Multichannel Fast QRD-LS Adaptive Filtering: New Technique and Algorithms

Athanasios A. Rontogiannis and Sergios Theodoridis

***Abstract*—In this paper, a direct, unified approach for deriving fast multichannel QR decompostion (QRD) least squares (LS) adaptive algorithms is introduced. The starting point of the new methodology is the efficient update of the Cholesky factor of the input data correlation matrix. Using the new technique, two novel fast multichannel algorithms are developed. Both algorithms comprise scalar operations only and are based exclusively on numerically robust orthogonal Givens rotations. The first algorithm assumes channels of equal orders and processes them all simultaneously. It is highly modular and provides enhanced pipelinability, with no increase in computational complexity, when compared with other algorithms of the same category. The second multichannel algorithm deals with the general case of channels with different number of delay elements and processes each channel separately. A modification of the algorithm leads to a scheme that can be implemented on a very regular systolic architecture. Moreover, both schemes offer substantially reduced computational complexity compared not only with the first algorithm but also with previously derived multichannel fast QRD schemes. Experimental results in two specific application setups as well as simulations in a finite precision environment are also included.**

***Index Terms*—Adaptive systems, multichannel LS algorithms, numerical stability, QR decomposition.**

I. Introduction

MULTICHANNEL least squares adaptive algorithms [1]–[3], [6]–[20], [27]–[29] are becoming increasingly popular due to their fast converging properties, and they find wide applications in diverse areas such as channel equalization, stereophonic echo cancellation, multidimentional signal processing, and Volterra-type nonlinear system identification, to name but a few. Among the various efficiency issues, characterizing the performance of an algorithm, those of computational complexity, parallelism, and numerical robustness are of particular importance, especially in applications where medium to long filter lengths are required. The general LS multichannel problem leads to adaptive algorithms of $O(k^2)$ computational complexity, where k is the sum of the channel orders. However, for the time series case, exploitation of the underlying shift-invariant property results in reduction of the computational complexity. Block-type multichannel schemes, which process all channels simultaneously, were the first to be derived. These algorithms encompass matrix operations, such as matrix inversions, and numerical problems associated with those operations are usually encountered. Multichannel algorithms involving scalar-only operations have recently become very popular. In these algorithms, a proper channel decomposition technique is performed, and each channel is processed separately. This results in further reduction of computational complexity and improvement of the numerical properties of the algorithms. The need for numerically robust schemes has also led to the development of a class of algorithms based on the QR decomposition of the input data matrix via the Givens rotations approach.

Following the development of single channel schemes [1], [2], [5], multichannel least squares lattice [6]–[14], [18]–[20], and transversal [14]–[17] algorithms were originally derived. These include block as well as scalar-type schemes and can handle channels with different number of associated parameters [9], [11], [14]–[20]. Transversal-type algorithms are of lower complexity and directly provide the channels' coefficients. Lattice algorithms, on the other hand, are highly modular and produce the LS estimation error, order recursively, in a pipelined fashion.

Another class of multichannel algorithms springs from the single channel fast QRD schemes [21]–[25], which are known to be numerically well behaved. Both the block- and the channel decomposition-based cases have been treated for channels of equal [27]–[28] or unequal [29] orders. Especially in [29], a novel channel decomposition technique is introduced, which makes possible the manipulation of channels of different orders. This channel decomposition procedure leads to a multichannel fast QRD algorithm consisting of l single channel fast QRD algorithms of length k, where l is the number of channels. These l single-channel algorithms are interdependent and are executed sequentially, one after the other. The resulting algorithm is of $O(kl)$ computational complexity.

In this paper, a novel, unified approach for deriving multichannel fast QRD algorithms is introduced. The new technique is based on the efficient time update of a particular vector quantity, which provides all the necessary for the LS error update rotation parameters. This vector quantity is basically the state vector of the equivalent state space description of the algorithmic process. It is directly related to the upper triangular R factor of the input data matrix in a QR factorizarion. In contrast, all previously derived QRD algorithms exploit quantities related to the Q factor. A direct consequence of the

Manuscript received February 28, 1996; revised November 11, 1997. The associate editor coordinating the review of this paper and approving it for publication was Prof. Tyseer Aboulnasr.

A. A. Rontogiannis is with the Greek Airforce.

S. Theodoridis is with the Department of Informatics, Division of Communications and Signal Processing, the University of Athens, Athens, Greece.

Publisher Item Identifier S 1053-587X(98)07797-6.

new technique is that explicit backward steps are essentially alleviated. This fact has a twofold advantage. First, derivations are simplified, and all existing fast QRD algorithms can be obtained in a simplified way. Second, it paves the way for the derivation of new more efficient algorithms. The proposed methodology is also direct and insightful in that all algorithmic quantities involved have an obvious LS meaning and interpretation.

Two new multichannel algorithms are presented in this paper. Both algorithms are based exclusively on numerically robust orthogonal Givens rotations. The first algorithm is a block-type scheme, which processes all channels jointly. The channel orders are assumed to be equal. In spite of its block nature, the new algorithm comprises scalar operations only, which in conjuction with the use of orthogonal Givens rotations guarantees the numerical robustness of the proposed scheme. In contrast with previously derived fast QRD algorithms of the same category [27], [28], the new algorithm is highly modular and pipelinable and generates the solutions of all lower order problems. The second algorithm deals with the general case of unequal channel lengths. The channel partitioning used in [29], in the context of Volterra filtering, is also adopted here, and the new algorithm consists of l single-channel algorithms of the type of [26], which are excecuted sequentially. A slight modification leads to an alternative form of the algorithm, which can be implemented on a circular systolic architecture where the single channel algorithms are executed in a pipelined fashion. In contrast, the channel decomposition-based algorithms of [27]–[29] are strictly sequential for each time iteration. Moreover, compared with [29], the new algorithms offer reduced computational complexity in terms of both multiplications/divisions and square roots. This fact becomes apparent from the methodology adopted and is demonstrated with a specific example that concerns Volterra-type nonlinear filtering.

The paper is organized as follows. An introductory framework for general LS adaptive filtering schemes is described in Section II. The new multichannel fast QRD algorithms are then presented in Sections III and IV. Experimental results of the use of the channel decomposition-based algorithm in two specific applications are provided in Section V. Some numerical simulations are also included. Section VI concludes this work. For clarity of presentation, real-valued signals are considered throughout the paper.

II. The General Least Squares Problem

The standard exponentially weighted least squares (LS) problem is that of selecting a $k\times m$ coefficients' matrix $C(N)$ to satisfy the optimization scheme

$$\min_{C(N)} \sum_{n=1}^{N} \lambda^{N-n}[\boldsymbol{y}(n)-C^T(N)\boldsymbol{u}(n)]^T[\boldsymbol{y}(n)-C^T(N)\boldsymbol{u}(n)] \tag{1}$$

where

- λ usual forgetting factor with $0 \ll \lambda \leq 1$;
- $\boldsymbol{u}(n)$ $k \times 1$ input data vector;
- $\boldsymbol{y}(n)$ $m \times 1$ desired response vector at time $n, n = 1,2,\cdots,N$.

The input–output information can be used to form the data matrix

$$U^+(N) = [Y(N)|U(N)] = \Lambda(N)\begin{bmatrix} \boldsymbol{y}^T(1) & \boldsymbol{u}^T(1) \\ \boldsymbol{y}^T(2) & \boldsymbol{u}^T(2) \\ \vdots & \vdots \\ \boldsymbol{y}^T(N) & \boldsymbol{u}^T(N) \end{bmatrix} \tag{2}$$

where $\Lambda(N) = \text{diag}\,[\lambda^{N-1/2} \quad \lambda^{N-2/2} \quad \cdots \quad 1]$.

According to the projection theorem [1], the solution of the LS problem requires the minimization of the Frobenius norm [4] of the error matrix $E(N)$, which is achieved after the orthogonal projection $\hat{Y}(N)$ of the column space of $Y(N)$ onto the column space of $U(N)$. $C(N)$ then contains the coefficients of this projection. This is compactly written as

$$\begin{aligned} E(N) &= Y(N) - \hat{Y}(N) = Y(N) - U(N)C(N) \\ &= U^+(N)\begin{bmatrix} I \\ -C(N) \end{bmatrix}. \end{aligned} \tag{3}$$

If now $Q(N)$ stands for the orthogonal matrix that converts $U(N)$ into the $k \times k$ upper triangular form $\tilde{R}(N)$, then

$$Q(N)U^+(N) = \begin{bmatrix} P(N) & \tilde{R}(N) \\ V(N) & \bigcirc \end{bmatrix} \tag{4}$$

where $P(N) \in \mathcal{R}^{k\times m}$, and $V(N) \in \mathcal{R}^{(N-k)\times m}$. Since multiplication with an orthogonal matrix is norm preserving, it is straightforward from (3) and (4) that $C(N)$ is given by

$$\tilde{R}(N)C(N) = P(N). \tag{5}$$

In a time-varying environment, the time update of $\tilde{R}(N)$ and $P(N)$ is required, as new information becomes available. It turns out that all necessary quantities for the update of these matrices can be obtained from the manipulation of a single vector term. Specifically, let us define the vector

$$\boldsymbol{g}(N+1) = \frac{\tilde{R}^{-T}(N)\boldsymbol{u}(N+1)}{\sqrt{\lambda}} \tag{6}$$

and assume that

$$\hat{Q}(N+1)\begin{bmatrix} -\boldsymbol{g}(N+1) \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{0} \\ \delta(N+1) \end{bmatrix} \tag{7}$$

is satisfied. $\hat{Q}(N+1)$ is a sequence of k elementary Givens rotations [4], [30] that successively annul the elements of $-\boldsymbol{g}(N+1)$ by rotating them against the last element (initially 1), starting from the first element of $-\boldsymbol{g}(N+1)$ and moving downwards. It has been shown [1], [30], [31] that this $\hat{Q}(N+1)$ also updates $\tilde{R}(N)$ and $P(N)$ of (5) according to

$$\begin{aligned} &\hat{Q}(N+1)\begin{bmatrix} \lambda^{1/2}\tilde{R}(N) & \lambda^{1/2}P(N) \\ \boldsymbol{u}^T(N+1) & \boldsymbol{y}^T(N+1) \end{bmatrix} \\ &\quad = \begin{bmatrix} \tilde{R}(N+1) & P(N+1) \\ \boldsymbol{0}^T & \tilde{\boldsymbol{e}}^T(N+1) \end{bmatrix}. \end{aligned} \tag{8}$$

From (6), (7), and the orthogonality of $\hat{Q}(N+1)$, it can be shown that $\delta(N+1)$ equals the inverse of the square root of the angle variable [3]. Therefore, the angle normalized error vector $\tilde{\boldsymbol{e}}(N+1)$ will be related to the *a priori* error vector $\boldsymbol{e}(N+1)$ as [1], [26]

$$\boldsymbol{e}(N+1) = \delta(N+1)\tilde{\boldsymbol{e}}(N+1). \tag{9}$$

The efficient update of $\boldsymbol{e}(N+1)$ is at the heart of our problem. It is defined as [1]

$$\boldsymbol{e}(N+1) = \boldsymbol{y}(N+1) - C^T(N)\boldsymbol{u}(N+1). \tag{10}$$

Finally, note that if $A_y(N)$ stands for the error covariance matrix, it will be given by

$$\begin{aligned} A_y(N) &= E^T(N)E(N) = E^T(N)Q^T(N)Q(N)E(N) \\ &= V^T(N)V(N). \end{aligned} \tag{11}$$

The update of $A_y(N)$ is possible by employing [3]

$$A_y(N+1) = \lambda A_y(N) + \tilde{\boldsymbol{e}}(N+1)\tilde{\boldsymbol{e}}^T(N+1). \tag{12}$$

In the following sections, two new multichannel LS fast adaptive algorithms based on Givens rotations are presented. It is shown that each step of these algorithms can be treated as an LS problem of the type described in this section. Such an approach unifies the derivation and provides a clear LS interpretation of all algorithmic quantities involved.

III. Block Multichannel Fast QRD Algorithm

Let us consider a system consisting of l input channels, each of length p ($k = lp$), and an output channel. The more general case of m output channels $m > 1$ essentially corresponds to m single-channel output problems and thus will not be further treated. Using input information up to time N, we can form the $N \times lp$ matrix[1]

$$\begin{aligned} U_p(N) &= \Lambda(N)\begin{bmatrix} \boldsymbol{u}_p^T(1) \\ \boldsymbol{u}_p^T(2) \\ \vdots \\ \boldsymbol{u}_p^T(N) \end{bmatrix} \\ &= \Lambda(N)\begin{bmatrix} \boldsymbol{u}_1^T & \boldsymbol{0}^T & \cdots & \boldsymbol{0}^T \\ \boldsymbol{u}_2^T & \boldsymbol{u}_1^T & \cdots & \boldsymbol{0}^T \\ \vdots & \vdots & & \vdots \\ \boldsymbol{u}_N^T & \boldsymbol{u}_{N-1}^T & \cdots & \boldsymbol{u}_{N-p+1}^T \end{bmatrix} \end{aligned} \tag{13}$$

where vector $\boldsymbol{u}_n^T$ contains the l inputs to the system at time $n, \boldsymbol{u}_n^T = [u_1(n) \quad u_2(n) \quad \cdots \quad u_l(n)]$. Note from (13) that the prewindowed assumption is adopted for each channel. Using the input data matrix given in (13) and the vector of the scalar desired responses (up to time N), we can form a LS problem similar to that presented in the previous section. As it has already been explained, the vector quantity of interest is

$$\boldsymbol{g}_p(N) = \frac{\tilde{R}_p^{-T}(N-1)\boldsymbol{u}_p(N)}{\sqrt{\lambda}} \tag{14}$$

[1]The subscripts $p, p+1, i$ or $i-1$ used in this section mean that the respective quantities correspond to the pth, $(p+1)$th, ith, or $(i-1)$th-order problem in which all channels have $p, p+1, i,$ or $i-1$ delay elements, respectively. Any exception to this rule will be explicitly stated. The dimensions of the respective quantities will be also explicitly given or will be easily inferred from the text.

where $\tilde{R}_p(N-1)$ is the $lp \times lp$ Cholesky factor of $U_p(N-1)$. It is clear from (14) that the time update of $\boldsymbol{g}_p(N)$ requires the time update of $\tilde{R}_p^{-1}(N-1)$. The latter is achieved by employing (for a proof see Appendix A)

$$\tilde{R}_{p+1}^{-1}(N) = \begin{bmatrix} \tilde{R}_p^{-1}(N) & -\tilde{R}_p^{-1}(N)P_p^b(N)(\tilde{A}_p^b(N))^{-1} \\ \bigcirc & (\tilde{A}_p^b(N))^{-1} \end{bmatrix} \tag{15}$$

and (16), shown at the bottom of the page, where $P_p^b(N), \tilde{A}_p^b(N)$, and $P_p^f(N), \tilde{A}_p^f(N)$ are quantities related to the backward and forward problems, respectively (Appendix A). According to (56) of Appendix A, the orthogonal matrix $\hat{Q}_p^f(N)$ satisfies the equation

$$\hat{Q}_p^f(N)\begin{bmatrix} \tilde{A}_p^f(N) \\ P_p^f(N) \end{bmatrix} = \begin{bmatrix} \tilde{A}_0^f(N) \\ \bigcirc \end{bmatrix}. \tag{17}$$

Matrix $P_p^f(N)$ can be split up into $pl \times l$ blocks as

$$P_p^f(N) = \begin{bmatrix} P_p^{f(1)}(N) \\ \vdots \\ P_p^{f(p)}(N) \end{bmatrix}.$$

It is clear from the discussion in Appendix A that if $\hat{Q}_p^{f(i)}(N)$ is the "part" of $\hat{Q}_p^f(N)$ that annihilates $P_p^{f(i)}(N), i = p, p-1, \cdots, 1$, then $\hat{Q}_p^{f(i)}(N)$ consists of l^2 elementary Givens rotation matrices and satisfies

$$\hat{Q}_p^{f(i)}(N)\begin{bmatrix} \tilde{A}_i^f(N) \\ \bigcirc_{l(i-1)\times l} \\ P_p^{f(i)}(N) \\ \bigcirc_{l(p-i-1)\times l} \end{bmatrix} = \begin{bmatrix} \tilde{A}_{i-1}^f(N) \\ \bigcirc_{l(i-1)\times l} \\ \bigcirc_{l(p-i)\times l} \end{bmatrix}. \tag{18}$$

Furthermore, it is straightforward that $\hat{Q}_p^f(N) = \hat{Q}_p^{f(1)}(N)\hat{Q}_p^{f(2)}(N)\cdots\hat{Q}_p^{f(p)}(N)$. Combining, (14), (15), and the input vector partition $\boldsymbol{u}_{p+1}^T(N+1) = [\boldsymbol{u}_p^T(N+1) \quad \boldsymbol{u}_{N-p+1}^T]$, we get

$$\boldsymbol{g}_{p+1}(N+1) = \begin{bmatrix} \boldsymbol{g}_p(N+1) \\ \boldsymbol{g}^{(p)}(N+1) \end{bmatrix} \tag{19}$$

where the vector $\boldsymbol{g}^{(p)}(N+1)$ corresponds to the last l elements of $\boldsymbol{g}_{p+1}(N+1)$ and is expressed as

$$\begin{aligned} &\boldsymbol{g}^{(p)}(N+1) \\ &\quad = \frac{(\tilde{A}_p^b(N))^{-T}}{\sqrt{\lambda}}[\boldsymbol{u}_{N-p+1} - (\tilde{R}_p^{-1}(N)P_p^b(N))^T\boldsymbol{u}_p(N+1)] \\ &\quad = \frac{(\tilde{A}_p^b(N))^{-T}\boldsymbol{e}_p^b(N+1)}{\sqrt{\lambda}}. \end{aligned} \tag{20}$$

Clearly, $\boldsymbol{e}_p^b(N+1)$ is the pth-order $l \times 1$ *a priori* backward error vector, and consequently, $\boldsymbol{g}^{(p)}(N+1)$ can be interpreted as the pth-order normalized *a priori* backward error vector. In addition, the nesting of $\boldsymbol{g}$ vectors in (19) indicates that the $(i+1)$th l-element block of $\boldsymbol{g}_{p+1}(N+1), \boldsymbol{g}^{(i)}(N+1)$

$$\tilde{R}_{p+1}^{-1}(N) = \begin{bmatrix} (\tilde{A}_p^f(N))^{-1} & \bigcirc \\ -\tilde{R}_p^{-1}(N-1)P_p^f(N)(\tilde{A}_p^f(N))^{-1} & \tilde{R}_p^{-1}(N-1) \end{bmatrix}(\hat{Q}_p^f(N))^T \tag{16}$$

corresponds to the ith-order normalized *a priori* backward error vector for $i = 0, 1, \cdots, p$. From (14), (16), and the input vector partition $\boldsymbol{u}_{p+1}^T(N+1) = [\boldsymbol{u}_{N+1}^T \quad \boldsymbol{u}_p^T(N)]$, we also obtain

$$\boldsymbol{g}_{p+1}(N+1) = \hat{Q}_p^f(N)\begin{bmatrix}\boldsymbol{r}_p(N+1)\\ \boldsymbol{g}_p(N)\end{bmatrix} \tag{21}$$

where

$$\begin{aligned}\boldsymbol{r}_p(N+1) &= \frac{(\tilde{A}_p^f(N))^{-T}}{\sqrt{\lambda}}[\boldsymbol{u}_{N+1} - (\tilde{R}_p^{-1}(N-1)P_p^f(N))^T\boldsymbol{u}_p(N)]\\ &= \frac{(\tilde{A}_p^f(N))^{-T}\boldsymbol{e}_p^f(N+1)}{\sqrt{\lambda}}\end{aligned} \tag{22}$$

is the pth-order $l\times 1$ *a priori* normalized forward error vector. If now, in (21), we take into consideration the order-recursive form of $\hat{Q}_p^f(N)$ and the nesting property of $\boldsymbol{g}_p(N), \boldsymbol{g}_{p+1}(N+1)$, then it can easily be deduced that matrix $\hat{Q}_p^{f(i)}(N)$ has the following effect:

$$\hat{Q}_p^{f(i)}(N)\begin{bmatrix}\boldsymbol{r}_i(N+1)\\ \boldsymbol{0}_{l(i-1)}\\ \boldsymbol{g}^{(i-1)}(N)\\ \boldsymbol{0}_{l(p-i-1)}\end{bmatrix} = \begin{bmatrix}\boldsymbol{r}_{i-1}(N+1)\\ \boldsymbol{0}_{l(i-1)}\\ \boldsymbol{g}^{(i)}(N+1)\\ \boldsymbol{0}_{l(p-i-1)}\end{bmatrix} \quad i = 1, 2, \cdots, p \tag{23}$$

where $\boldsymbol{r}_i(N+1)$ is the ith order normalized *a priori* forward error vector. Equations (21) and (19) provide the time update of $\boldsymbol{g}_p(N)$ through the application of a sequence of orthogonal Givens rotations $\hat{Q}_p^f(N)$. The time update of $\hat{Q}_p^f(N)$ is also possible if (17) [or equivalently (18)] is written at time $N+1$. As a consequence, expressions providing $P_p^f(N+1)$ and $\tilde{A}_i^f(N+1)$ for $i = 1, 2, \cdots, p$ are necessary. $P_p^f(N+1)$ can be obtained by employing a second set of orthogonal Givens rotations. Specifically, this set of rotations corresponds to $\hat{Q}_p(N)$, which originates from the annuling of $-\boldsymbol{g}_p(N)$ with respect to 1 [see (7)] and is also used in the time update of the scalar *a priori* error $(e_p(N+1))$. Indeed, if we identify the general problem presented in the previous section with the forward prediction problem considered in Appendix A, we conclude [see (4), (8), and (54)] that $\hat{Q}_p(N)$ updates $P_p^f(N)$ according to

$$\hat{Q}_p(N)\begin{bmatrix}\lambda^{1/2}P_p^f(N)\\ \boldsymbol{u}_{N+1}^T\end{bmatrix} = \begin{bmatrix}P_p^f(N+1)\\ (\tilde{\boldsymbol{e}}_p^f(N+1))^T\end{bmatrix}. \tag{24}$$

$\tilde{\boldsymbol{e}}_p^f(N+1)$ stands for the angle normalized forward error vector. Since the upper $li\times 1$ block of $\boldsymbol{g}_p(N)$ coincides with $\boldsymbol{g}_i(N), i = 1, 2, \cdots, p$, the first li rotations of $\hat{Q}_p(N)$ will be the rotations of the ith-order problem. This, in conjuction with the fact that the upper $li\times l$ block of $P_p^f(N)$ is equal to $P_i^f(N)$ (Appendix A), shows that the angle normalized error vectors $\tilde{\boldsymbol{e}}_i^f(N+1), i = 1, 2, \cdots, p$ are successively computed in (24). These error vectors are related to the *a priori* forward error vectors $\boldsymbol{e}_i^f(N+1)$ as

$$\boldsymbol{e}_i^f(N+1) = \delta_i(N)\tilde{\boldsymbol{e}}_i^f(N+1) \qquad i = 1, 2, \cdots, p \tag{25}$$

where $\delta_i(N)$ is obtained after nullifying $-\boldsymbol{g}_i(N)$.

The presentation of the new algorithm is completed with the derivation of an expression for $\tilde{A}_i^f(N+1), i = 1, 2, \cdots, p$. Specifically, if we write (12) for the ith-order forward prediction problem, we obtain

$$A_i^f(N+1) = \lambda A_i^f(N) + \tilde{\boldsymbol{e}}_i^f(N+1)(\tilde{\boldsymbol{e}}_i^f(N+1))^T$$

or

$$\begin{aligned}(\tilde{A}_i^f(N+1))^T\tilde{A}_i^f(N+1) &= (\sqrt{\lambda}\tilde{A}_i^f(N))^T\sqrt{\lambda}\tilde{A}_i^f(N) + \tilde{\boldsymbol{e}}_i^f(N+1)\cdot\\ &\quad\cdot(\tilde{\boldsymbol{e}}_i^f(N+1))^T\end{aligned}$$

From the last equation, the Cholesky factor $\tilde{A}_i^f(N+1)$ can be calculated as

$$\tilde{Q}_i(N+1)\begin{bmatrix}\lambda^{1/2}\tilde{A}_i^f(N)\\ (\tilde{\boldsymbol{e}}_i^f(N+1))^T\end{bmatrix} = \begin{bmatrix}\tilde{A}_i^f(N+1)\\ \boldsymbol{0}^T\end{bmatrix}. \tag{26}$$

The Givens rotations matrix $\tilde{Q}_i(N+1)$ successively annihilates the l elements of $\tilde{\boldsymbol{e}}_i^f(N+1)$ against the diagonal elements of $\lambda^{1/2}\tilde{A}_i^f(N)$, retaining the positive definiteness of the resulting factor.

The algorithm described so far is shown in Table I. The quantity $\hat{Q}_p^{(i)}(N)$ is the part of $\hat{Q}_p(N)$ that zeroes $-\boldsymbol{g}^{(i-1)}(N)$ with respect to $\delta_{i-1}(N)$. $\boldsymbol{p}_p^{(i)}(N)$ stands for the ith $l\times 1$ part of the $lp\times 1$ vector $\boldsymbol{p}_p(N)$ (corresponding to $P(N)$ of (4) for the single output, equal channel orders case). Note that the use of equations of the type of (22) for the calculation of $\boldsymbol{r}_i$'s have been avoided. Such expressions involve matrix inversions and could be a source of numerical instabilities. Equation (22) has been bypassed by observing that $\tilde{Q}_i(N+1)$, which updates $\tilde{A}_i^f(N)$, also zeroes $-(\boldsymbol{r}_i(N+1)/\delta_i(N))$ against 1. Indeed, it suffices to notice the duality between $\tilde{Q}_i(N+1), \tilde{A}_i^f(N), (\boldsymbol{r}_i(N+1)/\delta_i(N))$ and $\hat{Q}_p(N+1), \tilde{R}_p(N), \boldsymbol{g}_p(N+1)$. Therefore, the rotation parameters of $\tilde{Q}_i(N+1)$ obtained from (26) can be also used for the calculation of the elements of $\boldsymbol{r}_i(N+1)$. As depicted in Table I, the new algorithm is based exclusively on orthogonal Givens rotations. It also involves scalar-only operations, although it treats all channels simultaneously. Therefore, it is expected that the new scheme will exhibit very nice numerical features and will be implementable on CORDIC-based architectures.

The proposed algorithm is of $O(pl^3)$ computational complexity, which is similar to the complexity of the other block fast QRD multichannel schemes [27], [28]. The main computational load comes from step 4, which involves $O(pl^3)$ operations. The remaining steps of the algorithm require $O(pl^2)$ operations. The distinct advantage of the new algorithm, however, lies on its modularity and pipelinability. It is clear from Table I that the new algorithm is pipelinable at the order level, that is, the throughput offered is constant, regardless of the channels' length. Furthermore, the new scheme can be implemented as the interconnection of p independent, identical modules and simultaneously provides the solutions of all lower order problems. The above are not characteristics

TABLE I
NEW BLOCK MULTICHANNEL QRD ALGORITHM

$\tilde{\mathbf{e}}_0^f(N+1) = \mathbf{u}_{N+1}$;

Update of $\tilde{A}_0^f(N)$ and calculation of $\mathbf{r}_0(N+1)$.

$$\bar{Q}_0(N+1)\begin{bmatrix} \lambda^{1/2}\tilde{A}_0^f(N) & -\mathbf{r}_0(N+1) \\ (\tilde{\mathbf{e}}_0^f(N+1))^T & \delta_0(N) \end{bmatrix} = \begin{bmatrix} \tilde{A}_0^f(N+1) & \mathbf{0} \\ \mathbf{0}^T & * \end{bmatrix};$$

$\mathbf{g}^{(0)}(N+1) = \mathbf{r}_0(N+1)$; $\delta_0(N+1) = 1$; $\tilde{e}_0(N+1) = y(N+1)$;

for $i = 1 : p$

1. Update of $P_p^{f(i)}(N)$ and calculation of $\tilde{\mathbf{e}}_i^f(N+1)$.

$$\hat{Q}_p^{(i)}(N)\begin{bmatrix} \lambda^{1/2}P_p^{f(i)}(N) \\ (\tilde{\mathbf{e}}_{i-1}^f(N+1))^T \end{bmatrix} = \begin{bmatrix} P_p^{f(i)}(N+1) \\ (\tilde{\mathbf{e}}_i^f(N+1))^T \end{bmatrix};$$

2. Update of $\tilde{A}_i^f(N)$ and calculation of $\mathbf{r}_i(N+1)$.

$$\bar{Q}_i(N+1)\begin{bmatrix} \lambda^{1/2}\tilde{A}_i^f(N) & -\mathbf{r}_i(N+1) \\ (\tilde{\mathbf{e}}_i^f(N+1))^T & \delta_i(N) \end{bmatrix} = \begin{bmatrix} \tilde{A}_i^f(N+1) & \mathbf{0} \\ \mathbf{0}^T & * \end{bmatrix};$$

3. Calculation of $\mathbf{g}^{(i)}(N+1)$.

$$\hat{Q}_p^{f(i)}(N)\begin{bmatrix} \mathbf{r}_i(N+1) \\ \mathbf{0}_{l(i-1)} \\ \mathbf{g}^{(i-1)}(N) \\ \mathbf{0}_{l(p-i-1)} \end{bmatrix} = \begin{bmatrix} \mathbf{r}_{i-1}(N+1) \\ \mathbf{0}_{l(i-1)} \\ \mathbf{g}^{(i)}(N+1) \\ \mathbf{0}_{l(p-i-1)} \end{bmatrix};$$

4. Calculation of the rotations of $\hat{Q}_p^{f(i)}(N+1)$.

$$\hat{Q}_p^{f(i)}(N+1)\begin{bmatrix} \tilde{A}_i^f(N+1) \\ \bigcirc_{l(i-1)\times l} \\ P_p^{f(i)}(N+1) \\ \bigcirc_{l(p-i-1)\times l} \end{bmatrix} = \begin{bmatrix} \tilde{A}_{i-1}^f(N+1) \\ \bigcirc_{l(i-1)\times l} \\ \bigcirc_{l(p-i)\times l} \end{bmatrix};$$

5. Calculation of the rotations of $\hat{Q}_p^{(i)}(N+1)$ and $\delta_i(N+1)$.

$$\hat{Q}_p^{(i)}(N+1)\begin{bmatrix} -\mathbf{g}^{(i-1)}(N+1) \\ \delta_{i-1}(N+1) \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \delta_i(N+1) \end{bmatrix};$$

6. Update of $\mathbf{p}_p^{(i)}(N)$ and calculation of $\tilde{e}_i(N+1)$ (filtering part).

$$\hat{Q}_p^{(i)}(N+1)\begin{bmatrix} \lambda^{1/2}\mathbf{p}_p^{(i)}(N) \\ \tilde{e}_{i-1}(N+1) \end{bmatrix} = \begin{bmatrix} \mathbf{p}_p^{(i)}(N+1) \\ \tilde{e}_i(N+1) \end{bmatrix};$$

end;

$e_p(N+1) = \delta_p(N+1)\tilde{e}_p(N+1)$;

Initialization

$\mathbf{g}_p(0) = \mathbf{0}$, $\mathbf{p}_p(0) = \mathbf{0}$, $P_p^f(0) = \bigcirc$

$\delta_i(0) = 1$, $\tilde{A}_i(0) = \mu I$, $i = 1, 2, \ldots, p$ (μ : small constant)

$\hat{Q}_p^{(i)}(0) = I$, $\hat{Q}_p^{f(i)}(0) = I$, $i = 1, 2, \ldots, p$

of the corresponding block fast QRD multichannel algorithms of [27], [28], which are sequential for each time iteration.

IV. CHANNEL DECOMPOSITION-BASED MULTICHANNEL FAST QRD ALGORITHM

The algorithm of the previous section assumes channels of equal orders and processes them all simultaneously. Such an assumption, however, may be sometimes restrictive in practice. In this section, we deal with the general case of channels with different orders, and we present a new fast QR scheme that treats each channel individually. The proposed algorithm is of lower computational complexity compared not only with the algorithm of the previous section (for equal channel lengths) but also with the other channel decomposition-based fast QRD algorithms of the same categoty.

Let us consider l input channels of lengths $k_1, k_2, \cdots, k_l$, respectively, and $k = \Sigma_{r=1}^{l} k_r$. Without loss of generality, we assume that $k_1 \geq k_2 \geq \cdots \geq k_l$. A critical point of our methodology is the selection of an appropriate partitioning of the input samples that appear in the input data vector $\boldsymbol{u}_k(N)$. Specifically, we choose the $k_1 - k_2$ most recent samples of the first channel to be the leading elements of $\boldsymbol{u}_k(N)$ followed by $k_2 - k_3$ pairs of samples of the first and second channel, followed by $k_3 - k_4$ triples of samples of the first three channels $\cdots$ followed by k_l l-ples of samples of all channels. For instance, in the three-channel case with $k_1 = 6, k_2 = 3$, and $k_3 = 2$, $\boldsymbol{u}_k(N)$ will have the form shown at the bottom of the page.

It is now straightforward that the position of the first (most recent) sample of the ith channel is given by

$$m_i = \sum_{r=1}^{i-1} r(k_r - k_{r+1}) + i \qquad i = 1, 2, \cdots, l.$$

Starting from $\boldsymbol{u}_k(N)$, we define the input data vectors $\boldsymbol{u}_{k+1}^T(N+1) = [u_1(N+1) \quad \boldsymbol{u}_k^T(N)]$ and $\boldsymbol{u}_{k+i}^T(N+1) = [u_i(N+1) \quad \boldsymbol{u}_{k+i-1}^T(N+1)]S_i$ for $i = 2, 3, \cdots, l$. S_i is a permutation matrix that moves $u_i(N+1)$ to the m_ith position after left shifting the first $m_i - 1$ elements of $\boldsymbol{u}_{k+i-1}^T(N+1)$. It can be easily verified that $\boldsymbol{u}_{k+l}^T(N+1) = [\boldsymbol{u}_k^T(N+1) \quad u_1(N-k_1+1) \quad \cdots \quad u_l(N-k_l+1)]$, that is, the first k elements of $\boldsymbol{u}_{k+l}^T(N+1)$ provide the input vector of the next time instant. The following input data matrices can now be defined:

$$U_{k+i}(N) = \Lambda(N)\begin{bmatrix} \boldsymbol{u}_{k+i}^T(1) \\ \boldsymbol{u}_{k+i}^T(2) \\ \vdots \\ \boldsymbol{u}_{k+i}^T(N) \end{bmatrix} \qquad i = 0, 1, \cdots, l. \quad (27)$$

$$\boldsymbol{u}_k^T(N) = [\underbrace{u_1(N) \quad u_1(N-1) \quad u_1(N-2)}_{k_1-k_2}\ \underbrace{u_1(N-3) \quad u_2(N)}_{2(k_2-k_3)}$$
$$\cdot \underbrace{u_1(N-4) \quad u_2(N-1) \quad u_3(N) \quad u_1(N-5) \quad u_2(N-2) \quad u_3(N-1)}_{3k_3}]$$

If $\tilde{R}_{k+i}(N)$ stands for the Cholesky factor of $U_{k+i}(N)$, the corresponding vectors $\boldsymbol{g}_{k+i}(N+1)$ can be expressed as

$$\boldsymbol{g}_{k+i}(N+1) = \frac{\tilde{R}_{k+i}^{-T}(N)\boldsymbol{u}_{k+i}(N+1)}{\sqrt{\lambda}} \qquad i = 0, 1, \cdots, l. \tag{28}$$

The $\boldsymbol{g}_k(N)$ vector will also be the critical quantity here. From the discussion above and (28), it is not difficult to show that

$$\boldsymbol{g}_{k+l}(N+1) = \begin{bmatrix} \boldsymbol{g}_k(N+1) \\ \boldsymbol{g}^{(k)}(N+1) \end{bmatrix} \tag{29}$$

where $\boldsymbol{g}^{(k)}(N+1)$ consists of the last l elements of $\boldsymbol{g}_{k+l}(N+1)$. The time update of $\boldsymbol{g}_k(N)$ can now be realized according to

$$\boldsymbol{g}_k(N) \rightarrow \boldsymbol{g}_{k+1}(N+1) \rightarrow \boldsymbol{g}_{k+2}(N+1) \rightarrow \cdots \rightarrow \boldsymbol{g}_{k+l}(N+1).$$

This procedure involves l "forward" steps, which will be described below. It is clear that because of (29), explicit backward steps are essentially avoided, and thus, our methodology only requires forward steps.

A. Forward Step 1

From (27), the input data matrix $U_{k+1}(N)$ can be written as

$$U_{k+1}(N) = \begin{bmatrix} \lambda^{N-1/2}u_1(1) & \boldsymbol{0}^T \\ \lambda^{N-2/2}u_1(2) & \\ \vdots & U_k(N-1). \\ u_1(N) & \end{bmatrix} \tag{30}$$

The last expression defines a (forward) LS problem [see (2)] whose scalar desired response is the input of the first channel. Proceeding similarly to the forward problem of the previous algorithm (Appendix A), we easily deduce

$$\tilde{R}_{k+1}^{-1}(N) = \begin{bmatrix} \dfrac{1}{\tilde{a}_k^{(1)}(N)} & \boldsymbol{0}^T \\ -\dfrac{1}{\tilde{a}_k^{(1)}(N)}\tilde{R}_k^{-1}(N-1)\boldsymbol{p}_k^{(1)}(N) & \tilde{R}_k^{-1}(N-1) \end{bmatrix} \cdot (\hat{Q}_k^{f(1)}(N))^T \tag{31}$$

where $\boldsymbol{p}_k^{(1)}(N)$ is the rotated reference vector, and $\tilde{a}_k^{(1)}(N)$ is the square root of the minimum squared error (energy) corresponding to this LS problem. $\hat{Q}_k^{f(1)}(N)$ is a sequence of k Givens rotations that annihilate $\boldsymbol{p}_k^{(1)}(N)$ with respect to $\tilde{a}_k^{(1)}(N)$ as

$$\hat{Q}_k^{f(1)}(N)\begin{bmatrix} \tilde{a}_k^{(1)}(N) \\ \boldsymbol{p}_k^{(1)}(N) \end{bmatrix} = \begin{bmatrix} \tilde{a}_0^{(1)}(N) \\ \boldsymbol{0} \end{bmatrix}. \tag{32}$$

Note that after the application of the $(k-j)$th rotation matrix in (32), the first element of the resulting vector will be equal to $\tilde{a}_j^{(1)}(N), j = k-1, k-2, \cdots, 0$, where $\tilde{a}_j^{(1)}(N)$ is the energy of the LS problem that corresponds to data matrix consisting of the first $j+1$ columns of $U_{k+1}(N)$. This "order-recursiveness" leads to the pipelining of the algorithm.

Combining (28), (31), and the input vector partition $\boldsymbol{u}_{k+1}^T(N+1) = [u_1(N+1) \quad \boldsymbol{u}_k^T(N)]$, we get

$$\boldsymbol{g}_{k+1}(N+1) = \hat{Q}_k^{f(1)}(N)\begin{bmatrix} r_k^{(1)}(N+1) \\ \boldsymbol{g}_k(N) \end{bmatrix} \tag{33}$$

where

$$\begin{aligned} r_k^{(1)}(N+1) &= \frac{1}{\sqrt{\lambda}\tilde{a}_k^{(1)}(N)}[u_1(N+1) - (\tilde{R}_k^{-1}(N-1) \\ &\quad \cdot \boldsymbol{p}_k^{(1)}(N))^T\boldsymbol{u}_k(N)] \\ &= \frac{e_k^{(1)}(N+1)}{\sqrt{\lambda}\tilde{a}_k^{(1)}(N)} \end{aligned} \tag{34}$$

and where $e_k^{(1)}(N+1)$ is the obvious *a priori* error. From (33), the normalized errors

$$r_j^{(1)}(N+1) = \frac{e_j^{(1)}(N+1)}{\sqrt{\lambda}\tilde{a}_j^{(1)}(N)} \qquad j = 0, 1, \cdots, k \tag{35}$$

are computed, and $r_j^{(1)}(N+1)$ would appear directly in (33) if our initial data matrix consisted of the first $j+1$ columns of $U_{k+1}(N)$.

The time update of $\boldsymbol{p}_k^{(1)}(N)$ is realized according to

$$\hat{Q}_k^{(0)}(N)\begin{bmatrix} \lambda^{1/2}\boldsymbol{p}_k^{(1)}(N) \\ u_1(N+1) \end{bmatrix} = \begin{bmatrix} \boldsymbol{p}_k^{(1)}(N+1) \\ \tilde{e}_k^{(1)}(N+1) \end{bmatrix} \tag{36}$$

where $\hat{Q}_k^{(0)}(N)$ is a sequence of k Givens rotations that annul $-\boldsymbol{g}_k(N)$ with respect to 1 [see (7) and (8)]. During this procedure, the quantities $\delta_j^{(0)}(N)\ j = 1, 2, \cdots, k$ are successively generated. Furthermore, in (36), the "angle-normalized" errors $\tilde{e}_j^{(1)}(N+1)$ are calculated, which are related to $e_j^{(1)}(N+1)$ as

$$e_j^{(1)}(N+1) = \delta_j^{(0)}(N)\tilde{e}_j^{(1)}(N+1) \quad j = 1, 2, \cdots, k. \tag{37}$$

From (12) we also have

$$\tilde{a}_j^{(1)}(N+1) = \sqrt{(\sqrt{\lambda}\tilde{a}_j^{(1)}(N))^2 + (\tilde{e}_j^{(1)}(N+1))^2} \qquad j = 1, 2, \cdots, k. \tag{38}$$

Equations (38) and (36) provide the quantities required in the update of the rotation parameters of $\hat{Q}_p^{f(1)}(N)$ [(32) written at time $N+1$]. Finally, the annuling of $\boldsymbol{g}_{k+1}(N+1)$ obtained from (33)

$$\hat{Q}_{k+1}^{(1)}(N)\begin{bmatrix} -\boldsymbol{g}_{k+1}(N+1) \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{0} \\ \delta_{k+1}^{(1)}(N) \end{bmatrix} \tag{39}$$

yields the rotation angles of $\hat{Q}_{k+1}^{(1)}(N)$ as well as $\delta_j^{(1)}(N)\ j = 1, 2, \cdots, k+1$, which are used in the second forward step described as follows.

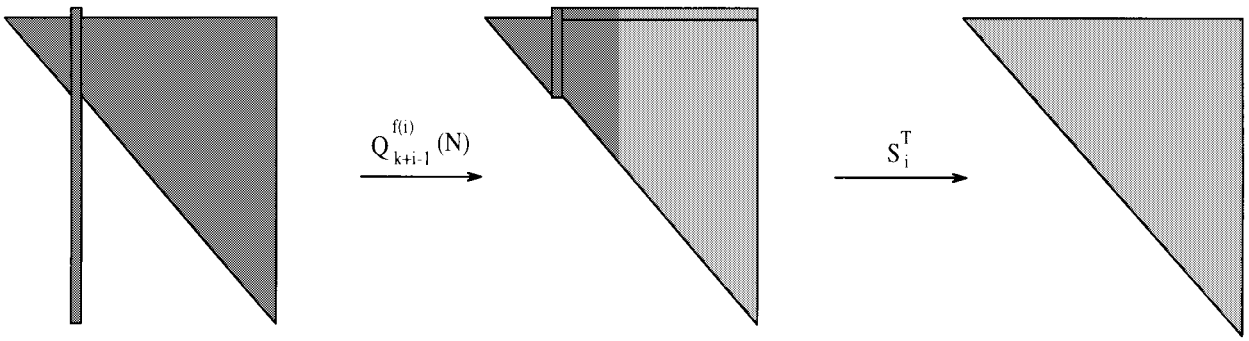

Fig. 1. Construction of the upper triangular factor $\tilde{R}_{k+i}(N)$.

B. Forward Step i for $i = 2, \cdots, l$

The input data matrix $U_{k+i}(N)$ is related to $U_{k+i-1}(N)$ as

$$U_{k+i}(N) = \begin{bmatrix} \lambda^{N-1/2} u_i(1) & \\ \lambda^{N-2/2} u_i(2) & \\ \vdots & U_{k+i-1}(N) \\ u_i(N) & \end{bmatrix} S_i. \tag{40}$$

The derivation of an expression between $\tilde{R}_{k+i}(N)$ and $\tilde{R}_{k+i-1}(N)$ is again necessary. Indeed, if $Q_{k+i-1}(N)$ stands for the orthogonal factor in the QR decomposition of $U_{k+i-1}(N)$, we have

$$\begin{bmatrix} 1 & \mathbf{0}^T \\ \mathbf{0} & Q_{k+i-1}(N) \end{bmatrix} \begin{bmatrix} \mathbf{0}^T \\ U_{k+i}(N) \end{bmatrix} = \begin{bmatrix} 0 & \mathbf{0}^T \\ \boldsymbol{p}_{k+i-1}^{(i)}(N) & \tilde{R}_{k+i-1}(N) \\ \boldsymbol{v}_{k+i-1}^{(i)}(N) & \mathbf{0} \end{bmatrix} S_i. \tag{41}$$

After annuling $\boldsymbol{v}_{k+i-1}^{(i)}(N)$ against the first element of the matrix with an appropriate orthogonal factor, the upper $(k+i) \times (k+i)$ part of the resulting matrix, say, $\hat{R}_{k+i}(N)$, will have the form

$$\hat{R}_{k+i}(N) = \begin{bmatrix} \tilde{a}_{k+i-1}^{(i)}(N) & \mathbf{0}^T \\ \boldsymbol{p}_{k+i-1}^{(i)}(N) & \tilde{R}_{k+i-1}(N) \end{bmatrix} S_i. \tag{42}$$

Note that the existence of S_i in (42) prevents the application of the usual technique, that is, the zeroing of $\boldsymbol{p}_{k+i-1}^{(i)}(N)$ against $\tilde{a}_{k+i-1}^{(i)}(N)$. The procedure that transforms $\hat{R}_{k+i}(N)$ into an upper triangular factor is illustrated in Fig. 1. We initially premultiply $\hat{R}_{k+i}(N)$ with a sequence of $(k+i)-m_i$ elementary Givens rotations $\hat{Q}_{k+i-1}^{f(i)}(N)$, which nullify the last $(k+i)-m_i$ elements of $\boldsymbol{p}_{k+i-1}^{(i)}(N)$ with respect to $\tilde{a}_{k+i-1}^{(i)}(N)$ in a bottom-up procedure. This step does not affect the upper triangular structure of $\tilde{R}_{k+i-1}(N)$ and creates $(k+i)-m_i+1$ nonzero elements at the end of the first row of the resulting matrix. The new matrix is then premultiplied by a permutation factor that moves its first row to the m_i position after upshifting its next m_i-1 rows. Clearly, this permutation factor coincides with S_i^T. It is not difficult to verify that the above procedure leads to a $(k+i) \times (k+i)$ upper triangular positive definite factor under the condition that $\tilde{a}_{k+i-1}^{(i)}(N)$ and the diagonal elements of $\tilde{R}_{k+i-1}(N)$ are positive. The latter is valid in our case, which means that the final factor equals $\tilde{R}_{k+i}(N)$, that is

$$\tilde{R}_{k+i}(N) = S_i^T \hat{Q}_{k+i-1}^{f(i)}(N) \cdot \begin{bmatrix} \tilde{a}_{k+i-1}^{(i)}(N) & \mathbf{0}^T \\ \boldsymbol{p}_{k+i-1}^{(i)}(N) & \tilde{R}_{k+i-1}(N) \end{bmatrix} S_i. \tag{43}$$

Equation (43) establishes the connection between $\tilde{R}_{k+i-1}(N)$ and $\tilde{R}_{k+i}(N)$. Combining (28), the inverted form of the matrix in (43), and the input vector partition $\boldsymbol{u}_{k+i}^T(N+1) = [u_i(N+1) \quad \boldsymbol{u}_{k+i-1}^T(N+1)] S_i$, we obtain

$$\boldsymbol{g}_{k+i}(N+1) = S_i^T \hat{Q}_{k+i-1}^{f(i)}(N) \begin{bmatrix} r_{k+i-1}^{(i)}(N+1) \\ \boldsymbol{g}_{k+i-1}(N+1) \end{bmatrix}. \tag{44}$$

From the definitions of S_i and $\hat{Q}_{k+i-1}^{f(i)}(N)$, we conclude that the first $m_i - 1$ elements of $\boldsymbol{g}_{k+i}(N+1), \boldsymbol{g}_{k+i-1}(N+1)$ are identical. Furthermore, the application of the rotations of $\hat{Q}_{k+i-1}^{f(i)}(N)$ in (44) successively produces the normalized errors $r_{k+i-1-t}^{(i)}(N+1), t = 1, 2, \cdots, (k+i) - m_i$. If our initial data matrix coincided with the first $(k+i)-t$ columns of $U_{k+i}(N)$, then $r_{k+i-1-t}^{(i)}(N+1)$ would appear directly in an equation of the type of (44). In such an equation, the corresponding $\hat{Q}^{f(i)}$ factor would be identical to the product of the last $(k+i)-m_i-t$ rotation matrices of $\hat{Q}_{k+i-1}^{f(i)}(N)$, and the corresponding $\boldsymbol{g}$-vectors would be equal to the upper $(k+i-t)\times 1$ and $(k+i-1-t)\times 1$ blocks of $\boldsymbol{g}_{k+i}(N+1)$ and $\boldsymbol{g}_{k+i-1}(N+1)$, respectively. Especially for $t = (k+i) - m_i$, the orthogonal factor $\hat{Q}^{f(i)}$ degenerates to the identity matrix, and the action of S_i^T indicates that $r_{m_i-1}^{(i)}(N+1)$ is the m_ith element of $\boldsymbol{g}_{k+i}(N+1)$. In a similar way, we conclude that the lower order energies $\tilde{a}_{k+i-i-t}^{(i)}, t = 1, 2, \cdots, (k+i) - m_i$ are successively produced in (43) through the application of the rotation matrices of $\hat{Q}_{k+i-1}^{f(i)}(N)$.

The time update of $\boldsymbol{p}_{k+i-1}^{(i)}(N)$ is accomplished according to

$$\hat{Q}_{k+i-1}^{(i-1)}(N) \begin{bmatrix} \lambda^{1/2} \boldsymbol{p}_{k+i-1}^{(i)}(N) \\ u_i(N+1) \end{bmatrix} = \begin{bmatrix} \boldsymbol{p}_{k+i-1}^{(i)}(N+1) \\ \tilde{c}_{k+i-1}^{(i)}(N+1) \end{bmatrix}. \tag{45}$$

In the last equation, except for $\tilde{e}_{k+i-1}^{(i)}(N+1)$, all lower order angle normalized errors are calculated as well. The rotations required in the next forward step are now computed from

$$\hat{Q}_{k+i}^{(i)}(N+1) \begin{bmatrix} -\boldsymbol{g}_{k+i}(N+1) \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \delta_{k+i}^{(i)}(N) \end{bmatrix}. \tag{46}$$

It has been stated, however, that the first $m_i - 1$ elements of $\boldsymbol{g}_{k+i}(N+1)$ coincide with the corresponding elements of $\boldsymbol{g}_{k+i-1}(N+1)$. Therefore, the first $m_i - 1$ rotation parameters of the ith forward step have already been calculated in the previous forward steps. Such an observation can lead to a significant reduction of the computational complexity of the proposed algorithm, depending on the channel orders, and will be revisited later on. Especially the rotations produced at the lth forward step pass to the filtering part of the algorithm as well as to the first forward step of the next time instant. Note from (29) that only the first k (out of $k+l$) rotation parameters, which are produced in (46) for $i = l$, are essentially necessary. As a consequence, in all forward steps, we can deal with the upper $k \times 1$ blocks of the corresponding vectors $\boldsymbol{g}$. Based on this observation, the initial angle normalized error in (44) will be given by

$$r_k^{(i)}(N+1) = \frac{\tilde{e}_k^{(i)}(N+1)\delta_k^{(i-1)}(N)}{\sqrt{\lambda}\tilde{a}_k^{(i)}(N)}$$

while the update of $\tilde{a}_k^{(i)}(N)$ is obtained as

$$\tilde{a}_k^{(i)}(N+1) = \sqrt{(\sqrt{\lambda}\tilde{a}_k^{(i)}(N))^2 + (\tilde{e}_k^{(i)}(N+1))^2}$$

The new channel decomposition based algorithm is shown in Table II. In Table II, $\theta_j^{(i-1)}(N), j = 1, 2, \cdots, k$ stand for the first k rotation angles of $\hat{Q}_{k+i-1}^{(i-1)}(N)$, whereas $\phi_j^{(i)}(N+1), j = m_i, m_i+1, \cdots, k$ are the last $k - m_i + 1$ rotation angles of $\hat{Q}_{k+i-1}^{f(i)}(N+1)$. $p_j^{(i)}(N)$ is the jth element of $\boldsymbol{p}_{k+i-1}^{(i)}(N)$, and $g_{j-1}^{(i)}(N+1)$ denotes the jth element of $\boldsymbol{g}_{k+i}(N+1)$. $p_j(N)$ stands for the jth element of the $k \times 1$ vector $\boldsymbol{p}_k(N)$ [corresponding to $P(N)$ of (4) for the single output, different channel orders case]. In order to maintain a unified notation, the following conventions are adopted for the lth forward step: $\theta_j^{(l)}(N) = \theta_j^{(0)}(N+1), \delta_j^{(l)}(N) = \delta_j^{(0)}(N+1)$ and $g_{j-1}^{(l)}(N) = g_{j-1}^{(0)}(N+1), j = 1, 2, \cdots, k$.

TABLE II
NEW CHANNEL DECOMPOSITION-BASED QRD ALGORITHM

$\delta_0^{(1)}(N) = 1; \tilde{e}_0(N+1) = y(N+1);$
for $i = 1 : l$,
  $\tilde{e}_0^{(i)}(N+1) = u_i(N+1);$
  for $j = 1 : k$, (**Step 1**)
    $p_j^{(i)}(N+1) = \lambda^{1/2}\cos[\theta_j^{(i-1)}(N)]p_j^{(i)}(N) + \sin[\theta_j^{(i-1)}(N)]\tilde{e}_{j-1}^{(i)}(N+1);$
    $\tilde{e}_j^{(i)}(N+1) = \cos[\theta_j^{(i-1)}(N)]\tilde{e}_{j-1}^{(i)}(N+1) - \lambda^{1/2}\sin[\theta_j^{(i-1)}(N)]p_j^{(i)}(N);$
  end;
  $r_k^{(i)}(N+1) = \frac{\tilde{e}_k^{(i)}(N+1)\delta_k^{(i-1)}(N)}{\sqrt{\lambda}\tilde{a}_k^{(i)}(N)};$
  for $j = k : -1 : m_i$, (**Step 2**)
    $r_{j-1}^{(i)}(N+1) = \cos[\phi_j^{(i)}(N)]r_j^{(i)}(N+1) - \sin[\phi_j^{(i)}(N)]g_{j-1}^{(i-1)}(N+1);$
    $g_j^{(i)}(N+1) = -\sin[\phi_j^{(i)}(N)]r_j^{(i)}(N+1) + \cos[\phi_j^{(i)}(N)]g_{j-1}^{(i-1)}(N+1);$
  end;
  $g_{m_i-1}^{(i)}(N+1) = r_{m_i-1}^{(i)}(N+1);$
  $\tilde{a}_k^{(i)}(N+1) = \sqrt{(\sqrt{\lambda}\tilde{a}_k^{(i)}(N))^2 + (\tilde{e}_k^{(i)}(N+1))^2};$
  for $j = k : -1 : m_i$, (**Step 3**)
    $\tilde{a}_{j-1}^{(i)}(N+1) = \sqrt{(\tilde{a}_j^{(i)}(N))^2 + (p_j^{(i)}(N+1))^2};$
    $\cos[\phi_j^{(i)}(N+1)] = \frac{\tilde{a}_j^{(i)}(N+1)}{\tilde{a}_{j-1}^{(i)}(N+1)};$
    $\sin[\phi_j^{(i)}(N+1)] = \frac{p_j^{(i)}(N+1)}{\tilde{a}_{j-1}^{(i)}(N+1)};$
  end;
  for $j = m_i : k$, (**Step 4**)
    $\delta_j^{(i)}(N) = \sqrt{(\delta_{j-1}^{(i)}(N))^2 + (g_{j-1}^{(i)}(N+1))^2};$
    $\cos[\theta_j^{(i)}(N)] = \frac{\delta_{j-1}^{(i)}(N)}{\delta_j^{(i)}(N)};$
    $\sin[\theta_j^{(i)}(N)] = \frac{g_{j-1}^{(i)}(N+1)}{\delta_j^{(i)}(N)};$
  end;
end; { i-loop }
for $j = 1 : k$, (**Step 5**)
  $p_j(N+1) = \lambda^{1/2}\cos[\theta_j^{(0)}(N+1)]p_j(N) + \sin[\theta_j^{(0)}(N+1)]\tilde{e}_{j-1}(N+1);$
  $\tilde{e}_j(N+1) = \cos[\theta_j^{(0)}(N+1)]\tilde{e}_{j-1}(N+1) - \lambda^{1/2}\sin[\theta_j^{(0)}(N+1)]p_j(N);$
end;
$e_k(N+1) = \delta_k^{(0)}(N+1)\tilde{e}_k(N+1);$
Initialization
$\mathbf{p}_k(0) = \mathbf{0}$, $g_j^{(l)}(0) = 0$, $\cos[\theta_j^{(0)}(0)] = 1$, $j = 1, 2, \ldots, k$
$p_j^{(i)}(0) = 0$, $\cos[\phi_j^{(i)}(0)] = 1$, $i = 1, 2, \ldots, l$, $j = 1, 2, \ldots, m_i$
$\delta_k^{(0)}(0) = 1$, $\tilde{a}_k^{(i)}(0) = \mu$, $i = 1, 2, \ldots, l$

C. Alternative Form of the Algorithm

We observe that the prediction section of the algorithm of Table II consists of l similar distinct parts, which correspond to the l input channels. The $(i+1)$th part is excecuted after the completion of the procedure for the ith channel, from which the necessary quantities $\theta^{(i)}, \delta^{(i)}$, and $g^{(i)}$ are collected. This sequential procedure can be avoided if a slightly different approach is adopted for the excecution of steps 2 and 3 of Table II. This approach is based on an alternative computation of the quantities $r_j^{(i)}(N+1)$ and $\tilde{a}_j^{(i)}(N+1)$. Indeed, if we use the relations

$$r_j^{(i)}(N+1) = \frac{\tilde{e}_j^{(i)}(N+1)\delta_j^{(i-1)}(N)}{\sqrt{\lambda}\tilde{a}_j^{(i)}(N)} \qquad \begin{matrix} i = 1, 2, \cdots, l \\ j = m_i, \cdots, k \end{matrix} \tag{47}$$

and

$$\tilde{a}_j^{(i)}(N+1) = \sqrt{(\sqrt{\lambda}\tilde{a}_j^{(i)}(N))^2 + (\tilde{e}_j^{(i)}(N+1))^2} \qquad i = 1, 2, \cdots, l \quad j = m_i, \cdots, k \tag{48}$$

the "ascending" excecution of steps 2 and 3 of Table II is achieved. In this way, we are led to the algorithm shown in Table III. The main feature of this algorithm is that the l forward steps can be excecuted in a pipelined fashion: a fact that favors its systolic implementation. A systolic architecture for the implementation of the algorithm of Table III is shown in Fig. 2. The architecture comprises l identical sections, and each section consists of k blocks. The ith section is excited from the ith channel and essentially implements the ith forward step. The blocks of the ith section transfer the necessary values of $g^{(i)}, \theta^{(i)}$, and $\delta^{(i)}$ to the corresponding blocks of the $(i+1)$th section. The last (lth) section passes

TABLE III
ALTERNATIVE FORM OF THE ALGORITHM OF TABLE II

$\delta_0^{(1)}(N) = 1; \tilde{e}_0(N+1) = y(N+1);$

$g_0^{(1)}(N+1) = \frac{u_1(N+1)}{\sqrt{\lambda}\tilde{a}_0^{(1)}(N)};$

$\tilde{a}_0^{(1)}(N+1) = \sqrt{(\sqrt{\lambda}\tilde{a}_0^{(1)}(N))^2 + (u_1(N+1))^2};$

for $i = 1 : l$,

$\tilde{e}_0^{(i)}(N+1) = u_i(N+1);$

for $j = 1 : k$,

$p_j^{(i)}(N+1) = \lambda^{1/2}\cos[\theta_j^{(i-1)}(N)]p_j^{(i)}(N) + \sin[\theta_j^{(i-1)}(N)]\tilde{e}_{j-1}^{(i)}(N+1);$

$\tilde{e}_j^{(i)}(N+1) = \cos[\theta_j^{(i-1)}(N)]\tilde{e}_{j-1}^{(i)}(N+1) - \lambda^{1/2}\sin[\theta_j^{(i-1)}(N)]p_j^{(i)}(N);$

If $j \geq m_i - 1$,

$\tilde{a}_j^{(i)}(N+1) = \sqrt{(\sqrt{\lambda}\tilde{a}_j^{(i)}(N))^2 + (\tilde{e}_j^{(i)}(N+1))^2};$

$r_j^{(i)}(N+1) = \frac{\tilde{e}_j^{(i)}(N+1)\delta_j^{(i-1)}(N)}{\sqrt{\lambda}\tilde{a}_j^{(i)}(N)};$

If $j = m_i - 1$,

$g_j^{(i)}(N+1) = r_j^{(i)}(N+1);$

If $j > m_i - 1$,

$g_j^{(i)}(N+1) = -\sin[\phi_j^{(i)}(N)]r_j^{(i)}(N+1) + \cos[\phi_j^{(i)}(N)]g_{j-1}^{(i-1)}(N+1);$

$\cos[\phi_j^{(i)}(N+1)] = \frac{\tilde{a}_j^{(i)}(N+1)}{\tilde{a}_{j-1}^{(i)}(N+1)};$

$\sin[\phi_j^{(i)}(N+1)] = \frac{p_j^{(i)}(N+1)}{\tilde{a}_{j-1}^{(i)}(N+1)};$

$\delta_j^{(i)}(N) = \sqrt{(\delta_{j-1}^{(i)}(N))^2 + (g_{j-1}^{(i)}(N+1))^2};$

$\cos[\theta_j^{(i)}(N)] = \frac{\delta_{j-1}^{(i)}(N)}{\delta_j^{(i)}(N)};$

$\sin[\theta_j^{(i)}(N)] = \frac{g_{j-1}^{(i)}(N+1)}{\delta_j^{(i)}(N)};$

end; { j-loop }

end; { i-loop }

for $j = 1 : k$,

$p_j(N+1) = \lambda^{1/2}\cos[\theta_j^{(0)}(N+1)]p_j(N) + \sin[\theta_j^{(0)}(N+1)]\tilde{e}_{j-1}(N+1);$

$\tilde{e}_j(N+1) = \cos[\theta_j^{(0)}(N+1)]\tilde{e}_{j-1}(N+1) - \lambda^{1/2}\sin[\theta_j^{(0)}(N+1)]p_j(N);$

end;

$e_k(N+1) = \delta_k^{(0)}(N+1)\tilde{e}_k(N+1);$

Initialization

$\mathbf{p}_k(0) = \mathbf{0}$, $g_j^{(l)}(0) = 0$, $\cos[\theta_j^{(0)}(0)] = 1$, $\delta_j^{(0)}(0) = 1$, $j = 1, 2, \ldots, k$

$p_j^{(i)}(0) = 0$, $\cos[\phi_j^{(i)}(0)] = 1$, $i = 1, 2, \ldots, l$, $j = 1, 2, \ldots, m_i$

$\tilde{a}_j^{(i)}(0) = \mu$, $i = 1, 2, \ldots, l$, $j = m_i - 1, \ldots, k$

these quantities to the first channel, leading to a circular implementation. It also sends the angles $\theta_j^{(l)}$ to the filtering section of the architecture (not shown in Fig. 2).

The proposed architecture is pipelinable at the order level, that is, the throughput provided is constant and independent of k. This is achieved if we let the l inputs to be applied in a skewed manner from top-to-bottom, i.e., $u_1(N+1)$ is first applied, then $u_2(N+1)$, etc. Thus, $u_1(N+2)$ excites the first section l "clock cycles" after the application of $u_1(N+1)$. This is so because for $j \geq m_l$, the rotations $\theta_j^{(0)}(N+1)$ required in the first forward step are produced in the lth section at the previous time instant N. The skewing of the inputs ensures the synhronization of the different building blocks of the circular architecture, which provides the output error every l "clock cycles."

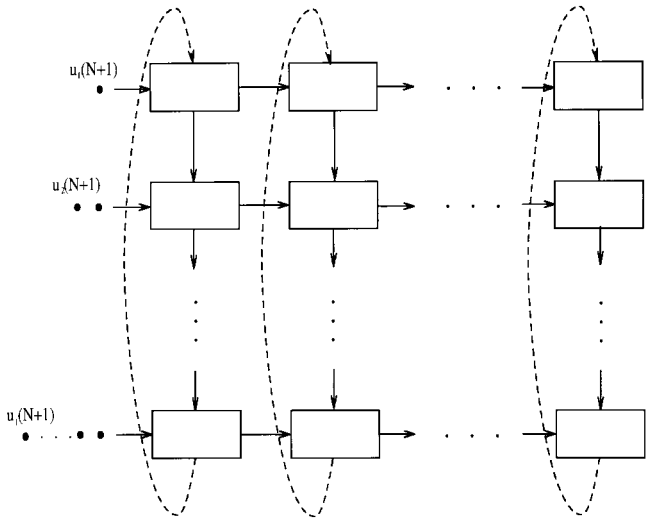

Fig. 2. Systolic architecture for the implementation of the algorithm of Table III.

D. Remarks

It can be shown that the channel decomposition based algorithm of [29] can be obtained if we adopt the vector

$$\boldsymbol{q}_k(N) = \tilde{R}_k^{-T}(N)\boldsymbol{u}_k(N)$$

in place of $\boldsymbol{g}_k$. Then, a similar methodology can also be followed for the derivation of this algorithm.[2] In other words, our technique provides a unified framework for the development of multichannel fast QRD algorithms. The proposed technique is simple, direct, and insightful with respect to the internal algorithmic quantities, which have a specific LS meaning and interpretation. Moreover, explicit backward steps are avoided [(19) and (29)], and our methodology essentialy comprises forward steps. This is in contrast to other methodologies, where the coexistence of forward and backward steps increases the complexity of the derivation procedure.

The multichannel algorithms of Tables II and III are based exclusively on orthogonal Givens rotations. As a result, their numerical performance is expected to be favorable. For channels of equal orders $(k = lp)$, the new schemes are of $O(pl^2)$ computational complexity, which is lower by an order of magnitude compared with the complexity of the algorithm of Section III and is similar to that of other known channel decomposition-based fast QRD schemes [27], [28]. In the general case of different channel orders, however, the proposed algorithms are of lower computational complexity if compared with the fast QRD scheme of [29] (which also treats unequal channel lengths). This concerns not only the number of multiplications/divisions but also, and more importantly, the number of square roots, as shown in Table IV. The complexity reduction is due to the fact that the $\boldsymbol{g}$ vectors of different forward problems have common blocks. Therefore, some rotation parameters of $\hat{Q}^{(i)}$'s—produced in a forward manner—are also common among the different channels and need to be computed only once. There does not exist such a

[2]The same holds for the block and channel decomposition based algorithms of [27] and [28], which are obtained if we follow the analysis of Sections III and IV.

TABLE IV
COMPARISON OF COMPLEXITIES OF CHANNEL DECOMPOSITION-BASED MULTICHANNEL FAST QRD ALGORITHMS

Algorithm	Mults/Divs	Square roots
Algorithm of Table II	$17kl - 12\Sigma_{i=1}^{l} m_i + 12l + 5k$	$2kl - 2\Sigma_{i=1}^{l} m_i + 2l$
Algorithm of Table III	$18kl - 13\Sigma_{i=1}^{l} m_i + 13l + 5k$	$2kl - 2\Sigma_{i=1}^{l} m_i + 2l$
Algorithm of [29]	$18kl - 8\Sigma_{i=1}^{l} m_i + 8l + 5k$	$2kl - \Sigma_{i=1}^{l} m_i + 2l$

TABLE V
COMPARISON OF COMPLEXITIES FOR SECOND-ORDER VOLTERRA FILTERING

Algorithm	Mults/Divs	Square roots
Algorithm of Table II	$6.5L^3 + 21.5L^2 + O(L)$	$\frac{2}{3}L^3 + 1.5L^2 + O(L)$
Algorithm of Table III	$6.83L^3 + 22.5L^2 + O(L)$	$\frac{2}{3}L^3 + 1.5L^2 + O(L)$
Algorithm of [29]	$7.66L^3 + 38.5L^2 + O(L)$	$\frac{5}{6}L^3 + 2.75L^2 + O(L)$

possibility for the algorithm of [29] because the corresponding rotation parameters are produced from the $\boldsymbol{q}$-vectors in a backward manner. As shown in Table IV, the reduction in complexity, which is offered by the new algorithms, depends on the channel lengths. A specific example is presented in the next section. From Table IV, we also observe that the computational complexity of the algorithm of Table III is slightly higher if compared with the algorithm of Table II. This is due to the use of (19) and (29) for the computation of $r_j^{(i)}(N+1)$ and $\tilde{a}_j^{(i)}(N+1)$, respectively. However, the algorithm of Table III is pipelinable at the order level and is implementable on a very regular systolic architecture. In contrast, the algorithm of Table II, as well as the fast channel decomposition-based QRD algorithms of [27]–[29], are strictly sequential for each time iteration.

V. APPLICATIONS-EXPERIMENTAL RESULTS

In this section, two specific applications, which accept a multichannel formulation, are discussed: Volterra-type nonlinear adaptive filtering and decision feedback adaptive channel equalization. Computer simulations demonstrate the validity of the proposed channel decomposition based algorithms when they are used as the necessary adaptive tool. Experiments in a limited precision environment are also provided.

A. Second-Order Volterra Filtering

We assume that the input–output relation of our reference model is described by the second-order triangular Volterra representation [29], [32], [33]

$$y(n) = \sum_{n_1=0}^{L-1} c_{n_1}(N)u(n-n_1) + \sum_{n_1=0}^{L-1}\sum_{n_2=n_1}^{L-1} c_{n_1,n_2}(N) \cdot u(n-n_1)u(n-n_2) + \eta(n) \tag{49}$$

where $c_{n_1}(N), c_{n_1,n_2}(N)$ stand for the linear and quadratic coefficients of the system at time N, and $\eta(n)$ represents a noise term. The nonlinear filtering problem of (49) can be transformed to an equivalent linear multichannel filtering problem with channels of unequal orders [29], [32], [33]. This is accomplished by considering $l = L+1$ channels, whose inputs at time n are expressed as

$$u_i(n) = \begin{cases} u(n), & i = 1 \\ u(n)u(n-i+2), & i = 2, \cdots, L+1 \end{cases}$$

and their orders are

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

Clearly, the output of the nonlinear model (49) can be adaptively estimated in the LS sense if a multichannel LS algorithm that is capable of processing different channel lengths is employed.

The computational requirements of the algorithms described in Section IV for the above Volterra-type problem are shown in Table V. Table V also includes the number of operations for the corresponding scheme of [29], which is the only known to the authors multichannel fast QRD algorithm treating diferrent channel orders.[3] We observe that the new algorithms offer significant computational savings in terms of both the number of multiplications/divisions (15% and 11% less) and the number of square roots (20% less). This improvement can be even greater for a different application (in the higher order Volterra case, for instance).

We have implemented the algorithm of Table II for a second-order Volterra system with $L = 5$. The input and noise terms are zero mean white Gaussian signals. The variance of the input was taken equal to 1, and the forgetting factor $\lambda = 0.98$. The squared error resulting from the application of the new multichannel scheme is depicted in Fig. 3. The curves of Fig. 3 are the average of 20 independent realizations of the algorithm and correspond to noise variances 10^{-2} and 10^{-3}, respectively. The fast convergence rate that characterizes the RLS type algorithms is clear from Fig. 3.

[3] In [29, Tab. III], the number of multiplications/divisions and the number of square roots are calculated as $9L^3 + (69L^2/2) + (57L/2) - 1$ and $L^3 + (7L^2/2) + (7L/2)$, respectively. We believe, however, that careful measuring leads to the figures of Table V.

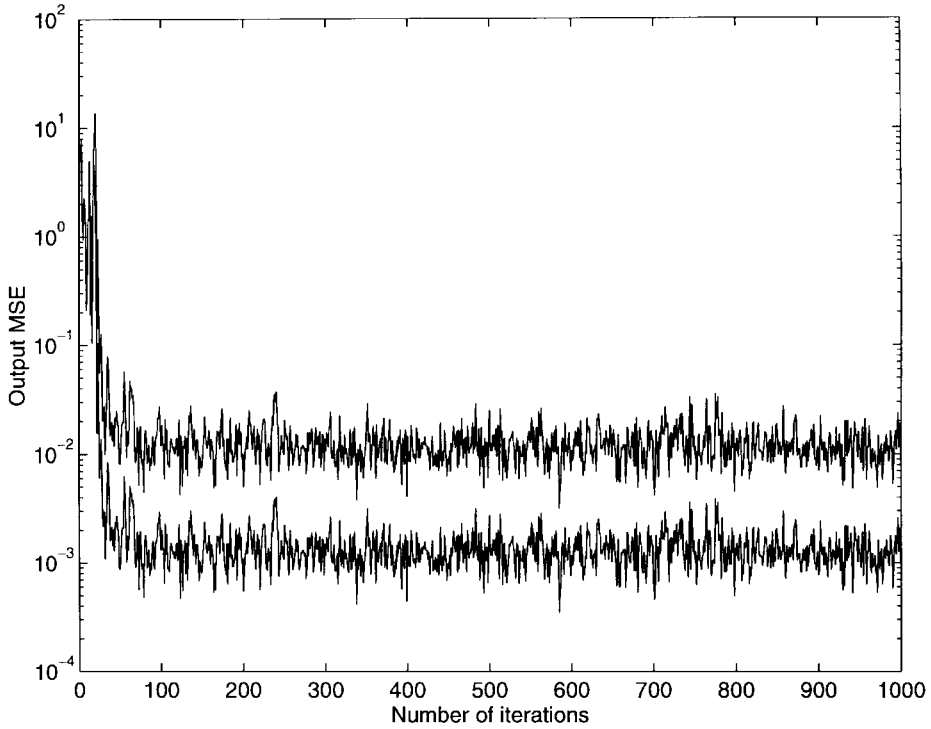

Fig. 3. Initial convergence curves of the Volterra filtering problem.

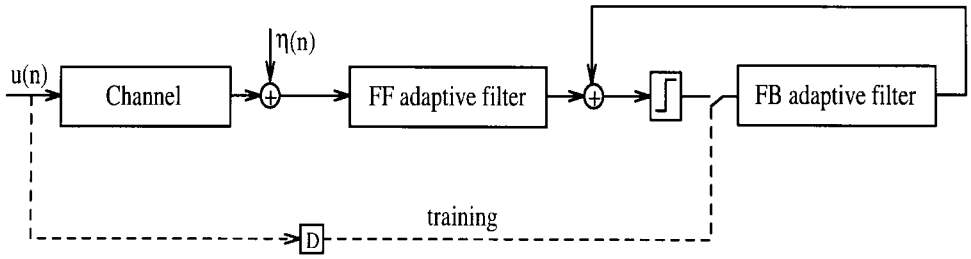

Fig. 4. Full decision feedback channel equalizer.

B. Decision Feedback Adaptive Channel Equalization

The basic decision feedback channel equalization problem [1] is illustrated in Fig. 4. An input sequence $u(n)$ taking values from a binary alphabet (± 1) excites a communication channel with nonminimum phase characteristics. The channel introduces both time dispersion in the form of intersymbol interference and additive noise ($\eta(n)$ in the figure). The reconstruction of the initial input sequence from the channel output samples is at the heart of the channel equalization problem.

A full decision feedback equalizer structure consists of a feedforward (FF) adaptive filter and a feedback (FB) adaptive filter [1]. The FF filter, which introduces a delay into the equalizer, removes the precursor part of the channel's impulse response while the FB filter cancels the part following the main peak of the channel's impulse response. Therefore, either two seperate single channel adaptive algorithms can be used or, following a multichannel interpretation, a multichannel algorithm with $l = 2$ can be applied.

We have implemented a decision feedback equalizer using the algorithm of Table III for a channel with the nonminimum phase transfer function

$$H(z) = 0.3627 + 0.1030z^{-1} - 0.7396z^{-2} + 0.3859z^{-3} - 0.4024z^{-4}.$$

The channel coefficients are properly chosen to assure that the variance of the output signal equals 1. The noise term is a zero mean white Gaussian process of variance 10^{-3}, and the forgetting factor $\lambda = 0.98$. The lengths of the FF and the FB sections are taken equal to 5 and 2, respectively, whereas a delay equal to 1 is sufficient. Fig. 5 shows the equalizer output error with the equalizer operating in the initial training period. The curve of Fig. 5 is the average of 40 independent realizations. We observe that the new algorithm combines fast initial convergence with low complexity and numerical robustness, which are highly desirable in a channel equalization application.

C. Numerical Simulations

In order to compare the numerical behavior of channel decomposition-based fast QRD algorithms, experiments have been performed in a finite precision environment. Specifically,

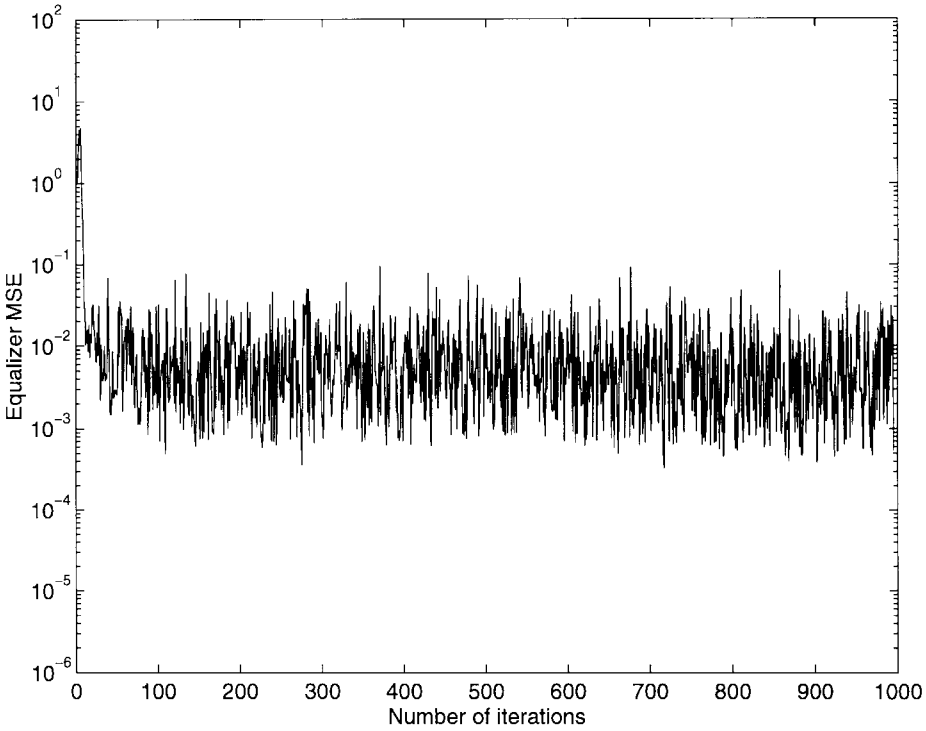

Fig. 5. Average output error of the equalizer in the training period.

the mantissa in the single precision floating-point representation is truncated in a specified position, whereas the exponent remains unaffected. The outcome of each arithmetic operation of the algorithms under study is immediately adjusted to the predifined mantissa length. In the following experiments, we consider that the desired signal $y(n)$ obeys to a three-input channel model of the form

$$y(n) = \boldsymbol{c}_{k_1}^T \boldsymbol{u}_{k_1}(n) + \boldsymbol{c}_{k_2}^T \boldsymbol{u}_{k_2}(n) + \boldsymbol{c}_{k_3}^T \boldsymbol{u}_{k_3}(n) + \eta(n).$$

The orders of $\boldsymbol{c}_{k_1}, \boldsymbol{c}_{k_2}, \boldsymbol{c}_{k_3}$ are 6, 4, and 3, respectively, and their elements are chosen randomly. The noise term $\eta(n)$ is a white Gaussian sequence with variance 10^{-2}. The input signals are autoregressive sequences of orders 3, 2, and 2, respectively, i.e.,

$$\begin{aligned} u_1(n) &= 0.9u_1(n-1) + 0.64u_1(n-2) - 0.576u_1(n-3) \\ &\quad + 0.188\eta_1(n) \\ u_2(n) &= -0.4787u_2(n-1) + 0.8u_2(n-2) + 0.25\eta_2(n) \\ u_3(n) &= 0.5u_3(n-1) - 0.9u_3(n-2) + 0.42\eta_3(n). \end{aligned}$$

The terms $\eta_1(n), \eta_2(n), \eta_3(n)$ are white Gaussian sequences of variance 1, and the coefficients of the above AR equations were chosen so that the input signal variances are also equal to 1. In our experiments, the forgetting factor λ equals 0.99. The *a priori* mean squared error of two-channel decomposition-based fast QRD algorithms for different mantissa lengths is shown in Fig. 6. For a specific number of mantissa bits, 50 different realizations of the algorithms were run, each of 3000 iterations. The last 1000 samples of the *a priori* mean squared error are then used for the evaluation of a time average value. Following this procedure, the value of each point in Fig. 6 is specified. We observe that the numerical accuracy of the algorithm of [29] is slightly better for mantissa lengths 7, 8, and 9. In contrast, for mantissa lengths greater than 9, the mean squared error of the algorithm of Table II is slightly lower. In general, the numerical accuracy of both algorithms seems to be similar: a fact that has also been verified for a variety of initial models and specifications. It is obvious from Fig. 6 that for mantissa lengths 12 and 13, the mean squared error of both algorithms is very close to the value ($\sim 10^{-2}$), which is expected for IEEE single precision floating-point arithmetic (23-bit mantissa). It must be noted that the algorithms were run for a high mumber of iterations ($>1\,000\,000$) with no indication of round-off error accumulation, even for very small mantissa lengths. Similar numerical results are also obtained for the new algorithm of Table III.

However, the algorithm of [29] appears to have numerical problems in the case of nonstationarities of the input signal statistics [13]. Indeed, in Fig. 7, the algorithms are compared when the input signal $u_2(n)$ is disturbed as

$$u_2(n) = -0.4787u_2(n-1) + 0.8u_2(n-2) + 40\eta_2(n) \quad \text{for } n = 400.$$

Even for 7 bits in the mantissa representation, the mean squared error of the new algorithm converges to its expected value (from Fig. 6) following a sudden increase, which is due to the above nonstationarity. In contrast, for mantissa lengths less than 14 bits, at least one of the 50 realizations of the algorithm of [29] stops at the point of the disturbance. This problem relates to a numerically not well-behaved hyberbolic rotation step that is included in the body of the algorithm of [29]. Specifically, due to the low arithmetic precision, the algorithm calls for the calculation of the square root of a negative number, which halts the algorithmic procedure. Note

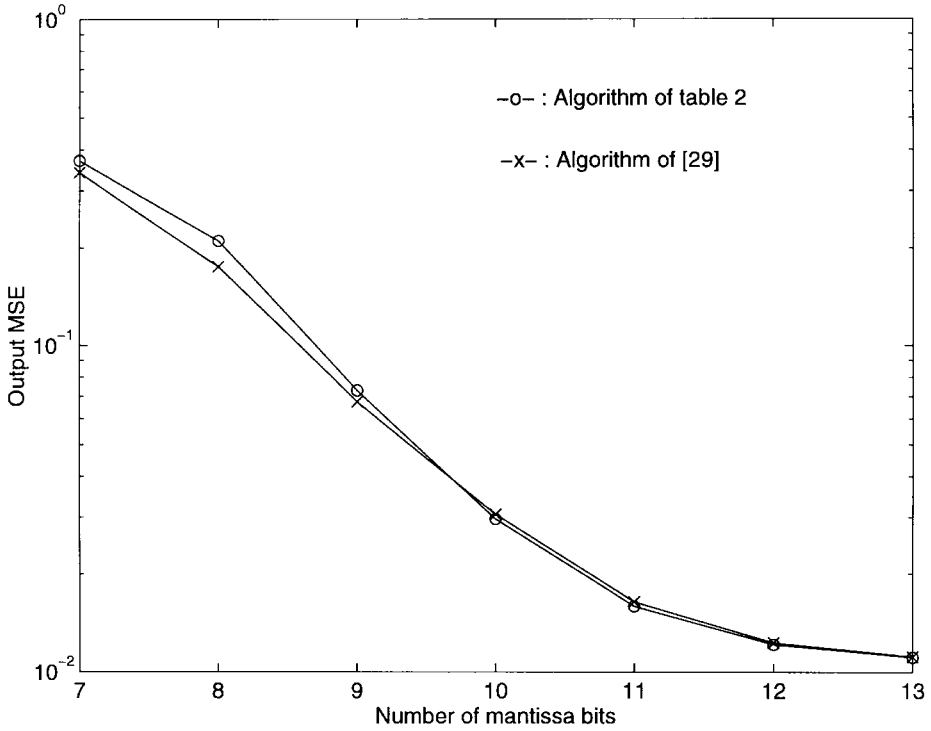

Fig. 6. Mean squared error of two multichannel fast QRD algorithms for different mantissa lengths.

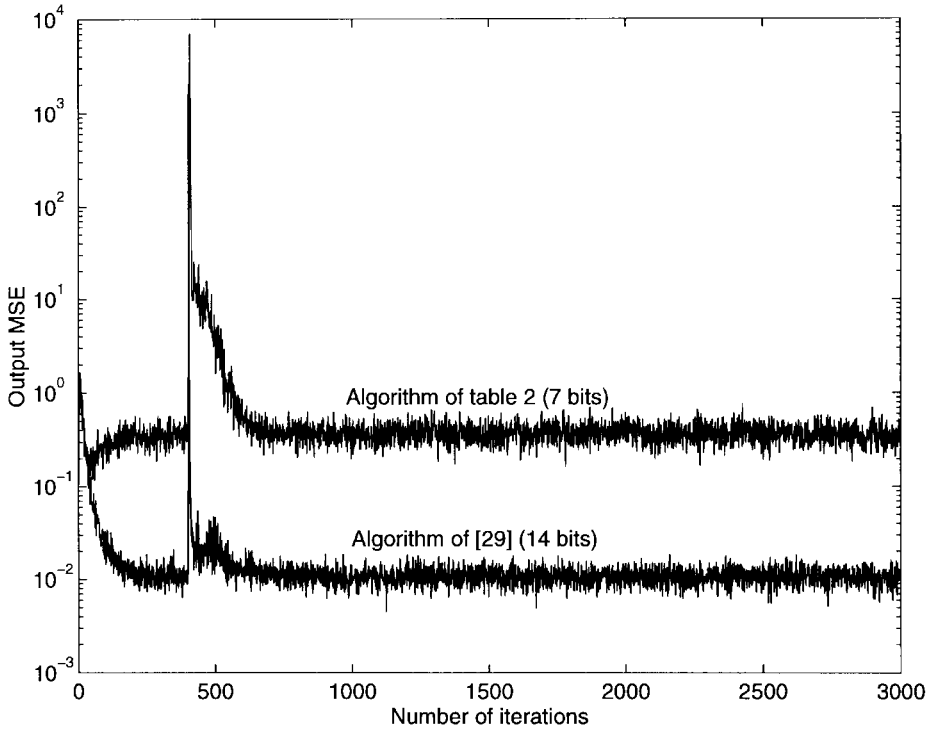

Fig. 7. Mean squared error of two multichannel fast QRD algorithms under a strong nonstationarity.

that the proposed algorithms do not include hyperbolic steps, resulting in superior numerical behavior.

VI. Concluding Remarks

Least squares adaptive algorithms based on the QR decomposition of the input data matrix are very promising due to their numerically robust performance. In this paper, following a novel technique, new multichannel fast QRD algorithms were developed. The new technique is direct and insightful and can easily be applied for the derivation of other, already existing multichannel fast QRD schemes. Besides their good numerical properties, the proposed algorithms exhibit

some other nice features, such as fast convergence, low complexity, and enhanced parallelism and pipelinability. As a consequence, the new algorithms are amenable to efficient implementations on systolic array architectures with short wordlengths and fixed-point arithmetic. As the number of applications that accept a multichannel formulation increases, schemes of the type presented in this paper appear to be appropriate algorithmic tools.

APPENDIX A

From (13), the input data matrix $U_{p+1}(N)$ can be written as

$$\begin{aligned} U_{p+1}(N) &= [U_p(N) \quad Y_p^b(N)] \\ &= \begin{bmatrix} \lambda^{N-1/2}\boldsymbol{u}_1^T & \mathbf{0}^T \\ Y_p^f(N) & U_p(N-1) \end{bmatrix} \end{aligned} \tag{50}$$

where $Y_p^b(N), Y_p^f(N)$ consist of the obvious desired responses of the backward and forward problems, respectively. From the definition of $\tilde{R}_p(N)$ and (4), it is easy to see that

$$Q_p(N)U_{p+1}(N) = \begin{bmatrix} \tilde{R}_p(N) & P_p^b(N) \\ \bigcirc & V_p^b(N) \end{bmatrix}. \tag{51}$$

If now $Q_p^b(N)$ is an orthogonal matrix that converts $V_p^b(N)$ into an $l \times l$ upper triangular form with positive diagonal elements, then (51) gives

$$\begin{bmatrix} I_{lp} & \bigcirc \\ \bigcirc & Q_p^b(N) \end{bmatrix} Q_p(N)U_{p+1}(N) = \begin{bmatrix} \tilde{R}_p(N) & P_p^b(N) \\ \bigcirc & \tilde{A}_p^b(N) \\ \bigcirc & \bigcirc \end{bmatrix}. \tag{52}$$

From (11) and the norm preserving property of $Q_p^b(N)$, it is clear that $\tilde{A}_p^b(N)$ is the Cholesky factor of the backward error covariance matrix. The relation between $\tilde{R}_{p+1}(N)$ and $\tilde{R}_p(N)$ is readily established from (52) as

$$\tilde{R}_{p+1}(N) = \begin{bmatrix} \tilde{R}_p(N) & P_p^b(N) \\ \bigcirc & \tilde{A}_p^b(N) \end{bmatrix}. \tag{53}$$

By using the second equation in (50), the connection between $\tilde{R}_{p+1}(N)$ and $\tilde{R}_p(N-1)$ can also be established. Indeed, after noticing that $\tilde{R}_{p+1}(N)$ is also the Cholesky factor of any matrix of the form $\begin{bmatrix} \bigcirc \\ U_{p+1}(N) \end{bmatrix}$, we get

$$\begin{aligned} &\begin{bmatrix} I_l & \bigcirc \\ \bigcirc & Q_p(N-1) \end{bmatrix} \begin{bmatrix} \bigcirc_{(l-1)\times l(p+1)} \\ U_{p+1}(N) \end{bmatrix} \\ &\quad = \begin{bmatrix} \bigcirc & \bigcirc \\ \lambda^{N-1/2}\boldsymbol{u}_1^T & \mathbf{0}^T \\ P_p^f(N) & \tilde{R}_p(N-1) \\ V_p^f(N) & \bigcirc \end{bmatrix}. \end{aligned} \tag{54}$$

An orthogonal matrix $Q_p^f(N)$ can now be found that transforms the matrix in (54) in the form

$$\begin{aligned} &Q_p^f(N)\begin{bmatrix} I_l & \bigcirc \\ \bigcirc & Q_p(N-1) \end{bmatrix} \begin{bmatrix} \bigcirc_{(l-1)\times l(p+1)} \\ U_{p+1}(N) \end{bmatrix} \\ &\quad = \begin{bmatrix} \tilde{A}_p^f(N) & \bigcirc \\ P_p^f(N) & \tilde{R}_p(N-1) \\ \bigcirc & \bigcirc \end{bmatrix} \end{aligned} \tag{55}$$

where $\tilde{A}_p^f(N)$ stands for the $l \times l$ Cholesky factor of the forward error covariance matrix. After isolating the upper $l(p+1) \times l(p+1)$ block of the matrix in the right-hand side of (55), we easily obtain

$$\tilde{R}_{p+1}(N) = \hat{Q}_p^f(N)\begin{bmatrix} \tilde{A}_p^f(N) & \bigcirc \\ P_p^f(N) & \tilde{R}_p(N-1) \end{bmatrix} \tag{56}$$

where $\hat{Q}_p^f(N)$ is a sequence of orthogonal Givens rotations that annihilate $P_p^f(N)$ with respect to $\tilde{A}_p^f(N)$. Specifically, each row of $P_p^f(N)$ is zeroed against the diagonal elements of the $l \times l$ upper triangular factor [initially $\tilde{A}_p^f(N)$]. The procedure starts from the last row of $P_p^f(N)$, moves upwards, and guarantees the upper triangular structure and the positive-definiteness of the resulting factor. It is now straightforward that inversion of (53) and (56) leads to (15) and (16).

Note that if instead of $U_{p+1}(N)$ we initially considered $U_i(N)$ in (50) for $i = 1, 2, \cdots, p$, then $\tilde{A}_{i-1}^f(N), P_{i-1}^f(N)$, and $\hat{Q}_i^f(N)$ would appear in (56), where $P_{i-1}^f(N)$ is identical to the upper $(i-1)l \times l$ block of $P_p^f(N)$. This observation, in conjuction with the nesting property of the $\tilde{R}$ factors in (53), justify the order recursiveness of the derived algorithm.

REFERENCES

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

[2] S. Haykin, *Adaptive Filter Theory*. Engelwood Cliffs, NJ: Prentice-Hall, 1991.

[3] G. Carayiannis, D. Manolakis, and N. Kalouptsidis, "A unified view of parametric processing algorithms for prewindowed signals," *Signal Process.*, vol. 10, pp. 335–368, 1986.

[4] G. H. Golub and C. Van Loan, *Matrix Computations*, 2nd ed. Baltimore, MD: Johns Hopkins Univ. Press, 1989.

[5] A. H. Sayed and T. Kailath, "A state-space approach to adaptive filtering," *IEEE Signal Processing Mag.*, pp. 18–60, July 1994.

[6] B. Friedlander, "Lattice filters for adaptive processing," *Proc. IEEE*, vol. 70, pp. 829–867, Aug. 1982.

[7] H. Sakai, "Circular lattice filtering using Pagano's method," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-30, pp. 279–287, Apr. 1982.

[8] T. Kawase, H. Sakai, and H. Tokumaru, "Recursive least squares lattice and escalator estimation algorithms," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-31, pp. 228–231, Feb. 1983.

[9] F. Ling and J. G. Proakis, "A generalized multichannel least squares lattice algorithm based on sequential processing stages," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-32, pp. 381–388, Apr. 1984.

[10] A. Lev-Ari, "Modular architectures for adaptive multichannel lattice algorithms," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-35, pp. 543–552, Apr. 1987.

[11] E. Karlsson and M. H. Hayes, "Least squares ARMA modeling of linear time-varying systems: Lattice filter structures and fast RLS algorithms," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-35, pp. 994–1014, July 1987.

[12] P. S. Lewis, "QR-based algorithms for multichannel adaptive least squares lattice filters," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 38, pp. 421–432, Mar. 1990.

[13] B. Yang and J. F. Bohme, "Rotation-based RLS algorithms: Unified derivations, numerical properties, and parallel implications," *IEEE Trans. Signal Processing*, vol. 40, pp. 1151–1167, May 1992.

[14] G.-O. A. Glentis and N. Kalouptsidis, "Fast adaptive algorithms for multichannel filtering and system identification," *IEEE Trans. Signal Processing*, vol. 40, pp. 2433–2458, Oct. 1992.

[15] S. H. Ardalan and L. J. Faber, "A fast ARMA transversal RLS filter algorithm," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 36, pp. 349–458, Mar. 1988.

[16] D. T. M. Slock, L. Chisci, H. Lev-Ari, and T. Kailath, "Modular and numerically stable fast transversal filters for multichannel and

multiexperiment RLS," *IEEE Trans. Signal Processing*, vol. 40, Apr. 1992.
[17] J. Lee and V. J. Mathews, "A fast recursive least squares adaptive second-order Volterra filter and its performance analysis," *IEEE Trans. Signal Processing*, vol. 41, pp. 1087–1102, Mar. 1992.
[18] B. H. Khalaj, A. H. Sayed, and T. Kailath, "A unified derivation of square-root multichannel least-squares filtering algorithms," in *Proc. IEEE ICASSP*, Minneapolis, MN, 1993, pp. 523–526.
[19] M. A. Syed and V. J. Mathews, "Lattice algorithms for recursive least-squares adaptive second-order Volterra filtering," *IEEE Trans. Circuits Syst. II*, vol. 41, pp. 202–214, Mar. 1994.
[20] B. Yang, "A QR multichannel least squares lattice algorithm for adaptive nonlinear filtering," *AEU Int. J. Electron. Commun.*, vol. 49, pp. 171–182, 1995.
[21] J. M. Cioffi, "The fast adaptive ROTOR's RLS algorithm," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 38, pp. 631–653, Apr. 1990.
[22] I. K. Proudler, J. G. McWhirter, and T. J. Shepherd, "Fast QRD-based algorithms for least squares linear prediction," in *Proc. IMA Conf. Math. Signal Process.*, Warwick, U.K., Dec. 12–15, 1988.
[23] M. G. Bellanger, "The FLS-QR algorithm for adaptive filtering," *Signal Process.*, vol. 17, pp. 291–304, 1989.
[24] ———, "A survey of QR based fast least squares adaptive filters: From principles to realization," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Toronto, Ont., Canada, 1991, pp. 1833–1836.
[25] P. A. Regalia and M. G. Bellanger, "On the duality between fast QR methods and lattice methods in least squares adaptive filtering," *IEEE Trans. Signal Processing*, vol. 39, pp. 879–891, Apr. 1991.
[26] A. A. Rontogiannis and S. Theodoridis, "New fast inverse QR least squares adaptive algorithms," in *Proc. IEEE ICASSP*, Detroit, MI, May 1995, pp. 1412–1415.
[27] M. G. Bellanger and P. A. Regalia, "The FLS-QR algorithm for adaptive filtering: The case of multichannel signals," *Signal Process.* vol. 22, pp. 115–126, 1991.
[28] M. A. Syed, "QRD-based fast RLS multichannel adaptive algorithms," *Proc. IEEE ICASSP*, Toronto, Ont., Canada, 1991, pp. 1837–1840.
[29] M. A. Syed and V. J. Mathews, "QR-decomposition based algorithms for adaptive Volterra filtering," *IEEE Trans. Circuits Syst. I*, vol. 40, pp. 372–382, June 1993.
[30] C. T. Pan, R. J. Plemmons, "Least squares modifications with inverse factorizations: Parallel implications," *J. Comput. Appl. Math.*, vol. 27, pp. 109–127, 1989.
[31] S. T. Alexander and A. L. Ghirnikar, "A method for recursive least squares adaptive filtering based upon an inverse QR decomposition," *IEEE Trans. Signal Processing*, vol. 41, pp. 20–30, Jan. 1993.
[32] V. J. Mathews, "Adaptive polynomial filters," *IEEE Signal Processing Mag.*, pp. 10–26, July 1991.
[33] N. Kalouptsidis, G. O. Glentis, and P. Koukoulas, "Efficient algorithms for parametric Volterra system identification," in *Proc. EUSIPCO*, Edinburgh, U.K., 1994.

Athanasios A. Rontogiannis was born in Athens, Greece, on June 16, 1968. He received the diploma in electrical and computer engineering from the National Technical University of Athens in 1991, the M.A.Sc degree in electrical and computer engineering from the University of Victoria, Victoria, B.C., Canada, in 1993, and the Ph.D degree in signal processing from the Department of Informatics, University of Athens, in 1997.

Since March 1997, he has been with the Greek Airforce. From November 1994 to March 1997, he was a recipient of a Scholarship from the State Scholarship Foundation for the completion of his Ph.D. degree. His research interests are focused on adaptive filtering algorithms and their application to channel equalization and echo cancellation schemes.

Sergios Theodoridis was born in Piraeus, Greece, on December 17, 1951. He received the B.Sc. degree with honors in physics from the University of Athens, Athens, Greece, in 1973 and the M.Sc. and Ph.D degrees in communications and signal processing, both from the University of Birmingham, Birmingham, U.K., in 1975 and 1978, respectively.

From 1978 to 1981, he was a Research Fellow with the Department of Electronics and Electrical Engineering, University of Birmingham. From 1981 to 1983, he was with the Greek Army. From 1984 to 1995, he was with the Department of Computer Engineering, University of Patras, Patras, Greece. Since 1995, he has been with the Department of Informatics, University of Athens. His current research interests are focused on the fields of digital signal processing, communications, and pattern recognition.