A GENERAL APPROACH TO THE DERIVATION OF BLOCK MULTICHANNEL FAST QRD-RLS ALGORITHMS

António L. L. Ramos,[1] *José A. Apolinário Jr.,*[1] *and Stefan Werner*[2]

[1]Department of Electrical Engineering
Instituto Militar de Engenharia
Praça General Tibúrcio, 80
22290-270, Rio de Janeiro, Brazil
antonioluis@ime.eb.br, apolin@ieee.org

[2]Signal Processing Laboratory
Helsinki University of Technology
P. O. Box 3000
FI-02015 HUT, Finland
stefan.werner@hut.fi

ABSTRACT

Multichannel Fast QR Decomposition Recursive Least Squares (QRD-RLS) adaptive filtering algorithms have been mostly treated in the literature for channels of equal orders. However, in many applications, such as in the case of Volterra filtering, multichannel algorithms tailored for unequal orders are desirable. In this paper, a general formulation for deriving block versions of the Multichannel Fast QRD-RLS algorithms is introduced. The block type multichannel algorithms favor parallel processing implementations and also attain the reduced computational complexity and numerical robustness of the Fast QRD algorithms.

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]. Multichannel adaptive filtering algorithms can be derived using two distinct approaches: 1) a *block-type approach*, where the channels are processed simultaneously, and; 2) a *sequential approach* that processes each channel individually [2]. This paper considers algorithms derived from the former approach which is advantageous for parallel implementations.

The Fast QRD-RLS algorithms for the single-channel case can be classified according to which error vector (*a priori* or *a posteriori*) to update and the type of prediction (forward or backward), see Table 1. The classification of Table 1 can be extended to the multichannel case; the *a posteriori* and *a priori* versions based on backward error updating were proposed in [3, 4], and their order recursive (lattice) versions were introduced in [4, 5]. A unified framework for block Multichannel Fast QRD-RLS algorithms was addressed in [6], where two new block versions were proposed. In [4], a more general approach for the sequential processing, which copes with either equal or unequal channel orders, was introduced and the *a priori* (transversal and order recursive) versions of the multiple-order algorithm were presented; the corresponding *a posteriori* (transversal and order recursive) versions were introduced later in [7] and [8], respectively.

In the following, we present a general approach for deriving block-based multichannel multiple-order fast QRD-RLS algorithms. Adopting the same structure of the input signal vector from [4], two new block 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 to the algorithm in [2], however, derived from a input signal matrix with a different structure.

This work was partially funded by CNPq, CAPES, FAPERJ, and Academy of Finland, Smart and Novel Radios (SMARAD) Center of Excellence.

Table 1: The FQRD-RLS algorithm classification

Error type	Prediction: forward	Prediction: backward
a posteriori	FQRD_POS_F	FQRD_POS_B
a priori	FQRD_PRI_F	FQRD_PRI_B

2. BASIC EQUATIONS

The multichannel algorithms of the Fast QRD-RLS family use the weighted least-squares (LS) objective function defined as

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

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

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

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

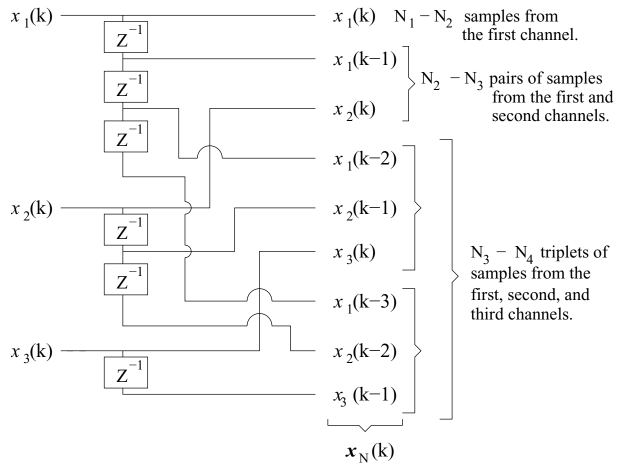

Figure 1: Obtaining the input vector.

and the optimal coefficient vector, $\boldsymbol{w}_N(k)$, is the one making the $MN \times 1$ vector $\boldsymbol{e}_{q2}(k)$ a null vector. To handle the multiple order-case, an alternative structure of the input signal vector will be defined in the next section.

3. DERIVING THE MULTIPLE-ORDER MULTICHANNEL FAST QRD-RLS ALGORITHMS

This section presents details of the input vector and the input data matrix needed in the derivation of the new algorithms.

3.1 Alternative definition of the input vector

Let M be the number of input channels and $N_1, N_2, \cdots, N_M$ the number of *taps* in the *tapped delay–lines* of each channel and, hereafter, $N = \sum_{r=1}^{M} N_r$ the overall number of *taps*. Without loss of generality, assume $N_1 \geq N_2 \geq \cdots \geq N_M$.

Fig. 1 shows an example of a multichannel scenario with 3 channels of unequal orders where $N_1 = 4$, $N_2 = 3$, $N_3 = 2$, i.e., $N = 4 + 3 + 2 = 9$. The following approach to construct the input vector, $\boldsymbol{x}_N(k)$, was considered in [4]: the first $N_1 - N_2$ *samples* from the first channel are chosen 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*–tuples of samples* of all channels. It is assumed that $N_{M+1} = 0$.

Using this definition, the expanded input vector, $\boldsymbol{x}_{N+M}(k+1)$, is given by

$$\boldsymbol{x}_{N+M}^T(k+1) = [x_1(k+1) \quad x_2(k+1) \quad \cdots \\ x_M(k+1) \quad \boldsymbol{x}_N^T(k)\Big] \boldsymbol{P} \tag{5}$$

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 to position $p_i = \sum_{r=1}^{i-1} r(N_r - N_{r+1}) + i$ (for $i = 1, 2, \cdots, M$) in vector $\boldsymbol{x}_{N+M}(k+1)$. After the above process is terminated, we have $\boldsymbol{x}_{N+M}^T(k+1) = [\boldsymbol{x}_N^T(k+1) \quad x_1(k-N_1+1) \cdots \quad x_M(k-N_M+1)]$, such that the first N elements of $\boldsymbol{x}_{N+M}^T(k+1)$ provide the input vector for the next iteration.

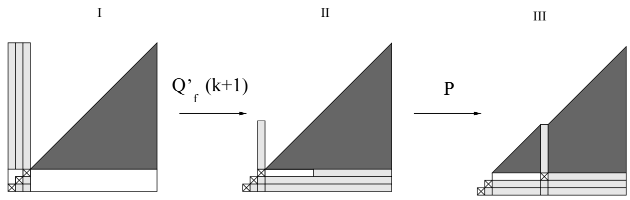

Figure 2: Obtaining the lower triangular $\boldsymbol{U}_{N+M}(k+1)$.

3.2 The input data matrix

The expanded input data matrix $\boldsymbol{X}_{N+M}(k+1)$ can be defined as

$$\begin{bmatrix} \boldsymbol{x}_{N+M}^T(k+1) \\ \lambda^{1/2}\boldsymbol{x}_{N+M}^T(k) \\ \vdots \\ \lambda^{k/2}\boldsymbol{x}_{N+M}^T(0) \\ \mathbf{0}_{(M-1)\times(N+M)} \end{bmatrix} = \left[\begin{array}{c|c} \boldsymbol{D}_f(k) & \begin{matrix}\boldsymbol{X}_N(k) \\ \mathbf{0}^T\end{matrix} \\ \hline \multicolumn{2}{c}{\mathbf{0}_{(M-1)\times(N+M)}} \end{array}\right] \tag{6}$$

For the triangularization of (6), three sets of Givens rotation matrices are needed [3, 4, 5], $\boldsymbol{Q}(k)$, $\boldsymbol{Q}_f(k+1)$, and $\boldsymbol{Q}'_f(k+1)$.

$$\boldsymbol{Q}'_f(k+1)\boldsymbol{Q}_f(k+1)\boldsymbol{Q}(k)\boldsymbol{X}_{N+M}(k+1) =$$

$$\boldsymbol{Q}'_f(k+1)\boldsymbol{Q}_f(k+1) \begin{bmatrix} \boldsymbol{E}_{fq1}(k+1) & \mathbf{0} \\ \boldsymbol{D}_{fq2}(k+1) & \boldsymbol{U}_N(k) \\ \lambda^{k/2}\boldsymbol{x}_0^T & \mathbf{0}^T \\ \multicolumn{2}{c}{\mathbf{0}_{(M-1)\times(N+M)}} \end{bmatrix} \mathbf{P} =$$

$$\boldsymbol{Q}'_f(k+1) \begin{bmatrix} \mathbf{0}^T & \mathbf{0}^T \\ \boldsymbol{D}_{fq2}(k+1) & \boldsymbol{U}_N(k) \\ \boldsymbol{E}_f(k+1) & \mathbf{0} \end{bmatrix} \boldsymbol{P} \tag{7}$$

In (7), $\boldsymbol{Q}(k)$ contains $\mathbf{Q}_N(k)$ as a partition and triangularizes $\boldsymbol{X}_N(k)$. $\boldsymbol{Q}_f(k+1)$ is responsible for the zeroing of matrix $\mathbf{E}_{fq1}(k+1)$; note that when working with a fixed order, it is equivalent to annihilating $\boldsymbol{e}_{fq1}^T(k+1)$, the first row of $\boldsymbol{E}_{fq1}(k+1)$, against the diagonal of $\lambda^{1/2}\boldsymbol{E}_f(k)$, generating $\boldsymbol{E}_f(k+1)$.

From (7) and using the fixed-order matrices $\boldsymbol{Q}_\theta(k)$ embedded in $\boldsymbol{Q}(k)$ and $\overline{\boldsymbol{Q}}_f(k+1)$ embedded in $\boldsymbol{Q}_f(k+1)$, after some algebraic work, one can write the following equations:

$$\begin{bmatrix} \boldsymbol{e}_{fq1}^T(k+1) \\ \boldsymbol{D}_{fq2}(k+1) \end{bmatrix} = \boldsymbol{Q}_\theta(k) \begin{bmatrix} \boldsymbol{x}_{k+1}^T \\ \lambda^{1/2}\boldsymbol{D}_{fq2}(k) \end{bmatrix} \tag{8}$$

$$\begin{bmatrix} \mathbf{0}^T \\ \boldsymbol{E}_f(k+1) \end{bmatrix} = \overline{\boldsymbol{Q}}_f(k+1) \begin{bmatrix} \boldsymbol{e}_{fq1}^T(k+1) \\ \lambda^{1/2}\boldsymbol{E}_f(k) \end{bmatrix} \tag{9}$$

In (8), $\boldsymbol{x}_{k+1}^T = [x_1(k+1) \quad x_2(k+1) \cdots x_M(k+1)]$ is the *forward reference signal* and $\mathbf{e}_{fq1}^T(k+1)$ is the *rotated forward error*; in (9), $\boldsymbol{E}_f(k+1)$ is the $M \times M$ *forward prediction error covariance matrix*.

However, the permutation matrix $\boldsymbol{P}$ in (7) 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 $\boldsymbol{Q}'_f(k+1) = \boldsymbol{Q}'^{(M)}_f(k+1) \cdots \boldsymbol{Q}'^{(2)}_f(k+1)\boldsymbol{Q}'^{(1)}_f(k+1)$.

It can be seen from (7) 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, one need to nullify only the first $N+i-p_i$ elements of each $\boldsymbol{d}_{fq2}^{(i)}(k+1)$. Note that when the position $p_i = i$, the corresponding permutation factor $\boldsymbol{P}_i$ will degenerate to an identity matrix. If this is true for all M channels, this formulation leads to the equal-order algorithms of [3, 4, 6, 5].

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 three of Fig. 2 shows the final result of the process in (7) for this particular case. The figure illustrates that the desired triangular shape was not reached so far. Hence, another permutation factor to up-shift the $(N+M-i+1)$th row to the $(N+M-p_i+1)$th position is needed. Removing the ever-increasing null section in (7) and using the fixed-order matrix $\boldsymbol{Q}'_{\theta f}(k+1)$ embedded in $\boldsymbol{Q}'_f(k+1)$, we get

$$\begin{aligned}\boldsymbol{U}_{N+M}(k+1) &= \\ &\overline{\boldsymbol{P}}\boldsymbol{Q}'_{\theta f}(k+1)\begin{bmatrix}\boldsymbol{D}_{fq2}(k+1) & \boldsymbol{U}_N(k)\\ \lambda^{1/2}\boldsymbol{E}_f(k) & \boldsymbol{0}\end{bmatrix}\boldsymbol{P}\end{aligned} \quad (10)$$

where the permutation matrix $\overline{\boldsymbol{P}} = \overline{\boldsymbol{P}}_1\overline{\boldsymbol{P}}_2\cdots\overline{\boldsymbol{P}}_M$, $\overline{\boldsymbol{P}}_i$ is responsible for up-shifting the $L+M-i+1$ row to the $L+M-p_i+1$ position. From (10) it is possible to obtain

$$\begin{aligned}[\boldsymbol{U}_{N+M}(k)]^{-1} &= \boldsymbol{P}^T\\ \times&\begin{bmatrix}\boldsymbol{0} & \boldsymbol{E}_f^{-1}(k+1)\\ \boldsymbol{U}_N^{-1}(k+1) & -\boldsymbol{U}_N^{-1}(k)\boldsymbol{D}_{fq2}(k+1)\boldsymbol{E}_f^{-1}(k+1)\end{bmatrix}\\ \times&\boldsymbol{Q}'^T_{\theta f}(k+1)\overline{\boldsymbol{P}}^T\end{aligned} \quad (11)$$

which will be used in the next section to derive the *a priori* and the *a posteriori* versions of the algorithm. Also from (10), we can write

$$\begin{bmatrix}\boldsymbol{0}\\ *\\ \boldsymbol{E}_f^0(k+1)\end{bmatrix} = \boldsymbol{Q}'_{\theta f}(k+1)\begin{bmatrix}\boldsymbol{D}_{fq2}(k+1)\\ \boldsymbol{E}_f(k+1)\end{bmatrix} \quad (12)$$

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

4. *A PRIORI* AND *A POSTERIORI* VERSIONS

The *a priori* and the *a posteriori* versions of this algorithm are based on updating vectors $\boldsymbol{a}_{N+M}(k+1)$ or $\boldsymbol{f}_{N+M}(k+1)$, respectively, also known as the *a priori* and the *a posteriori* backward error vectors [4, 3] where

$$\boldsymbol{a}_{N+M}(k+1) = \lambda^{-1/2}\boldsymbol{U}_{N+M}^{-T}(k)\boldsymbol{x}_{N+M}(k+1) \quad (13)$$

$$\boldsymbol{f}_{N+M}(k+1) = \boldsymbol{U}_{N+M}^{-T}(k+1)\boldsymbol{x}_{N+M}(k+1) \quad (14)$$

From (5), (11), and (13), we can write

$$\boldsymbol{a}_{N+M}(k+1) = \overline{\boldsymbol{P}}\lambda^{-1/2}\boldsymbol{Q}'_{\theta f}(k)\begin{bmatrix}\boldsymbol{a}_N(k)\\ \boldsymbol{r}(k+1)\end{bmatrix} \quad (15)$$

where

$$\begin{aligned}\boldsymbol{r}(k+1) &= \boldsymbol{E}_f^{-T}(k+1)\left[\boldsymbol{x}_{k+1} - \boldsymbol{W}_f^T(k+1)\boldsymbol{x}_N(k)\right]\\ &= \lambda^{-1/2}\boldsymbol{E}_f^{-T}(k+1)\boldsymbol{e}'_f(k+1)\end{aligned} \quad (16)$$

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

Likewise, combining Equations (5), (11), and (14) gives

$$\boldsymbol{f}_{N+M}(k+1) = \overline{\boldsymbol{P}}\boldsymbol{Q}'_{\theta f}(k+1)\begin{bmatrix}\boldsymbol{f}_N(k)\\ \boldsymbol{p}(k+1)\end{bmatrix} \quad (17)$$

where

$$\begin{aligned}\boldsymbol{p}(k+1) &= \boldsymbol{E}_f^{-T}(k+1)\left[\boldsymbol{x}_{k+1} - \boldsymbol{W}_f^T(k+1)\boldsymbol{x}_N(k)\right]\\ &= \boldsymbol{E}_f^{-T}(k+1)\boldsymbol{e}_f(k+1)\end{aligned} \quad (18)$$

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

The matrix inversion operations in (16) and (18) can be avoided using the solutions presented in [4] and [5] also applicable to the more general case considered in this work.

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

$$\boldsymbol{Q}_\theta(k+1)\begin{bmatrix}1\\ \boldsymbol{0}\end{bmatrix} = \begin{bmatrix}\gamma(k+1)\\ \boldsymbol{f}_N(k+1)\end{bmatrix} \quad (19)$$

Finally, the joint process estimation is performed as

$$\begin{bmatrix}\boldsymbol{e}_{q1}(k+1)\\ \boldsymbol{d}_{q2}(k+1)\end{bmatrix} = \boldsymbol{Q}_\theta(k+1)\begin{bmatrix}d(k+1)\\ \lambda^{1/2}\boldsymbol{d}_{q2}(k)\end{bmatrix} \quad (20)$$

and the *a priori error* is given by [3, 4]

$$\varepsilon(k+1) = e_{q1}(k+1)/\gamma(k+1) \quad (21)$$

The *a posteriori* and *a priori* algorithms are summarized in Tables 2 and 3. As for the computational complexity, the algorithm of Table 2 presents a slightly lower load among the block multichannel fast QRD-RLS algorithms. The number of multiplications, divisions, and squared roots can be seen in Table 4. Note that N corresponds to the length of each channel for the case of equal order and the sum of the length of each channel in the multiple order case. In the table, the computational complexities of four algorithms are shown, the first two corresponding to the proposed approach. Also note that the algorithm proposed in [2] is expected to have similar computational complexity as the *a posteriori* version derived here since they differ only in the structure of the input vector and the corresponding permutation matrices.

5. SIMULATION RESULTS

This section describes the application of the proposed algorithms in a second-order Volterra system identification. Space limitations prohibit minute details, see [2] for a complete simulation setup. The input signal to the unknown plant with 65 coeffcients was a zero-mean white Gaussian noise sequence (variance 0.0248) filtered through an FIR system given by $H(z) = 0.9045 + z^{-1} + 0.9045z^{-2}$. The forgetting factor was set to $\lambda = 0.995$ and the output signal-to-observation noise ratio was set to 30dB. The MSE ($E[e^2(k)]$, $e(k)$ being the *a priori* error) versus iterations is presented in Fig. 3. The curves for both versions considered in this paper are identical up to numerical accuracy, assuming equivalent initialization.

6. CONCLUSIONS

In this paper, a general formulation for block multiple-order multichannel fast QRD-RLS algorithms was introduced. The new formulation, based on the alternative definition of the input signal vector as found in [4], provides a block-type multichannel capable of processing all channels simultaneously and amenable for parallel implementations. Both *a posteriori* and *a priori* versions were derived, where the former is similar to the algorithm introduced in [2] and the latter is a new multiple-order block-type version of the sequential algorithm introduced in [4].

Table 4: Computational complexity of Block Multichannel Fast QRD-RLS algorithms.

ALGORITHM	MULTIPLICATIONS	DIVISIONS	SQUARED ROOTS
Algorithm of Table 2	$4NM^2+11NM+5M^2+6M+7N-(4M^2+6M)\sum_{i=1}^{M}(p_i-i)$	$2NM+2M+N-2M\sum_{i=1}^{M}(p_i-i)$	$2NM+M+N-2M\sum_{i=1}^{M}(p_i-i)$
Algorithm of Table 3	$4NM^2+11NM+5M^2+6M+9N-(4M^2+6M)\sum_{i=1}^{M}(p_i-i)$	$2NM+3M+2N-2M\sum_{i=1}^{M}(p_i-i)+2$	$2NM+M+N-2M\sum_{i=1}^{M}(p_i-i)$
Lattice *a posteriori* [5]	$4M^3N+17M^2N+12MN+5M^2+5M$	$2M^2N+3MN+2M$	$M^2N+2MN+M$
Lattice *a priori* [4]	$4M^3N+17M^2N+14MN+5M^2+6M$	$2M^2N+5MN+3M$	$M^2N+2MN+M$

Table 2: The MCFQRD_POS_B Equations.

For each k, do

{

1. Obtaining $\boldsymbol{D}_{fq2}(k+1)$ and $\boldsymbol{e}_{fq1}(k+1)$

$$\begin{bmatrix} \boldsymbol{e}_{fq1}^T(k+1) \\ \boldsymbol{D}_{fq2}(k+1) \end{bmatrix} = \boldsymbol{Q}_\theta(k) \begin{bmatrix} \boldsymbol{x}_{k+1}^T \\ \lambda^{1/2}\boldsymbol{D}_{fq2}(k) \end{bmatrix} \quad (8)$$

2. Obtaining $\boldsymbol{E}_f(k+1)$

$$\begin{bmatrix} \boldsymbol{0}^T \\ \boldsymbol{E}_f(k+1) \end{bmatrix} = \overline{\boldsymbol{Q}}_f(k+1) \begin{bmatrix} \boldsymbol{e}_{fq1}^T(k+1) \\ \lambda^{1/2}\boldsymbol{E}_f(k) \end{bmatrix} \quad (9)$$

3. Obtaining $\boldsymbol{p}(k+1)$

$$\begin{bmatrix} * \\ \boldsymbol{p}(k+1) \end{bmatrix} = \overline{\boldsymbol{Q}}_f(k+1) \begin{bmatrix} \gamma(k) \\ \boldsymbol{0} \end{bmatrix} \quad \text{implements } (18)$$

4. Obtaining $\boldsymbol{Q}'_{\theta f}(k+1)$

$$\begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{*} \\ \boldsymbol{E}_f^0(k+1) \end{bmatrix} = \boldsymbol{Q}'_{\theta f}(k+1) \begin{bmatrix} \boldsymbol{D}_{fq2}(k+1) \\ \boldsymbol{E}_f(k+1) \end{bmatrix} \quad (12)$$

5. Obtaining $\boldsymbol{f}_N(k+1)$

$$\boldsymbol{f}_{N+M}(k+1) = \overline{\boldsymbol{P}}\boldsymbol{Q}'_{\theta f}(k+1) \begin{bmatrix} \boldsymbol{f}_N(k) \\ \boldsymbol{p}(k+1) \end{bmatrix} \quad (17)$$

6. Obtaining $\boldsymbol{Q}_\theta(k+1)$ and $\gamma(k+1)$

$$\boldsymbol{Q}_\theta(k+1) \begin{bmatrix} 1 \\ \boldsymbol{0} \end{bmatrix} = \begin{bmatrix} \gamma(k+1) \\ \boldsymbol{f}_N(k+1) \end{bmatrix} \quad (19)$$

7. Joint Estimation

$$\begin{bmatrix} e_{q1}(k+1) \\ \boldsymbol{d}_{q2}(k+1) \end{bmatrix} = \boldsymbol{Q}_\theta(k+1) \begin{bmatrix} d(k+1) \\ \lambda^{1/2}\boldsymbol{d}_{q2}(k) \end{bmatrix} \quad (20)$$

8. Obtaining the *a priori* error

$$\varepsilon(k+1) = e_{q1}(k+1)/\gamma(k+1) \quad (21)$$

}

Table 3: The MCFQRD_PRI_B Equations.

For each k, do

{

1. Obtaining $\boldsymbol{D}_{fq2}(k+1)$ and $\boldsymbol{e}_{fq1}(k+1)$

$$\begin{bmatrix} \boldsymbol{e}_{fq1}^T(k+1) \\ \boldsymbol{D}_{fq2}(k+1) \end{bmatrix} = \boldsymbol{Q}_\theta(k) \begin{bmatrix} \boldsymbol{x}_{k+1}^T \\ \lambda^{1/2}\boldsymbol{D}_{fq2}(k) \end{bmatrix} \quad (8)$$

2. Obtaining $\boldsymbol{E}_f(k+1)$

$$\begin{bmatrix} \boldsymbol{0}^T \\ \boldsymbol{E}_f(k+1) \end{bmatrix} = \overline{\boldsymbol{Q}}_f(k+1) \begin{bmatrix} \boldsymbol{e}_{fq1}^T(k+1) \\ \lambda^{1/2}\boldsymbol{E}_f(k) \end{bmatrix} \quad (9)$$

3. Obtaining $\boldsymbol{r}(k+1)$

$$\begin{bmatrix} * \\ \boldsymbol{0} \end{bmatrix} = \overline{\boldsymbol{Q}}_f(k+1) \begin{bmatrix} 1/\gamma(k) \\ -\boldsymbol{r}(k+1) \end{bmatrix} \quad \text{implements } (16)$$

4. Obtaining $\boldsymbol{a}_N(k+1)$

$$\boldsymbol{a}_{N+M}(k+1) = \overline{\boldsymbol{P}}\boldsymbol{Q}'_{\theta f}(k+1) \begin{bmatrix} \boldsymbol{a}_N(k) \\ \boldsymbol{r}(k+1) \end{bmatrix} \quad (15)$$

5. Obtaining $\boldsymbol{Q}'_{\theta f}(k+1)$

$$\begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{*} \\ \boldsymbol{E}_f^0(k+1) \end{bmatrix} = \boldsymbol{Q}'_{\theta f}(k+1) \begin{bmatrix} \boldsymbol{D}_{fq2}(k+1) \\ \boldsymbol{E}_f(k+1) \end{bmatrix} \quad (12)$$

6. Obtaining $\boldsymbol{Q}_\theta(k+1)$ and $\gamma(k+1)$

$$\begin{bmatrix} 1/\gamma(k+1) \\ \boldsymbol{0} \end{bmatrix} = \boldsymbol{Q}_\theta(k+1) \begin{bmatrix} 1 \\ -\boldsymbol{a}_N(k+1) \end{bmatrix} \quad (19)$$

7. Joint Estimation

$$\begin{bmatrix} e_{q1}(k+1) \\ \boldsymbol{d}_{q2}(k+1) \end{bmatrix} = \boldsymbol{Q}_\theta(k+1) \begin{bmatrix} d(k+1) \\ \lambda^{1/2}\boldsymbol{d}_{q2}(k) \end{bmatrix} \quad (20)$$

8. Obtaining the *a priori* error

$$\varepsilon(k+1) = e_{q1}(k+1)/\gamma(k+1) \quad (21)$$

}

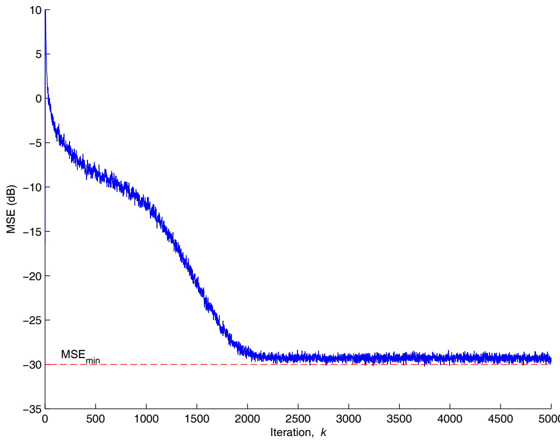

Figure 3: Learning curves (*a priori* error).

REFERENCES

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

[2] M. A. Syed and V. J. Mathews, "QR-Decomposition based algorithms for adaptive Volterra filtering," *IEEE Trans. Signal Processing*, vol. 40, pp. 372–382, June 1993.

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

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

[5] A. L. L. Ramos and J. A. Apolinário Jr., "A lattice version of the multichannel FQRD algorithm based on *a posteriori* backward errors," *Brazil, ICT'04, LNCS*, vol. 3124, pp. 488–497, Aug. 2004.

[6] 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, USA*, vol. 3, Aug. 2002.

[7] A. L. L. Ramos and J. A. Apolinário Jr., "A new multiple order multichannel fast QRD algorithm and its application to non-linear system identification," *SBT'04, Brazil*, Sept. 2004.

[8] A. L. L. Ramos, J. A. Apolinário Jr., and M. G. Siqueira, "A new order recursive multiple order multichannel fast QRD algorithm," *Asilomar'04, USA*, Nov. 2004.