A New Multiple Order Multichannel Fast QRD Algorithm and its Application to Non-linear System Identification

António L. L. Ramos and José A. Apolinário Jr.

***Abstract*— Multichannel versions of the Fast QRD–RLS algorithms have been mostly addressed in the literature for channels of equal orders. However, in many applications, such as in the case of the Volterra filtering problem, one has to deal with channels of unequal orders. This, along with the fact that the Fast QRD–RLS algorithms based on backward prediction errors are well known for their good numerical behavior and low complexity, has motivated the development of multichannel Fast QRD–RLS algorithms that cope with these cases. This paper presents a new multichannel Fast QRD–RLS algorithm based on *a posteriori* backward error updating, comprising scalar operations only, that attains both cases of equal and unequal channel orders.**

I. INTRODUCTION

Digital processing of multichannel signals using adaptive filters has recently found a variety of new applications including color image processing, multispectral remote sensing imagery, biomedicine, channel equalization, stereophonic echo cancellation, multidimensional signal processing, Volterra–type nonlinear system identification, and speech enhancement [1]. This increased number of applications has spawned a renewed interest in efficient multichannel algorithms. One class of algorithms, known as multichannel Fast QRD-RLS adaptive algorithms based on backward errors updating [2], [3], has become an attractive option because of their fast convergence and reduced computational complexity.

Unified formulations of Fast QRD-RLS algorithms are available in [4], for the single channel case, and in [5], for the multichannel case. In this paper, a new multiple order Multichannel Fast QRD-RLS algorithm is developed from the fixed order multichannel algorithm recently proposed in [6], using an approach similar to the one used in [3] for the *a priori* version. This new multichannel Fast QRD algorithm, using the *a posteriori* backward error updating, can be used in problems dealing with channels of equal or unequal orders while comprising scalar operations only.

The QRD-RLS family of Multichannel algorithms uses the least-squares (LS) objective function defined as

$$\xi_{LS}(k) = \sum_{i=0}^{k} \lambda^{k-i} e^2(i) = \boldsymbol{e}^T(k)\boldsymbol{e}(k) \tag{1}$$

where $\boldsymbol{e}(k) = \begin{bmatrix} e(k) & \lambda^{1/2}e(k-1) & \cdots & \lambda^{k/2}e(0) \end{bmatrix}^T$ is a weighted error vector and may be represented as follows.

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

where

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

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 initially defined as the number of filter coefficients per channel (fixed order), M is the number of input channels, and $\boldsymbol{w}_N(k)$ is the $MN \times 1$ coefficient vector at time instant k.

If $\boldsymbol{U}_N(k)$ is the Cholesky factor of the $(k+1) \times M$ input data matrix $\boldsymbol{X}_N(k)$, obtained through the Givens rotation matrix $\boldsymbol{Q}_N(k)$, then

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

and the optimal coefficient vector, $\boldsymbol{w}_N(k)$, is obtained by making $\boldsymbol{e}_{q2}(k)$ a null vector. In order to cope with the multiple order case, the input signal vector needs to be redefined.

This paper is organized as follows. In Section II, a new multiple-order Multichannel Fast QRD based on the

The authors thank CAPES, FAPERJ, and CNPq for partial funding of this paper.

A. L. L. Ramos and J. A. Apolinário Jr. are with the Departamento de Engenharia Elétrica, Instituto Militar de Engenharia, Praça General Tibúrcio 80, Rio de Janeiro, RJ, 22.290-270 (e-mail: antonioluis@ime.eb.br and apolin@ieee.org).

updating of the *a posteriori* error vector is introduced. Simulation results and conclusions are Summarized in Sections III and IV, respectively.

II. The New Multiple Order Multichannel Fast QRD-RLS Algorithm

To derive this new algorithm, which is based on the updating of the *a posteriori* backward error vector (more about the Multichannel Fast QRD-RLS algorithm based on the backward prediction update and its *a posteriori* direct and lattice versions can be found in [6]), a new approach is adopted to construct the input vector $\boldsymbol{x}_N(k)$.

As we shall see later in this work, this new algorithm, beside the advantage of dealing with channels of unequal orders, has another major one, viz, a lower computational complexity when compared to previously proposed multichannel algorithms. The following notation is adopted:

M	is the number of input channels;
$N_1, N_2, \cdots, N_M$	are the number of *taps* in the *tapped delay–lines* of each input channel;
$N = \sum_{r=1}^{M} N_r$	Overall number of *taps*.

Without loss of generality, it is assumed here that $N_1 \geq N_2 \geq \cdots \geq N_M$.

As previously mentioned, the starting point for the derivation of this new algorithm is the construction of the input vector such that the general case of equal or unequal order is attained. It is also taken into consideration the fact that M steps are executed for each time iteration of the algorithm. That means that the M channels are processed separately but they are interdependent: the quantities collected after the *i*-th channel is processed are used as initial values for the processing of the (*i+1*)-th channel and so on. Finally, the quantities collected during the processing of the last channel, in a given instant k, are used as initial values for the processing of the first channel in $k+1$. This will become clear during the derivation of the algorithm.

A. Redefining the input vector

From the fact that the M channels are processed separately, the updating of the input vector is performed likewise: in a given time instant k, we have vector $\boldsymbol{x}_N(k)$[1] from which we successively obtain $\boldsymbol{x}_{N+1}(k+1)$ by appending the most recent sample of channel one at time instant $k+1$, $\boldsymbol{x}_{N+2}(k+1)$ by appending the most recent sample of channel two, and so on. At the and of this process, we have the updated vector $\boldsymbol{x}_{N+M}(k+1)$.

[1]The subscript $_N$ denotes the N-th order vector.

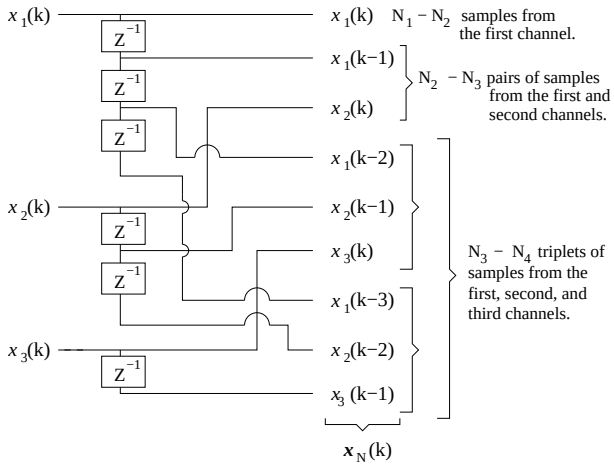

Fig. 1. Obtaining the input vector.

Actually, this is not that simple because the position to be occupied by the newer samples from each channel in the updated vector $\boldsymbol{x}_{N+M}(k+1)$ must be carefully determined. The vector $\boldsymbol{x}_N(k)$, used as the starting point to obtain $\boldsymbol{x}_{N+M}(k+1)$ is construct as follows: we first choose $N_1 - N_2$ *samples* from the first channel to be the leading elements of $\boldsymbol{x}_N(k)$, 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 far till the $N_M - N_{M+1}$ M*-ples of samples* of all channels. It is assumed that $N_{M+1} = 0$.

Fig. 1 shows an example where vector $\boldsymbol{x}_N(k)$ is obtained for a configuration with $M = 3$, $N_1 = 4$, $N_2 = 3$, $N_3 = 2$, and $N_4 = 0$; thus, $N = 4 + 3 + 2 + 0 = 9$. By carefully observing the diagram of this figure, one can realize which position will be occupied by each new sample of each channel to form $\boldsymbol{x}_{N+M}(k+1)$. This position p_i of the most recent sample of the *i*-th channel can be expressed compactly as [3] $p_i = \sum_{r=1}^{i-1} r(N_r - N_{r+1}) + i$, for $i = 1, 2, \cdots, M$. Moreover, the M successive input vectors for a given instant k, obtained from $\boldsymbol{x}_N(k)$, can be defined as follows.

$$\boldsymbol{x}_{N+1}^T(k+1) = \begin{bmatrix} x_1(k+1) & \boldsymbol{x}_N^T(k) \end{bmatrix} \quad (5)$$

$$\boldsymbol{x}_{N+i}^T(k+1) = \begin{bmatrix} x_i(k+1) & \boldsymbol{x}_{N+1-i}^T(k+1) \end{bmatrix} \boldsymbol{P}_i \quad (6)$$

where $\boldsymbol{P}_i$ is a permutation matrix which takes the most recent sample $x_i(k+1)$ of the *i*-th channel to the position p_i, after left shifting the first $p_i - 1$ elements of $\boldsymbol{x}_{N-i+1}^T(k+1)$. After concluding this process for the M channels, it can be observed that $\boldsymbol{x}_{N+M}^T(k+1) = [\boldsymbol{x}_N^T(k+1) \quad x_1(k-N_1+1) \cdots \quad x_M(k-N_M+1)]$ which clearly means that the first N elements of $x_{N+M}^T(k+1)$ provide the input vector of the next iteration. We can now define the input data

matrices as follows.

$$\boldsymbol{X}_{N+i}(k) = \begin{bmatrix} \boldsymbol{x}_{N+i}^T(k) \\ \lambda^{1/2}\boldsymbol{x}_{N+i}^T(k-1) \\ \vdots \\ \lambda^{k/2}\boldsymbol{x}_{N+i}^T(0) \end{bmatrix} \quad i = 1, 2, \cdots, M \quad (7)$$

With $\boldsymbol{U}_{N+i}(k)$ being the Cholesky factor of $\boldsymbol{X}_{N+i}(k)$, we can define the *a posteriori* backward error vector, $\boldsymbol{f}_{N+i}(k+1)$, as

$$\begin{aligned} \boldsymbol{f}_{N+i}(k+1) &= \boldsymbol{U}_{N+i}^{-T}(k+1)\boldsymbol{x}_{N+i}(k+1) \\ &\quad \text{for } i = 1, 2, \cdots, M. \end{aligned} \quad (8)$$

From (8) and the definition of the input vector, we can write

$$\boldsymbol{f}_{N+M}(k+1) = \begin{bmatrix} \boldsymbol{f}_N(k+1) \\ \boldsymbol{f}^{(N)}(k+1) \end{bmatrix} \quad (9)$$

where $\boldsymbol{f}^{(N)}(k+1)$ are the last M elements of $\boldsymbol{f}_{N+M}(k+1)$.

The updating of $\boldsymbol{f}_{N+i}(k+1)$ is carried out in M forward steps at each instant k:
$\boldsymbol{f}_N(k) \rightarrow \boldsymbol{f}_{N+1}(k+1) \rightarrow \boldsymbol{f}_{N+2}(k+1) \rightarrow \cdots \cdots \rightarrow \boldsymbol{f}_{N+M}(k+1)$

B. Triangularization of the information matrix

Equation (7) suggests that the updating of the information matrix is performed in M forward steps for each iteration.

1) First step ($i = 1$): $\boldsymbol{X}_{N+1}(k)$ can be defined as

$$\begin{aligned} \boldsymbol{X}_{N+1}(k) &= \begin{bmatrix} x_1(k) & \\ \lambda^{1/2}x_1(k-1) & \boldsymbol{X}_N(k-1) \\ \vdots & \\ \lambda^{k/2}x_1(0) & \boldsymbol{0}^T \end{bmatrix} \\ &= \left[\begin{array}{c|c} \boldsymbol{d}_f^{(1)}(k) & \boldsymbol{X}_N(k-1) \\ & \boldsymbol{0}^T \end{array}\right] \end{aligned} \quad (10)$$

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

If $\boldsymbol{U}_N(k-1)$ and $\boldsymbol{Q}_N^{(1)}(k)$ stand, respectively, for the Cholesky factor of $\boldsymbol{X}_N(k-1)$ and the orthogonal matrix associated to this process, we can write, from (10), that

$$\begin{bmatrix} \boldsymbol{Q}_N^{(1)}(k) & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I}_{1\times 1} \end{bmatrix} \left[\begin{array}{c|c} \boldsymbol{d}_f^{(1)}(k) & \boldsymbol{X}_N(k-1) \\ & \boldsymbol{0}^T \end{array}\right] = \begin{bmatrix} \boldsymbol{e}_{fq1}^{(1)}(k) & \boldsymbol{0} \\ \boldsymbol{d}_{fq2}^{(1)}(k) & \boldsymbol{U}_N(k-1) \\ \lambda^{k/2}x_1(0) & \boldsymbol{0}^T \end{bmatrix} \quad (11)$$

To complete the triangularization process of $\boldsymbol{X}_{N+1}(k)$ leading to $\boldsymbol{U}_{N+1}(k)$, we premultiply (11) by two other Given rotation matrices as follows.

$$\begin{aligned} \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{U}_{N+1}(k) \end{bmatrix} &= \boldsymbol{Q'}_f{}^{(1)}(k)\boldsymbol{Q}_f{}^{(1)}(k) \\ &\quad \cdot \begin{bmatrix} \boldsymbol{e}_{fq1}^{(1)}(k) & \boldsymbol{0} \\ \boldsymbol{d}_{fq2}^{(1)}(k) & \boldsymbol{U}_N(k-1) \\ \lambda^{k/2}x_1(0) & \boldsymbol{0}^T \end{bmatrix} \\ &= \boldsymbol{Q'}_f{}^{(1)}(k) \begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{d}_{fq2}^{(1)}(k) & \boldsymbol{U}_N(k-1) \\ e_{fN}^{(1)}(k) & \boldsymbol{0}^T \end{bmatrix} \end{aligned} \quad (12)$$

where $\boldsymbol{Q}_f{}^{(1)}(k)$ is the orthogonal matrix responsible for zeroing the first $k-N$ rows and $\boldsymbol{Q'}_f{}^{(1)}(k)$ completes the triangularization process by zeroing $\boldsymbol{d}_{fq2}^{(1)}(k)$ in a top down procedure against $e_{fN}^{(1)}(k)$. After removing the resulting null section in the upper part of (12), the result is:

$$\boldsymbol{U}_{N+1}(k) = \boldsymbol{Q'}_{\theta f}{}^{(1)}(k) \begin{bmatrix} \boldsymbol{d}_{fq2}^{(1)}(k) & \boldsymbol{U}_N(k-1) \\ e_{fN}^{(1)}(k) & \boldsymbol{0}^T \end{bmatrix} \quad (13)$$

From (13), we obtain the following relation that will be useful to obtain an expression for the updating of $\boldsymbol{f}_N(k)$.

$$\begin{aligned} &[\boldsymbol{U}_{N+1}(k+1)]^{-1} = \\ &\quad \begin{bmatrix} \boldsymbol{0}^T & \frac{1}{e_{fN}^{(1)}(k+1)} \\ \boldsymbol{U}_N^{-1}(k) & -\frac{1}{e_{fN}^{(1)}(k+1)}\boldsymbol{U}_N^{-1}(k)\boldsymbol{d}_{fq2}^{(1)}(k+1) \end{bmatrix} \\ &\quad \cdot \left[\boldsymbol{Q'}_{\theta f}{}^{(1)}(k+1)\right]^T \end{aligned} \quad (14)$$

From (12), we know that $\boldsymbol{Q'}_f{}^{(1)}(k)$ is the Givens rotation matrix responsible for zeroing $\boldsymbol{d}_{fq2}^{(1)}(k)$ against $e_{fN}^{(1)}(k)$. Thus, it is straightforward to write

$$\begin{bmatrix} \boldsymbol{0} \\ e_{f0}^{(1)}(k+1) \end{bmatrix} = \boldsymbol{Q'}_{\theta f}{}^{(1)}(k+1) \begin{bmatrix} \boldsymbol{d}_{fq2}^{(1)}(k+1) \\ e_{fN}^{(1)}(k+1) \end{bmatrix} \quad (15)$$

Now, recalling (8), we can use (14) and the definition of the input vector to obtain the recursive expression for $\boldsymbol{f}_{N+1}(k+1)$.

$$\begin{aligned} \boldsymbol{f}_{N+1}(k+1) &= \lambda^{-1/2}\boldsymbol{Q'}_{\theta f}{}^{(1)}(k+1) \\ &\quad \cdot \begin{bmatrix} \boldsymbol{U}_N^{-T}(k)\boldsymbol{x}_N(k) \\ \frac{x_1(k+1)}{e_{fN}^{(1)}(k+1)} - \frac{\left[\boldsymbol{U}_N^{-1}(k)\boldsymbol{d}_{fq2}^{(1)}(k+1)\right]^T\boldsymbol{x}_N(k)}{e_{fN}^{(1)}(k+1)} \end{bmatrix} \\ &= \boldsymbol{Q'}_{\theta f}{}^{(1)}(k+1) \begin{bmatrix} \boldsymbol{f}_N(k) \\ p^{(1)}(k+1) \end{bmatrix} \end{aligned} \quad (16)$$

where

$$\begin{aligned} p^{(1)}(k+1) &= \lambda^{-1/2} \\ &\cdot \left[\frac{x_1(k+1)}{e_{fN}^{(1)}(k+1)} - \frac{\left[\boldsymbol{U}_N^{-1}(k) \boldsymbol{d}_{fq2}^{(1)}(k+1) \right]^T \boldsymbol{x}_N(k)}{e_{fN}^{(1)}(k+1)} \right] \\ &= \frac{\lambda^{-1/2}}{e_{fN}^{(1)}(k+1)} \\ &\cdot \left[x_1(k+1) - \boldsymbol{w}_{f_N}^T(k+1) \boldsymbol{x}_N(k) \right] \\ &= \frac{{e_N}^{(1)}(k+1)}{\lambda^{1/2} e_{fN}^{(1)}(k+1)} \end{aligned} \quad (17)$$

with ${e_N}^{(1)}(k+1)$ being the *a posteriori* error of the forward prediction for the first channel. The updating of $\boldsymbol{d}_{fq2}^{(1)}(k)$ is performed according to

$$\begin{bmatrix} \tilde{e}_{fq1}^{(1)}(k+1) \\ \boldsymbol{d}_{fq2}^{(1)}(k+1) \end{bmatrix} = \boldsymbol{Q}_{\theta_N}^{(0)}(k) \begin{bmatrix} x_1(k+1) \\ \lambda^{1/2} \boldsymbol{d}_{fq2}^{(1)}(k) \end{bmatrix} \quad (18)$$

and the matrix $\boldsymbol{Q}_{\theta_{N+1}}^{(1)}(k+1)$, necessary in the next steps, is obtained from

$$\boldsymbol{Q}_{\theta_{N+1}}^{(1)}(k+1) \begin{bmatrix} 1 \\ \boldsymbol{0} \end{bmatrix} = \begin{bmatrix} \gamma_{N+1}^{(i)}(k+1) \\ \boldsymbol{f}_{N+1}(k+1) \end{bmatrix} \quad (19)$$

2) Following steps ($i > 1$): The input information matrix $\boldsymbol{X}_{N+i}(k)$ is related to $\boldsymbol{X}_{N+i-1}(k)$ according to

$$\boldsymbol{X}_{N+i}(k) = \begin{bmatrix} x_i(k) & \\ \lambda^{1/2} x_i(k-1) & \\ \vdots & \boldsymbol{X}_{N+i-1}(k) \\ \lambda^{k/2} x_i(0) & \end{bmatrix} \boldsymbol{P}_i \quad (20)$$

As in the first step, $\boldsymbol{X}_{N+i}(k)$ must be triangularized generating $\boldsymbol{U}_{N+i}(k)$ that corresponds to its Cholesky factor. This process is detailed as follows. If $\boldsymbol{Q}_{\theta_{N+1-i}}(k)$ stands for orthogonal matrix obtained from the QR decomposition of $\boldsymbol{X}_{N+i-1}(k)$, with $\boldsymbol{U}_{N+i-1}(k)$ being its Cholesky factor, we can write from (20)

$$\begin{aligned} &\begin{bmatrix} \boldsymbol{Q}_{\theta_{N+1-i}}(k) & \boldsymbol{0} \\ \boldsymbol{0}^T & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{X}_{N+1-i}(k) \\ \boldsymbol{0}^T \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{e}_{fq1_{N+1-i}}^{(i)}(k) & \boldsymbol{0} \\ \boldsymbol{d}_{fq2}^{(i)}(k) & \boldsymbol{U}_{N+i-1}(k) \\ \boldsymbol{0} & \boldsymbol{0}^T \end{bmatrix} \boldsymbol{P}_i \\ &= \begin{bmatrix} \boldsymbol{d}_{fq2}^{(i)}(k) & \boldsymbol{U}_{N+i-1}(k) \\ e_{f_{N+i-1}}^{(i)}(k) & \boldsymbol{0}^T \end{bmatrix} \boldsymbol{P}_i \end{aligned} \quad (21)$$

Equation (21) results from the annihilation of $\boldsymbol{e}_{fq1_{N+1-i}}^{(i)}(k)$ against the first element of the last row of the matrix, using an appropriate orthogonal factor, and removing the resulting null section.

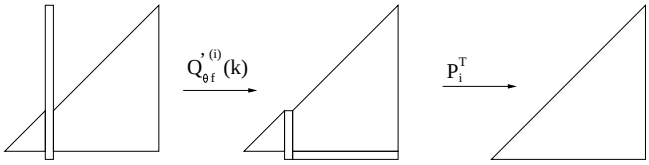

Fig. 2. Obtaining the lower triangular factor $\boldsymbol{U}_{N+i-1}(k)$.

The existence of the permutation matrix $\boldsymbol{P}_i$ in (21) prevents a direct annihilating of $\boldsymbol{d}_{fq2}^{(i)}(k)$ against $e_{f_{N+i-1}}^{(i)}(k)$ to complete the triangularization of matrix $\boldsymbol{X}_{N+1-i}(k)$. Fig. 2 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 $\boldsymbol{d}_{fq2}^{(i)}(k)$ to the *i*-th position as shown in the first part of the figure. Afterwards, a set of $N+i-p_i$ Given rotation matrices, ${\boldsymbol{Q}'_{\theta f}}^{(i)}$, are used to nullify the first $N+i-p_i$ elements of $\boldsymbol{d}_{fq2}^{(i)}(k)$ against $e_{f_{N+i-1}}^{(i)}(k)$ in a top down procedure. In order to obtain the desired triangular structure, we need another permutation factor that moves the last row of the matrix to the $N-p_i+1$ position, after the downshift of the previous $N-p_i$ rows. This permutation factor coincides with $\boldsymbol{P}_i^T$.

The positive definiteness of the lower triangular matrix $\boldsymbol{U}_{N+i-1}(k)$, obtained in the latter process, is guaranteed if its diagonal elements, along with $e_{f_{N+i-1}}^{(i)}(k)$, are positive. Recalling that $e_{f_{N+i-1}}^{(i)}(k)$ is, actually, the absolute value of the forward error, the latter is valid and, $\boldsymbol{U}_{N+i-1}(k)$ being properly initialized, its positive definiteness is guaranteed.

The procedure above can be written compactly as

$$\begin{aligned} \boldsymbol{U}_{N+i}(k) &= \boldsymbol{P}_i^T {\boldsymbol{Q}'_{\theta f}}^{(i)}(k) \\ &\cdot \begin{bmatrix} \boldsymbol{d}_{fq2}^{(i)}(k) & \boldsymbol{U}_{N+i-1}(k) \\ \boldsymbol{e}_{f_{N+i-1}}^{(i)}(k) & \boldsymbol{0}^T \end{bmatrix} \boldsymbol{P}_i \end{aligned} \quad (22)$$

From (22), it is possible to obtain the following relation.

$$\begin{aligned} [\boldsymbol{U}_{N+i}(k+1)]^{-1} &= \boldsymbol{P}_i^T \\ &\cdot \begin{bmatrix} \boldsymbol{0}^T & \frac{1}{e_{f_{N+i-1}}^{(i)}(k+1)} \\ \boldsymbol{U}_{N+i-1}^{-1}(k+1) & -\frac{\boldsymbol{U}_{N+i-1}^{-1}(k+1) \boldsymbol{d}_{fq2}^{(i)}(k+1)}{e_{f_{N+i-1}}^{(i)}(k+1)} \end{bmatrix} \\ &\cdot {\boldsymbol{Q}'^T_{\theta f}}^{(i)}(k+1) \boldsymbol{P}_i \end{aligned} \quad (23)$$

From (23), (6), and (8), it is possible to obtain the

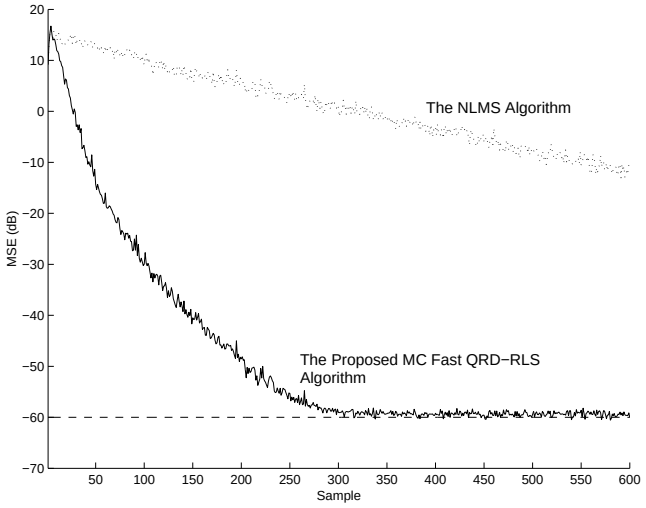

Fig. 3. Learning curves.

following recursive expression to compute $\boldsymbol{f}_{N+i}(k+1)$.

$$\boldsymbol{f}_{N+i}(k+1) = \boldsymbol{P}_i^T \boldsymbol{Q'}_{\theta f}{}^{(i)}(k+1) \begin{bmatrix} \boldsymbol{f}_{N+i-1}(k+1) \\ p^{(i)}_{N+i-1}(k+1) \end{bmatrix} \tag{24}$$

where

$$p^{(i)}_{N+i-1}(k+1) = \frac{e^{(i)}_{N+i-1}(k+1)}{e^{(i)}_{f_{N+i-1}}(k+1)} \tag{25}$$

The scalar quantity $e^{(i)}_{N+i-1}(k+1)$ is the *a posteriori* forward prediction error of the *i*-th channel. Now, looking carefully at (24) and recalling the definitions of $\boldsymbol{P}_i$ and $\boldsymbol{Q'}_{\theta f}{}^{(i)}(k+1)$, we conclude that the last $p_i - 1$ elements of $\boldsymbol{f}_{N+i}(k+1)$ and $\boldsymbol{f}_{N+i-1}(k+1)$ are identical. This is an important observation that allows a significant reduction in the computational complexity of the algorithm.

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

$$\begin{bmatrix} \tilde{e}^{(i)}_{fq1}(k+1) \\ \boldsymbol{d}^{(i)}_{fq2}(k+1) \end{bmatrix} = \boldsymbol{Q}^{(i-1)}_{\theta_{N+i-1}}(k+1) \begin{bmatrix} x_i(k+1) \\ \lambda^{1/2} \boldsymbol{d}^{(i)}_{fq2}(k) \end{bmatrix} \tag{26}$$

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

$$\boldsymbol{Q}^{(i)}_{\theta_{N+i}}(k+1) \begin{bmatrix} 1 \\ \boldsymbol{0} \end{bmatrix} = \begin{bmatrix} \gamma^{(i)}_{N+i}(k+1) \\ \boldsymbol{f}_{N+i}(k+1) \end{bmatrix} \tag{27}$$

After the main loop ($i = 1 : M$), the *join process estimation* is performed as in [6]. The computational complexity of the proposed algorithm is shown in Tab. 1 and the complete algorithm is summarized in Tab. 2. In this table we have considered the more general case of a complex implementation.

TABLE I
COMPUTATIONAL COMPLEXITY (COMPLEX ENVIRONMENT.)[2]

Algorithm	Mults.	Divs.	Sqrts
Alg. of Tab. II	$14NM + 13M$ $-9\sum_{i=1}^{M} p_i + 5N$	$3NM + 4M$ $-3\sum_{i=1}^{M} p_i$	$2NM + 3M$ $-2\sum_{i=1}^{M} p_i$
Alg. of [3]	$15NM + 14M$ $-10\sum_{i=1}^{M} p_i + 5N$	$4NM + 5M$ $-4\sum_{i=1}^{M} p_i$	$2NM + 3M$ $-2\sum_{i=1}^{M} p_i$

[2] Note that $p_i = \sum_{r=1}^{i-1} r(N_r - N_{r+1}) + i, \quad i = 1, 2, \cdots, M.$

III. SIMULATION RESULTS

The performance of the proposed algorithm is evaluated in a nonlinear system identification. The plant is a truncated second order Volterra system [7] which can be described as

$$d(k) = \sum_{n_1=0}^{L-1} w_{n_1}(k) x(k-n_1) + \sum_{n_1=0}^{L-1} \sum_{n_2=0}^{L-1} w_{n_1,n_2}(k) x(k-n_1) x(k-n_2) + \rho(k) \tag{28}$$

Equation (28) can be easily reformulated as a multichannel problem with $M = L + 1$ channels, where the most recent sample of the *i*-th 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 $L = 4$, $\lambda = 0.98$, and an SNR of $60dB$. Fig. 3 shows the learning curves for the proposed algorithm in comparison with the Normalized LMS (NLMS) [8] algorithm. The learning curve of this new multiple order multichannel algorithm is identical to its *a priori* counterpart proposed in [3] but the former presents a considerable saving in the computational burden; in the particular case simulated here, the overall saving is of $8,7\%$ in terms of complex floating point operations (multiplications and divisions). On a real environment implementation, it is possible to simplify the way of computing the rotation parameters (*sines* and *cosines*) of the rotation angles. In such a case, the overall saving is of 5,0%. Since the number of operations per output sample increases non-linearly with a positive rate as M and N increase, it is expected that the efficiency of the proposed algorithm will improve in comparison with the algorithm of [3] for larger M and N.

Although not appearing in the figure, an average of 100 independent runs of 2×10^4 samples each was carried out. The RLS [8] and the Inverse QRD-RLS [9] algorithms were also used in this same experiment and,

as expected, their speed of convergence is comparable to that of the new algorithm. But the RLS algorithm diverges after 1300 samples, approximately. Conversely, the proposed algorithm has shown no sign of divergence. It is worth mentioning that the computational complexity of the Fast QRD-RLS algorithms is lower by one order, $O(N)$, than that of the conventional QRD-RLS and the Inverse QRD-RLS algorithms, $O(N^2)$.

IV. CONCLUSIONS

The proposed algorithm, besides presenting numerical robustness, has lower computational complexity when compared to other multichannel fast QRD-RLS algorithms available in the literature. This new algorithm can be used in a large number of multichannel applications, whether equal or unequal channel orders are required. Finally, for channels of equal orders, i.e., $N_1 = N_2 = \cdots = N_M = K$ and $N = KM$, it can be realized from Tab.1 that this algorithm is of $O(M^2)$ computational complexity, lower by one order of magnitude when compared to the $O(M^3)$ block multichannel algorithms of [3], [5] and [6].

REFERENCES

[1] N. Kalouptsidis and S. Theodoridis, *Adaptive Systems Identification and Signal Processing Algorithms*, Englewood Cliffs, NJ: Prentice Hall, 1993.

[2] Maurice G. Bellanger, "The FLS-QR algorithm for adaptive filtering: The case of multichannel signals," *Signal Processing*, vol. 22, pp. 115–126, 1991.

[3] A. A. Rontogiannis and S. Theodoridis, "Multichannel fast QRD-LS adaptive filtering: New technique and algorithms," *IEEE Transactions on Signal Processing*, vol. 46, pp. 2862–2876, November 1998.

[4] J. A. Apolinário, M. G. Siqueira, and P. S. R. Diniz, "Fast QR Algorithms Based on Backward Prediction Errors: A New Implementation and Its Finite Precision Performance," *Birkhäuser, Systems, and Signal Processing*, vol. 22, no. 4, pp. 335–349, July/August 2003.

[5] C. A. Medina S., J. A. Apolinário Jr., and M. G. Siqueira, "A unified framework for multichannel fast QRD-LS adaptive filters based on backward prediction errors," *MWSCAS'O2 Tulsa–USA*, vol. 3, August 2002.

[6] A. L. L. Ramos and J. A. Apolinário Jr, "A lattice version of the multichannel FQRD algorithm based on *a posteriori* backward erros," *ICT'2004, Fortaleza, Brazil*.

[7] V. John Mathews and Giovanni L. Sicuranza, *Polynomial Signal Processing*, Wiley–Intercience: John Wiley and Sons, 2000.

[8] P. S. R. Diniz, *Adaptive Filtering: Algorithms and Practical Implementations*, 2nd Edition, Kluwer Academic Publishers, Boston, 2002.

[9] S. T. Alexander and A. L. Ghirnikar, "A method for recursive least squares filtering based upon an inverse QR decomposition," *IEEE Transactions on Signal Processing*, vol. 41, pp. 20–30, January 1993.

TABLE II

THE NEW MULTIPLE ORDER MULTICHANNEL FAST QRD-RLS ALGORITHM (COMPLEX VERSION).

Initializations:

$\boldsymbol{d}_{fq2}^{(i)} = zeros(N,1); \quad \boldsymbol{f}^{(M)}(0) = \mathbf{0}; \quad \boldsymbol{d}_{q2} = \mathbf{0};$

$\gamma_N^{(0)}(0) = 1; \quad e_{f_N}^{(i)}(0) = \mu; \quad i = 1, 2, \cdots, M.$

All cosines = 1; all sines = 0;

for $k = 1, 2, \cdots$

{ $\gamma_0^{(1)} = 1; \quad e_{q1}^{(0)}(k+1) = d(k+1);$

for $i = 1 : M$,

{ $e_{fq1_0}^{(i)}(k+1) = x_i(k+1);$

for $j = 1 : N$, % Obtaining $e_{fq1}^{(i)}(k+1)$ and $\boldsymbol{d}_{fq2}^{(i)}(k+1)$:

$$\{e_{fq1_j}^{(i)}(k+1) = \cos\left[\theta_j^{(i-1)}(k)\right] e_{fq1_{j-1}}^{(i)}(k+1) + \lambda^{1/2}\sin\left[\theta_j^{(i-1)}(k)\right] \boldsymbol{d}_{fq2_{N-j+1}}^{(i)}(k);$$

$$\boldsymbol{d}_{fq2_{N-j+1}}^{(i)}(k+1) = \lambda^{1/2}\cos\left[\theta_j^{(i-1)}(k)\right] \boldsymbol{d}_{fq2_{N-j+1}}^{(i)}(k) - \sin^*\left[\theta_j^{(i-1)}(k)\right] e_{fq1_{j-1}}^{(i)}(k+1);$$

}

$$\|e_{f_N}^{(i)}(k+1)\| = \sqrt{\left(\lambda^{1/2}\|e_{f_N}^{(i)}(k)\|\right)^2 + \|e_{fq1_N}^{(i)}(k+1)\|^2};$$

for $j = N : -1 : p_i$, % Obtaining $\boldsymbol{Q}'_{\theta f}{}^{(i)}(k+1)$:

$$\{e_{f_{j-1}}^{(i)}(k+1) = \sqrt{\|e_{f_j}^{(i)}(k+1)\|^2 + \|\boldsymbol{d}_{fq2_{N-j+1}}^{(i)}(k+1)\|^2};$$

$$\cos\theta'^{(i)}_{f_j}(k+1) = \|e_{f_j}^{(i)}(k+1)/e_{f_{j-1}}^{(i)}(k+1)\|;$$

$$\sin\theta'^{(i)}_{f_j}(k+1) = \left[\cos\theta'^{(i)}_{f_j}(k+1) \cdot \boldsymbol{d}_{fq2_{N-j+1}}^{(i)}(k+1)/e_{f_j}^{(i)}(k+1)\right]^*;$$

}

$$p_N^{(i)}(k+1) = \gamma_N^{(i-1)}(k)\left[e_{fq1_N}^{(i)}(k+1)\right]^* / \|e_{f_N}^{(i)}(k+1)\|;$$

for $j = N : -1 : p_i$, % Obtaining $\boldsymbol{f}^{(i)}(k+1)$:

$$\{\boldsymbol{f}_{N-j+1}^{(i)}(k+1) = \cos\theta'^{(i)}_{f_j}(k+1)\boldsymbol{f}_{N-j+2}^{(i-1)}(k+1) - \left[\sin\theta'^{(i)}_{f_j}(k+1)\right]^* p_j^{(i)}(k+1);$$

$$p_{j-1}^{(i)}(k+1) = \sin\theta'^{(i)}_{f_j}(k+1)\boldsymbol{f}_{N-j+2}^{(i-1)}(k+1) + \cos\theta'^{(i)}_{f_j}(k+1)p_j^{(i)}(k+1);$$

}

$$\boldsymbol{f}_{N+1-p_i+1}^{(i)}(k+1) = p_{p_i-1}^{(i)}(k+1);$$

for $j = p_i : N$, % Obtaining $\boldsymbol{Q}_\theta^{(i)}(k)$:

$$\{\sin\theta_j^{(i)}(k) = -\left[\boldsymbol{f}_{N-j+2}^{(i)}(k+1)\right]^* / \gamma_{j-1}^{(i)};$$

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

$$\gamma_j^{(i)}(k) = \cos\theta_j^{(i)}(k)\gamma_{j-1}^{(i)}(k+1);$$

}

} for i

for $j = 1 : N$, % Join process estimation:

$$\{e_{q1}^{(j)}(k+1) = \cos\theta_j^{(0)}(k+1)e_{q1}^{(j-1)}(k+1) + \lambda^{1/2}\sin\theta_j^{(0)}(k+1)\boldsymbol{d}_{q2}^{(N-j+1)}(k);$$

$$\boldsymbol{d}_{q2}^{(N-j+1)}(k+1) = \lambda^{1/2}\cos\theta_j^{(0)}(k+1)\boldsymbol{d}_{q2}^{(N-j+1)}(k) - \left[\sin\theta_j^{(0)}(k+1)\right]^* e_{q1}^{(j-1)}(k+1);$$

}

$$e(k+1) = \left[e_{q1}^{(N)}(k+1)\right]^* / \gamma_N^{(0)}(k+1);$$

} for k

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