1415.

[13] P. S. R. Diniz, Adaptive Filtering: Algorithms and Practical Implementations, Kluwer Academic Publishers, Boston,USA, (2nd Edition) 2002.

[14] M. Harteneck, J. G. McWhirter, I. K. Proudler, R. W. Stewart, Algorithmically engineered fast multichannel adaptive filter based qr-rls, IEE Proc.-Vis. Image Signal Process. 146 (1) (1999) 7–13.

[15] A. L. L. Ramos, J. A. Apolinário Jr., A lattice version of the multichannel FQRD algorithm based on *a posteriori* backward errors, in: Proc. 11th Internacional Conference on Telecommunications, Fortaleza, Brazil, ICT'2004, LNCS, Vol. 1, pp. 488–497.

[16] A. L. L. Ramos, J. A. Apolinário Jr., S. Werner, A general approach to the derivation of block multichannel fast QRD-RLS algorithms, in: Proc. European Signal Processing Conference EUSIPCO'2005, Antalya, Turkey, Vol. 1, 2005, pp. 1–4.

[17] A. L. Ghirnikar, S. T. Alexander, Performance and implementation of the inverse QR adaptive filter, in: Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP-92, San Francisco, USA, Vol. 4, 1992, pp. 29–32.

[18] S. T. Alexander, A. L. Ghirnikar, A method for recursive least squares filtering based upon an inverse QR decomposition, IEEE Transactions on Signal Processing 41 (1993) 20–30.

[19] A. L. L. Ramos, J. A. Apolinário Jr., M. G. Siqueira, A new order recursive multiple order multichannel fast QRD algorithm, in: Proc. 38th Midwest Asilomar Conference on Signals, Systems, and Computers, Pacific Grove, USA, Vol. 1, 2004, pp. 965–969.

[20] A. L. L. Ramos, J. A. Apolinário Jr., A new multiple order multichannel fast QRD algorithm and its application to non-linear system identification, in: Proc. XXI Simpósio Brasileiro de Telecomunicações, SBT 2004, Belém, Brazil, Vol. 1, pp. 1–4.

[21] G. H. Golub, C. F. V. Loan, Matrix Computations, The Johns Hopkins University Press, Baltimore, USA, (3rd edition) 1983.