Multichannel Fast QR-Decomposition Algorithms: Weight Extraction Method and Its Applications

Mobien Shoaib, *Student Member, IEEE*, Stefan Werner, *Senior Member, IEEE*, and José Antonio Apolinário Jr., *Senior Member, IEEE*

***Abstract*—Multichannel fast QR decomposition RLS (MC-FQRD-RLS) algorithms are well known for their good numerical properties and low computational complexity. The main limitation is that they lack an explicit weight vector term, limiting themselves to problems seeking an estimate of the output error signal. This paper presents techniques which allow us to use MC-FQRD-RLS algorithms with applications that previously have required explicit knowledge of the adaptive filter weights. We first consider a multichannel system identification setup and present how to obtain, at any time, the filter weights associated with the MC-FQRD-RLS algorithm. Thereafter, we turn to problems where the filter weights are periodically updated using training data, and then used for fixed filtering of a useful data sequence, e.g., burst-trained equalizers. Finally, we consider a particular control structure, indirect learning, where a copy of the coefficient vector is filtering a different input sequence than that of the adaptive filter. Simulations are carried out for Volterra system identification, decision feedback equalization, and adaptive predistortion of high-power amplifiers. The results verify our claims that the proposed techniques achieve the same performance as the inverse QRD-RLS algorithm at a much lower computational cost.**

***Index Terms*—Adaptive systems, equalizer, fast algorithms, indirect learning, multichannel algorithms, predistortion, QR decomposition, Volterra system identification, weight extraction.**

I. Introduction

FAST QR-DECOMPOSITION RECURSIVE LEAST-SQUARES (FQRD-RLS) algorithms based on backward prediction errors are a popular class of least-squares based algorithms that are known for their numerical stability and reduced computational complexity, unlike fast versions of the RLS algorithm [1]–[3].

The idea in FQRD-RLS algorithms (single or multichannel versions) is to exploit the underlying time-shift structure of the input data vector in order to replace matrix update equations with vector update equations [4]. The time-shift structure of a multichannel input data vector is not necessarily trivial, as it may contain a number of single channel vectors of different orders. The vector update equations are derived from forward and backward predictions. This paper considers multichannel algorithms [4]–[7] based on the update of backward prediction errors which are numerically robust [8]. Note that single channel FQRD-RLS algorithm is a particular case of multichannel FQRD-RLS (MC-FQRD-RLS) algorithms. In this paper our focus is on MC-FQRD-RLS algorithms and its applications, unless mentioned otherwise.

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

The goal of this paper is to extend the range of application of the MC-FQRD-RLS algorithm. We will focus on three different application scenarios:

1) System identification, where knowledge of the explicit weights of the adaptive filter is not needed at each algorithm iteration.
2) Burst-trained systems, where the coefficient adaptation is performed in a training block. The weight vector obtained at the end of the training block is then kept fixed and used for output filtering, e.g., periodically updated channel equalizers.
3) Indirect-learning, where at each time instant a copy of adaptive filter is used for filtering a different input sequence than that of the adaptive filter, e.g., predistortion of high-power amplifiers (HPAs).

For the case of system identification, we propose a mechanism in which the coefficients of the transversal weight vector can be obtained in a sequential manner at any chosen iteration at a total computational complexity cost of $\mathcal{O}(P^2)$, i.e., $\mathcal{O}(P)$ per coefficient, without compromising the accuracy, where P is the total number of filter coefficients. Obviously, if the transversal weight vector is not required at every iteration (typically after convergence), then the overall computational complexity using this approach will be much lower than when using a conventional QRD-RLS algorithm as in [9]. In addition, the peak-complexity (when we have different complexity in distinct iterations, corresponds to the maximum value within a time frame)

Manuscript received September 19, 2008; accepted July 13, 2009. First published August 18, 2009; current version published December 16, 2009. This work was supported in part by the Academy of Finland, Smart and Novel Radios (SMARAD) Center of Excellence, GETA, PSATRI/STC-Chair, CAPES, and by CNPq. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Dennis R. Morgan.

M. Shoaib is with the Department of Signal Processing and Acoustics, Helsinki University of Technology, Finland. He is also with the Prince Sultan Advanced Technology Research Institute (PSATRI), King Saud University, Saudi Arabia (e-mail: mobien@ieee.org).

S. Werner is with the Department of Signal Processing and Acoustics, Helsinki University of Technology, Finland.

J. A. Apolinário, Jr. is with the Military Institute of Engineering, Rio de Janeiro, Brazil.

Digital Object Identifier 10.1109/TSP.2009.2030594

during adaptation is one order of magnitude lower than using a QRD-RLS algorithm. Preliminary results on system identification with single-channel and equal-order block multichannel FQRD-RLS algorithm are reported in [10] and [11].

In the case of burst-trained systems, we show how the internal variables of the MC-FQRD-RLS algorithm related to the implicit weight vector can be frozen at any time instant, and used for fixed filtering. The resulting output signal is exactly the same as if we would have known the MC-FQRD-RLS weights. This *equivalent-output* filtering approach renders an implementation complexity of $\mathcal{O}(P)$ during both *training* and *filtering* phases. In any case, the cost for fixed filtering is still slightly higher than using the known weights in a transversal structure. Thus, increasing the length of the data sequence may yield an overall computational cost that is similar that of an inverse QRD-RLS algorithm [16], which requires P operations for fixed filtering and $\mathcal{O}(P^2)$ for coefficient adaptation. This problem is circumvented by combining our equivalent-filtering results with a *distributed weight extraction* that identifies one weight per each new incoming sample. After all weight have been acquired, filtering is carried out in a transversal structure. Preliminary results on equivalent-output filtering for single-channel FQRD-RLS algorithms can be found in [17].

In the indirect-learning structure [18], a copy of the adaptive filter weights are used for filtering a different input sequence than used for adaptation. This problem shows up in predistortion design for linearizing the effects of HPA amplifiers [19], [20], as well as in active noise control [21]. By using our results on equivalent-output filtering, we show how to obtain an MC-FQRD-RLS predistorter design. To the best of our knowledge, MC-FQRD-RLS algorithms have not been used in predistorter applications. A lattice based variant of FQRD-RLS algorithm was used for active noise control in [21], which used a system-identification mechanism of complexity $\mathcal{O}(P^2)$. To reduce the overall complexity, it was suggested to extract and copy the weights on a periodical basis. Our results show how to avoid such approximate solution and reproduce the exact output at $\mathcal{O}(P)$ cost per iteration.

The paper is organized as follows. Section II reviews the basic concepts of QRD-RLS and MC-FQRD-RLS algorithms and introduces the necessary notations used in the following. We only consider so-called *channel-decomposition* based MC-FQRD-RLS algorithms, where the channels are processed individually and in a sequential manner. Section III shows how to identify the implicit MC-FQRD-RLS weights by extracting the columns of the Cholesky factor embedded in the MC-FQRD-RLS algorithm. From the Cholesky factor, we can obtain the true weights of the underlying least-squares problem by reusing the known MC-FQRD-RLS variables. The problem of fixed filtering appearing in burst-trained equalizers is treated in Section IV. We show how to reproduce the equivalent output signal associated with the true equalizer weights from the MC-FQRD-RLS variables. Section V-B provides details on how to apply MC-FQRD-RLS algorithms for predistorter design based on an indirect-learning architecture. Section VI provides simulation results for Volterra system identification, decision feedback equalization, and Volterra predistortion. Finally, the conclusions are drawn in the Section VII.

II. Channel Decomposition Based Multichannel Multiple Order FQRD-RLS Algorithm

This section introduces the multichannel multiple-order fast QRD-RLS (MC-FQRD-RLS) algorithm [4], [22], [23], to provide the basic set of equations used in the development of new applications in Sections III, IV and V. There are two approaches to implement the multichannel algorithms: a block-type approach, where the different channels are processed simultaneously, and a channel decomposition based approach that processes each channel in a sequential manner. The former renders efficient implementation on parallel architectures. The channel decomposition approach on the other hand, avoids matrix equations and is, as a result, a computationally efficient solution. This paper considers only the channel decomposition-based approach. In the following, we first present the basics of the QR-decomposition based least-squares algorithms. Thereafter, we explain the special structure of the multichannel input vector used in [22]. Finally, the necessary details of the MC-FQRD-RLS are briefly explained.

A. QR Decomposition Basics

The multichannel multiple-order setup comprises of M FIR delay-lines, each of order $N_l - 1$ such that the total number of taps is $P = \sum_{l=1}^{M} N_l$. The cost function used in least-squares algorithms for a multichannel multiple-order input vector is defined as

$$\begin{aligned}\xi(k) &= \sum_{i=0}^{k} \lambda^{k-i} \left| d^*(i) - \sum_{l=1}^{M} \sum_{j=0}^{N_l-1} x_l^*(k-j) w_{l,j}(k) \right|^2 \\ &= \sum_{i=0}^{k} \lambda^{k-i} \left| d^*(i) - \mathbf{x}^{\mathbf{H}}(i)\mathbf{w}(k) \right|^2 = \|\mathbf{e}(k)\|^2 \end{aligned} \quad (1)$$

where for the kth time instant $d(k)$ is the desired input signal, $\mathbf{w}(k) \in \mathbb{C}^{P\times 1}$ is the weight vector sought for, $w_{l,j}(k)$ is the jth coefficient of lth channel, $x_l(k)$ is lth channel input signal, and $\mathbf{e}(k) \in \mathbb{C}^{(k+1)\times 1}$ is the *a posteriori* weighted error vector. The particular arrangements (positions) of the multichannel data $x_l(k)$ in multichannel input vector $\mathbf{x}(k) \in \mathbb{C}^{P\times 1}$ (and correspondingly $w_{l,j}(k)$ in $\mathbf{w}(k)$) is elaborated upon in Section II-B. In this paper, $*$ denotes the complex conjugate, λ is the forgetting factor, and $[\cdot]^{\mathrm{H}}$ denotes the Hermitian of a matrix. Vector $\mathbf{e}(k)$ can also be written in a compact form as

$$\mathbf{e}^*(k) = \mathbf{d}^*(k) - \mathbf{X}(k)\mathbf{w}(k) \quad (2)$$

where $\mathbf{d}(k) \in \mathbb{C}^{(k+1)\times 1}$ and $\mathbf{X}(k) \in \mathbb{C}^{(k+1)\times P}$ are the respective desired signal vector and input data matrix defined as

$$\mathbf{d}(k) = \begin{bmatrix} d(k) \\ \lambda^{1/2} d(k-1) \\ \vdots \\ \lambda^{k/2} d(0) \end{bmatrix} \quad (3)$$

$$\mathbf{X}(k) = \begin{bmatrix} \mathbf{x}^{\mathbf{H}}(k) \\ \lambda^{1/2} \mathbf{x}^{\mathbf{H}}(k-1) \\ \vdots \\ \lambda^{k/2} \mathbf{x}^{\mathbf{H}}(0) \end{bmatrix}. \quad (4)$$

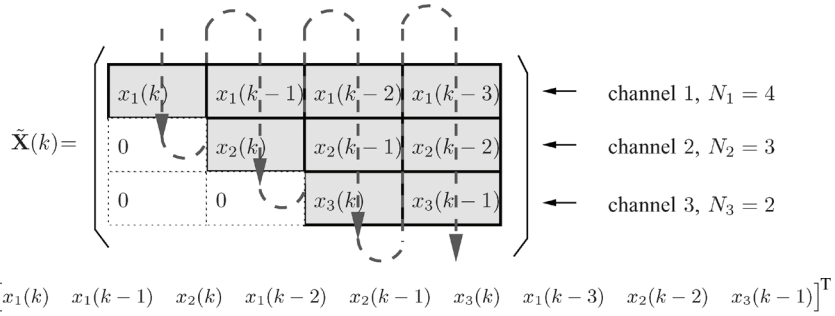

Fig. 1. The construction of the input data delay-line vector $\mathbf{x}(k)$ used in the MC-FQRD-RLS for the case of $M = 3$ channels with $N_1 = 4$, $N_2 = 3$, and $N_3 = 2$ taps.

The QRD-RLS algorithm uses an orthogonal rotation matrix $\mathbf{Q}(k) \in \mathbb{C}^{(k+1)\times(k+1)}$ to triangularize matrix $\mathbf{X}(k)$ [24] as in

$$\begin{bmatrix} \mathbf{0}_{k-P+1\times P} \\ \mathbf{U}(k) \end{bmatrix} = \mathbf{Q}(k)\mathbf{X}(k) \tag{5}$$

where $\mathbf{U}(k) \in \mathbb{C}^{P\times P}$ is the Cholesky factor of the deterministic autocorrelation matrix $\mathbf{R}(k) = \mathbf{X}^{\mathrm{H}}(k)\mathbf{X}(k)$. In this paper we assume that the Cholesky factor is always lower triangular. Premultiplying (2) with $\mathbf{Q}(k)$ gives

$$\mathbf{Q}(k)\mathbf{e}^*(k) = \begin{bmatrix} \mathbf{e}_{q1}(k) \\ \mathbf{e}_{q2}(k) \end{bmatrix} = \begin{bmatrix} \mathbf{d}_{q1}(k) \\ \mathbf{d}_{q2}(k) \end{bmatrix} - \begin{bmatrix} \mathbf{0}_{(k+1-P)\times P} \\ \mathbf{U}(k) \end{bmatrix} \mathbf{w}(k). \tag{6}$$

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

$$\mathbf{w}(k) = \mathbf{U}^{-1}(k)\mathbf{d}_{q2}(k). \tag{7}$$

The QRD-RLS algorithm updates vector $\mathbf{d}_{q2}(k)$ and matrix $\mathbf{U}(k)$ as [13]

$$\begin{bmatrix} \mathbf{0}_{1\times P} \\ \mathbf{U}(k) \end{bmatrix} = \mathbf{Q}_\theta(k) \begin{bmatrix} \mathbf{x}^{\mathbf{H}}(k) \\ \lambda^{1/2}\mathbf{U}(k-1) \end{bmatrix} \tag{8}$$

$$\begin{bmatrix} e_{q1}(k) \\ \mathbf{d}_{q2}(k) \end{bmatrix} = \mathbf{Q}_\theta(k) \begin{bmatrix} d^*(k) \\ \lambda^{1/2}\mathbf{d}_{q2}(k-1) \end{bmatrix} \tag{9}$$

where matrix $\mathbf{Q}_\theta(k) \in \mathbb{C}^{(P+1)\times(P+1)}$ is a sequence of Givens rotation matrices which annihilates the input vector $\mathbf{x}^{\mathrm{H}}(k)$ in (8) and can be partitioned as [7]

$$\mathbf{Q}_\theta(k) = \begin{bmatrix} \gamma(k) & \mathbf{g}^{\mathbf{H}}(k) \\ \mathbf{f}(k) & \mathbf{E}(k) \end{bmatrix} \tag{10}$$

where

$$\begin{aligned} \mathbf{g}(k) &= -\lambda^{-1/2}\gamma(k)\mathbf{U}^{-\mathbf{H}}(k-1)\mathbf{x}(k) = -\lambda^{-1/2}\gamma(k)\bar{\mathbf{a}}(k) \\ \mathbf{f}(k) &= \mathbf{U}^{-\mathbf{H}}(k)\mathbf{x}(k) \\ \mathbf{E}(k) &= \lambda^{1/2}\mathbf{U}^{-\mathbf{H}}(k)\mathbf{U}^{\mathbf{H}}(k-1) \end{aligned} \tag{11}$$

where $\gamma(k)$ is a positive, real-valued scalar [13], $\bar{\mathbf{a}}(k)$ is the *a priori* backward prediction error vector at instant k normalized by the *a posteriori* backward error energies at instant $k-1$, and $\mathbf{f}(k)$ is referred to as the normalized *a posteriori* backward prediction error vector at instant k [8].

B. Construction of $\mathbf{x}(k)$

The particular arrangement of the elements in $\mathbf{x}(k)$ presented in the following is widely used and facilitates the derivation of MC-FQRD-RLS algorithms [4], [9], [22].

We assume without loss of generality that $N_1 \geq N_2 \geq \cdots \geq N_M$. The internal structure of $\mathbf{x}(k)$ is rather intuitive and it is easily constructed by considering the $N_1 \times M$ matrix $\tilde{\mathbf{X}}(k)$ whose ith row contains the N_i input data samples of channel i, i.e., [see (12) at the bottom of the page] where the zero-vectors appearing to the left in each row are of proper size to maintain the dimension of $\tilde{\mathbf{X}}(k)$ (if $N_1 = N_2 = \ldots = N_M$, the zeros will disappear). The input vector $\mathbf{x}(k)$ is obtained by simply stacking the columns of matrix $\tilde{\mathbf{X}}(k)$ and excluding the zeros that were inserted in (12). The position of the most recent sample values of channel l, $x_l(k)$ in vector $\mathbf{x}(k)$ is then given by [4]

$$p_l = \sum_{r=1}^{l-1} r(N_r - N_{r+1}) + l. \tag{13}$$

Fig. 1 illustrates how to construct $\tilde{\mathbf{X}}(k)$ for the case of $N_1 = 4$, $N_2 = 3$, and $N_3 = 2$, i.e., $\tilde{\mathbf{X}}(k)$ becomes a 3×4 matrix. Equation (13) tells us that the positions for the most recent samples of the first, second, and third channel are $p_1 = 1$, $p_2 = 3$, and $p_3 = 6$, respectively.

Note that according to (12), the last M samples of vector $\mathbf{x}(k)$ are $\{x_l(k-N_l)\}_{l=1}^{M}$. As a result, updating the input vector from one time instant to another, i.e., from $\mathbf{x}(k-1)$ to $\mathbf{x}(k)$, becomes particularly simple. Since we know, by construction, that the last

$$\tilde{\mathbf{X}}(k) = \begin{bmatrix} x_1(k) & x_1(k-1) & x_1(k-2) & \ldots & x_1(k-N_1+1) \\ \mathbf{0}_{1\times(N_1-N_2)} & x_2(k) & x_2(k-1) & \ldots & x_2(k-N_2+1) \\ \vdots & & \vdots & & \vdots \\ \mathbf{0}_{1\times(N_1-N_M)} & & x_M(k) & \ldots & x_M(k-N_M+1) \end{bmatrix} \tag{12}$$

M samples of $\mathbf{x}(k-1)$ are to be removed, the update can be performed channel by channel in M sequential steps as

$$\mathbf{\Pi}_l \begin{bmatrix} x_l(k) \\ \mathbf{x}^{(l-1)}(k-1) \end{bmatrix} = \begin{bmatrix} \mathbf{x}^{(l)}(k-1) \\ x_l(k-N_l) \end{bmatrix}, \quad l = 1, \ldots, M \quad (14)$$

where $\mathbf{x}^{(l)}(k-1)$ corresponds to the vector which has been updated with input samples up to channel l, and $\mathbf{\Pi}_l$ is a permutation matrix that takes the most recent sample $x_l(k)$ and places it at the p_lth position of vector $\mathbf{x}^{(l-1)}(k-1)$. After processing all the M channels, $\mathbf{x}^{(M)}(k-1)$ will constitute the input vector at iteration k, i.e.

$$\mathbf{x}(k-1) \rightarrow \mathbf{x}^{(1)}(k-1) \rightarrow \cdots \rightarrow \mathbf{x}^{(M)}(k-1) = \mathbf{x}(k). \quad (15)$$

Next we describe the main steps and the key equations of the MC-FQRD-RLS algorithm [22].

C. Multichannel Decomposition Algorithm

Let us define the input data matrix of the lth sequential step in (15) as

$$\mathbf{X}^{(l)}(k-1) = \begin{bmatrix} \mathbf{x}^{\mathbf{H}(l)}(k-1) \\ \lambda^{1/2}\mathbf{x}^{\mathbf{H}(l)}(k-2) \\ \vdots \\ \lambda^{k/2}\mathbf{x}^{\mathbf{H}(l)}(-1) \end{bmatrix}. \quad (16)$$

Similar to (5) we can obtain the corresponding Cholesky factor

$$\begin{bmatrix} \mathbf{0}_{(k-P+1)\times P} \\ \mathbf{U}^{(l)}(k-1) \end{bmatrix} = \mathbf{Q}^{(l)}(k)\mathbf{X}^{(l)}(k-1). \quad (17)$$

The update equation for the Cholesky factor given in (8) can also be applied to (17), as a result we have

$$\begin{bmatrix} \mathbf{0}_{1\times P} \\ \mathbf{U}^{(l)}(k-1) \end{bmatrix} = \mathbf{Q}_\theta^{(l)}(k) \begin{bmatrix} \mathbf{x}^{\mathbf{H}(l)}(k-1) \\ \lambda^{1/2}\mathbf{U}^{(l-1)}(k-1) \end{bmatrix}. \quad (18)$$

MC-FQRD-RLS algorithms exploit the shift structure of the input vector, and replace the matrix update equation similar to (8) with vector updates of either $\mathbf{f}(k)$ or $\mathbf{g}(k)$ in (11). We will here adopt the algorithm based on *a posteriori* errors in [22] that updates vector $\mathbf{f}(k)$. Our results in Sections III–V are easily adapted to other algorithms, e.g., based on *a priori* errors [22]. In the MC-FQRD-RLS algorithm, vector $\mathbf{f}(k)$ is updated in M successive steps [4], i.e.

$$\mathbf{f}(k-1) \rightarrow \mathbf{f}^{(1)}(k-1) \rightarrow \cdots \rightarrow \mathbf{f}^{(M)}(k-1) = \mathbf{f}(k) \quad (19)$$

where

$$\mathbf{f}^{(l)}(k-1) = \mathbf{U}^{-\mathbf{H}(l)}(k-1)\mathbf{x}^{(l)}(k-1). \quad (20)$$

The update equation for vector $\mathbf{f}^{(l-1)}(k-1)$ is given as

$$\begin{bmatrix} \frac{\varepsilon_b^{(l)}(k)}{\|\mathbf{e}_b^{(l)}(k)\|} \\ \mathbf{f}^{(l)}(k-1) \end{bmatrix} = \mathbf{\Pi}_l \mathbf{Q}_{f\theta}^{(l)}(k) \begin{bmatrix} \mathbf{f}^{(l-1)}(k-1) \\ \frac{\varepsilon_f^{(l)}(k)}{\|\mathbf{e}_f^{(l)}(k)\|} \end{bmatrix} \quad (21)$$

where $\varepsilon_b^{(l)}(k)$ and $\varepsilon_f^{(l)}(k)$ are the *a posteriori* backward and forward prediction errors, $\|\mathbf{e}_b^{(l)}(k)\|$ and $\|\mathbf{e}_f^{(l)}(k)\|$ are the norms of the backward and forward prediction error vectors, and $\mathbf{\Pi}_l$ is the permutation matrix in (14). Appendix I outlines the derivation for the update of $\mathbf{f}(k)$ in (21) and provides the definitions related to the forward and backward prediction variables used in the MC-FQRD-RLS algorithm. The resulting MC-FQRD-RLS algorithm is summarized in Table I. See [4] and [22] for more details on other versions.

TABLE I
THE MCFQRD-RLS ALGORITHM

Multiple order sequential MC-FQR_PRI_B algorithm [22]

Initializations:
$\mathbf{d}_{fq2}^{(l)}(0) = \mathbf{0}_{P\times 1}$
$\mathbf{f}^{(l)}(0) = \mathbf{0}_{P\times 1}$
$\mathbf{d}_{q2}(0) = \mathbf{0}_{P\times 1}$
$\gamma^{(0)}(k) = 1$; $\|\mathbf{e}_f^{(l)}(0)\| = \mu$ (a small constant)
$\mathbf{Q}_\theta^{(0)}(k) = \mathbf{I}_{P+1\times P+1}$
$\mathbf{Q}_{\theta f}^{(1)}(k) = \mathbf{I}_{P+1\times P+1}$
for each k
{
$\mathbf{Q}_{\theta f}^{(1)}(k) = \mathbf{Q}_{\theta f}^{(M)}(k-1)$
$\mathbf{Q}_\theta^{(0)}(k) = \mathbf{Q}_\theta(k-1) = \mathbf{Q}_{\theta f}^{(M)}(k-1)$
$\mathbf{f}^{(0)}(k) = \mathbf{f}^{(M)}(k-1)$
for each l from 1 to M
{
Compute $\mathbf{d}_{fq2}^{(l)}(k)$:
$e_{fq1}^{(l)}(k) = x_l(k)$
$\begin{bmatrix} e_{fq1}^{(l)}(k) \\ \mathbf{d}_{fq2}^{(l)}(k) \end{bmatrix} = \mathbf{Q}_\theta^{(l-1)}(k-1) \begin{bmatrix} x_l(k) \\ \lambda^{1/2}\mathbf{d}_{fq2}^{(l)}(k-1) \end{bmatrix}$
Compute $\|\mathbf{e}_f^{(l)}(k)\|$:
$\|\mathbf{e}_f^{(l)}(k)\| = \sqrt{(e_{fq1}^{(l)}(k))^2 + \lambda\|\mathbf{e}_f^{(l)}(k-1)\|^2}$
Compute $\mathbf{Q}_{\theta f}^{(l)}(k)$:
$\begin{bmatrix} \mathbf{0}_{P\times 1} \\ \|\mathbf{e}_{f0}^{(l)}(k)\| \end{bmatrix} = \mathbf{Q}_{\theta f}^{(l)}(k) \begin{bmatrix} \mathbf{d}_{fq2}^{(l)}(k) \\ \|\mathbf{e}_f^{(l)}(k)\| \end{bmatrix}$
Compute $\mathbf{f}^{(l)}(k)$:
$e_f^{(l)}(k) = \gamma^{(l-1)}(k)e_{fq1}^{(l)}(k)$
$\begin{bmatrix} \frac{e_b^{(l)}(k)}{\|\mathbf{e}_b^{(l)}(k-1)\|} \\ \mathbf{f}^{(l)}(k-1) \end{bmatrix} = \mathbf{\Pi}_l \mathbf{Q}_{\theta f}^{(l)}(k) \begin{bmatrix} \mathbf{f}^{(l-1)}(k-1) \\ \frac{e_f^{(l)}(k)}{\|\mathbf{e}_f^{(l)}(k-1)\|} \end{bmatrix}$
Compute $\mathbf{Q}_\theta^{(l)}(k)$:
$\begin{bmatrix} 1 \\ \mathbf{0}_{P\times 1} \end{bmatrix} = \mathbf{Q}_\theta^{\mathrm{H}(l)}(k) \begin{bmatrix} \gamma^{(l)}(k) \\ \mathbf{f}^{(l)}(k-1) \end{bmatrix}$
}
Joint Process Estimation:
$\begin{bmatrix} e_{q1}(k) \\ \mathbf{d}_{q2}(k) \end{bmatrix} = \mathbf{Q}_\theta(k) \begin{bmatrix} d(k) \\ \lambda^{1/2}\mathbf{d}_{q2}(k-1) \end{bmatrix}$
$e(k) = e_{q1}^*(k)/\gamma^{(M)}(k)$
}

III. MULTICHANNEL SYSTEM IDENTIFICATION

Considering the MC-FQRD-RLS algorithm in Table I, this section provides the details on how to identify (or extract) the transversal weight coefficients embedded in its internal variables. The *weight extraction technique* presented can be invoked at any iteration of the MC-FQRD-RLS algorithm, and consists of P sequential steps, i.e., one for each of the P coefficients. The weight extraction can also be implemented using a parallel architecture, i.e., M weight extraction procedures are applied simultaneously for M channels, each containing N_l sequential steps.

From (7) we see that the transversal weight vector in QRD-RLS based algorithms is given by the inverse Cholesky

factor matrix $\mathbf{U}^{-1}(k)$ and the rotated desired signal vector $\mathbf{d}_{q2}(k)$. More specifically, each element, $w_{l,j}(k)$, of vector $\mathbf{w}(k)$ is associated with one of the column vectors, $\mathbf{u}_{l,j}(k)$, in $\mathbf{U}^{-\mathrm{H}}(k)$

$$w_{l,j}(k) = \mathbf{u}_{l,j}^{\mathbf{H}}(k)\mathbf{d}_{q2}(k) \tag{22}$$

where the l, j are the index values corresponding to $x_l(k-j+1)$ in $\mathbf{x}(k)$, see, e.g., Fig 1. This means that, when $\mathbf{d}_{q2}(k)$ is given, the elements of vector $\mathbf{w}(k)$ can be computed if all the columns of the matrix $\mathbf{U}^{-\mathrm{H}}(k)$ are known. The two lemmas stated below enable us to obtain all the column vectors $\mathbf{u}_{l,j}(k)$ in a serial manner. The main result is that column vector $\mathbf{u}_{l,j+1}(k)$ can be obtained from the column vector $\mathbf{u}_{l,j}(k-1)$.

Lemma 1: Let $\mathbf{u}_{l,j}(k) \in \mathbb{C}^{P\times 1}$ be the column of $\mathbf{U}^{-\mathrm{H}}(k) \in \mathbb{C}^{P\times P}$ in (5) corresponding to coefficient j of channel l. Given $\mathbf{Q}_\theta(k) \in \mathbb{C}^{(P+1)\times(P+1)}$ from Table I, then $\mathbf{u}_{l,j}(k-1)$ is obtained from $\mathbf{u}_{l,j}(k)$ using

$$\begin{bmatrix} 0 \\ \lambda^{-1/2}\mathbf{u}_{l,j}(k-1) \end{bmatrix} = \mathbf{Q}_\theta^{\mathbf{H}}(k)\begin{bmatrix} z_j(k) \\ \mathbf{u}_{l,j}(k) \end{bmatrix} \tag{23}$$

where $z_j(k) = -\mathbf{f}^{\mathrm{H}}(k)\mathbf{u}_{l,j}(k)/\gamma^{(M)}(k)$, and $j = 0, \ldots, N_i - 1$, and $l = 1, \ldots, M$.

Lemma 2: Let $\mathbf{u}_{l,j}(k) \in \mathbb{C}^{P\times 1}$ be the column of $\mathbf{U}^{-\mathrm{H}}(k) \in \mathbb{C}^{P\times P}$ in (5) corresponding to coefficient j of channel l. Then $\mathbf{u}_{l,j+1}(k)$ is obtained from $\mathbf{u}_{l,j}(k-1)$ in M sequential steps

$$\mathbf{u}_{l,j}(k-1) \rightarrow \mathbf{u}_{l,j}^{(1)}(k-1) \rightarrow \cdots \rightarrow \mathbf{u}_{l,j}^{(M)}(k-1) = \mathbf{u}_{l,j+1}(k)$$

where step $\mathbf{u}_{l,j}^{(i-1)}(k-1) \rightarrow \mathbf{u}_{l,j}^{(i)}(k-1)$ is given by

$$\begin{bmatrix} \frac{-w_{b,j}^{(i)}(k-1)}{\|\mathbf{e}_b^{(i)}(k-1)\|} \\ \mathbf{u}_{l,j}^{(i)}(k-1) \end{bmatrix} = \mathbf{\Pi}_i\mathbf{Q}_{\theta f}^{(i)}(k)\begin{bmatrix} \mathbf{u}_{l,j}^{(i-1)}(k-1) \\ \frac{-w_{f,j}^{(i)}(k-1)}{\|\mathbf{e}_f^{(i)}(k-1)\|} \end{bmatrix}, \quad i=1,\ldots,M. \tag{24}$$

with $\mathbf{Q}_{\theta f}^{(i)}(k) \in \mathbb{C}^{(P+1)\times(P+1)}$ and $\mathbf{\Pi}_i \in \mathbb{C}^{(P+1)\times(P+1)}$ in (21), and

$$w_{f,j}^{(i)}(k-1) = \begin{cases} 0 & \text{for } j < l-1 \\ -1 & \text{for } j = l-1 \\ -\mathbf{u}_{l,j}^{\mathbf{H}(i-1)}(k-1)\mathbf{d}_{fq2}^{(i)}(k) & \text{otherwise.} \end{cases} \tag{25}$$

The recursion in (24), Appendix I-A, is performed to $i = 1,\ldots,M$, $j = 0,\ldots,N_l - 1$, and $l = 1,\ldots,M$, and initialized with $\mathbf{u}_{l,j}^{(0)}(k-1) = \mathbf{u}_{l,j-1}(k-1)$ and $\mathbf{u}_{l,-1}^{(0)}(k-1) = \mathbf{0}_{P\times 1}$.

The proofs of Lemma 1 and Lemma 2 are provided in Appendix II.

In order to extract the implicit weights $\mathbf{w}(k)$ associated with each channel, Lemma 2 is initialized with $\mathbf{u}_{l,0}^{(0)}(k-1)$, and, as a result, we get column vector $\mathbf{u}_{l,1}(k)$. Lemma 1 is then invoked to get the column vector $\mathbf{u}_{l,1}(k-1)$. From $\mathbf{u}_{l,1}(k-1)$ we can compute $\mathbf{u}_{l,2}(k)$ using Lemma 2. By induction we can conclude that all $\mathbf{u}_{l,j}(k)$ can be obtained. As a consequence, the elements of $\mathbf{w}(k)$ can be obtained from (22) in a serial manner. The column extraction procedure is illustrated in Fig. 2 for the example in Section II-B. The weight extraction algorithm is summarized in Table II.

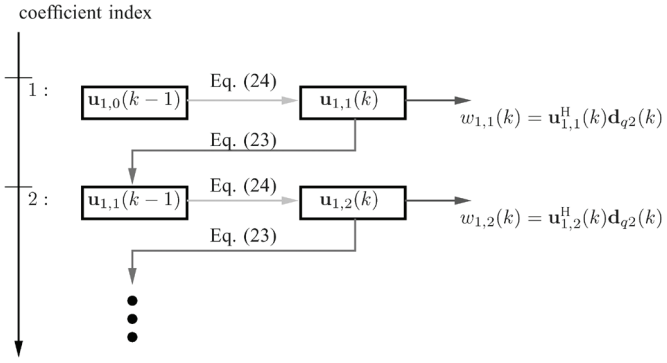

Fig. 2. The extraction of the first and second coefficient of channel 1.

TABLE II
WEIGHT EXTRACTION ALGORITHM

Weight extraction at any chosen time instant k

Available from the MC-FQRD-RLS algorithm:
$\mathbf{f}(k), \gamma^{(M)}(k), \mathbf{Q}_\theta(k), \mathbf{d}_{fq2}^{(i)}(k)$,
$\|\mathbf{e}_f^{(i)}(k-1)\|, \mathbf{Q}_{\theta f}^{(i)}(k)$ and $\mathbf{d}_{q2}(k)$, $i = 1,\ldots,M$

for each $l = 1 : M$
{
  Initialization:
  $x_i = 0, \forall i \epsilon \{1, N_l\}$
  $x_l = 1$
  $\mathbf{u}_{l,-1}(k-1) = \mathbf{0}_{P\times 1}$

  Compute $\mathbf{u}_{l,j+1}(k-1)$ from $\mathbf{u}_{l,j+1}(k)$:
  $z_j(k) = -\mathbf{f}^{\mathrm{H}}(k)\mathbf{u}_{l,j+1}(k)/\gamma^{(M)}(k)$
  $\begin{bmatrix} 0 \\ \lambda^{-1/2}\mathbf{u}_{l,j+1}(k-1) \end{bmatrix} = \mathbf{Q}_\theta^{\mathrm{H}}(k)\begin{bmatrix} z_j(k) \\ \mathbf{u}_{l,j+1}(k) \end{bmatrix}$
  Compute $\mathbf{u}_{l,j+1}(k)$ from $\mathbf{u}_{l,j}(k-1)$:
  for each $j = 0 : N_l - 1$
  {
    $\mathbf{u}_{l,j}^{(0)}(k-1) = \mathbf{u}_{l,j-1}(k-1)$
    for each $i = 1 : M$
    {
      $w_{f,j}^{(i)}(k-1) = x_i - [\mathbf{u}_{l,j-1}^{(i-1)}(k-1)]^{\mathrm{H}}\mathbf{d}_{fq2}^{(i)}(k)$
      $\begin{bmatrix} \frac{-w_{b,j}^{(i)}(k-1)}{\|\mathbf{e}_b^{(i)}(k-1)\|} \\ \mathbf{u}_{l,j}^{(i)}(k-1) \end{bmatrix} = \mathbf{\Pi}_i\mathbf{Q}_{\theta f}^{(i)}(k)\begin{bmatrix} \mathbf{u}_{l,j}^{(i-1)}(k-1) \\ \frac{-w_{f,j}^{(i)}(k-1)}{\|\mathbf{e}_f^{(i)}(k-1)\|} \end{bmatrix}$
    }

    $\mathbf{u}_{l,j+1}(k) = \mathbf{u}_{l,j-1}^{(M)}(k-1)$
    Compute $w_{l,j+1}(k)$:
    $w_{l,j+1}(k) = \mathbf{u}_{l,j+1}^{\mathrm{H}}(k)\mathbf{d}_{q2}(k)$

  }
}

The computational complexity of the weight extraction is given in Table IV.

Remark 1: From Table II it is noted that the weight extraction algorithm can be applied to each channel simultaneously. Thus the coefficients of each channel are extracted in N_i sequential steps, where $i \in [1 \cdots M]$. M such parallel realizations allow extraction of all the P coefficients.

Remark 2: Using Lemma 1 and 2 the Cholesky factor matrix $\mathbf{U}^{-\mathrm{H}}(0)$ corresponding to the initial conditions of the MC-FQRD-RLS algorithm can be obtained and used in the

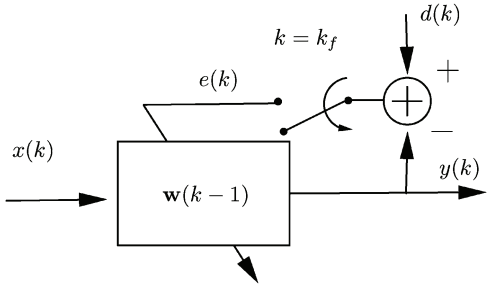

Fig. 3. Burst-trained adaptive filter setup. The adaptive filter is in *training mode* for $k \leq k_f$, and in *data mode* for $k > k_f$. Note that there is no weight update after $k > k_f$.

IQRD-RLS algorithm. This initialization renders precisely the same learning curves for both algorithms.

Remark 3: The proposed method utilizes the internal variables with the aid of Givens rotation matrices to compute the columns of $\mathbf{U}^{-\mathrm{H}}(k)$. It is known, for the case of updating backward prediction errors (resulting in lower triangular Cholesky factor, as in the notation adopted in this paper) that the rows of matrix $\mathbf{U}^{-H}(k)$ are actually the coefficients of the backward prediction filter of all orders (0 to N) [12]. The weight extraction method based on computing the backward prediction filter coefficients has been previously detailed in [13]. This method uses the Levinson-Durbin recursions and the *alternative Kalman gain vector* to extract the transversal weight vector from the lattice coefficients which according to [14] requires $O(N^2)$ computations. Recently, another method was presented in [15] that uses the Levinson-Durbin recursions coupled with Cholesky factor update equation to extract the transversal weights from the lattice coefficients.

IV. Burst-Trained Equalizers

This section shows how to apply MC-FQRD-RLS algorithms to problems where the filter coefficients are periodically updated and then used for fixed filtering of a useful data sequence, e.g., burst-trained equalizers.

A. System Description

The problem under consideration is illustrated in Fig. 3. As can be seen from the figure, the adaptive filter for time instants $k \leq k_f$ is updated using its input and desired signal pair $\{\mathbf{x}(k), d(k)\}$; we call it *training mode*. At time instant $k = k_f$, the adaptive process is stopped and from there onward the coefficient vector at hand $\mathbf{w}(k_f)$ is frozen and used for filtering; we call it *data mode*.

Such scenario can occur, for example, in periodic training where the adaptive filter weights are not updated at every iteration but after a certain data block. So, the adaptive filter acts as an ordinary filter for the data block. As an example, consider an equalizer design in a GSM environment, where the blocks of training symbols are located within the data stream, and the estimation process is only carried out when training symbols are encountered. The output of the filter is given by

$$y(k) = \begin{cases} \mathbf{w}^{\mathbf{H}}(k-1)\mathbf{x}(k) & k \leq k_f \\ \mathbf{w}^{\mathbf{H}}(k_f)\mathbf{x}(k) & k > k_f \end{cases} \tag{26}$$

TABLE III
Equivalent-Output Filtering Using Approach 1 Algorithm

Output-filtering after any chosen time instant K

Available from the MC-FQRD-RLS algorithm:
$\mathbf{f}(k_f), \gamma^{(M)}(k_f), \mathbf{Q}_\theta(k_f), \mathbf{d}_{fq2}^{(i)}(k_f)$,
$\|\mathbf{e}_f^{(i)}(k_f)\|, \mathbf{Q}_{\theta f}^{(i)}(k_f)$ and $\mathbf{d}_{q2}(k_f)$, $i = 1, \ldots, M$

Initialization:
$\mathbf{r}(k) = \mathbf{U}^{-\mathrm{H}}(k_f)\mathbf{x}(k)$
$\tilde{\mathbf{r}}(k) = \mathbf{U}^{-\mathrm{H}}(k_f - 1)\mathbf{x}(k)$

for each $k > K$
{
  Compute $\tilde{\mathbf{r}}(k)$ from $\mathbf{r}(k)$:
  $\tilde{z}(k) = -\mathbf{f}^{\mathrm{H}}(k_f)\tilde{\mathbf{r}}(k)/\gamma^{(M)}(k_f)$
  $\begin{bmatrix} 0 \\ \lambda^{-1/2}\tilde{\mathbf{r}}(k) \end{bmatrix} = \mathbf{Q}_\theta^{\mathrm{H}}(k_f) \begin{bmatrix} \tilde{z}(k) \\ \tilde{\mathbf{r}}(k) \end{bmatrix}$

  Compute $\mathbf{r}(k)$ from $\tilde{\mathbf{r}}(k)$:
  $\mathbf{r}^{(0)}(k-1) = \tilde{\mathbf{r}}(k)$
  for each $i = 1 : M$
  {
    $\varepsilon_f^{(i)}(k) = x_i(k) - \mathbf{r}^{\mathrm{H}(i-1)}(k-1)\mathbf{d}_{fq2}^{(i)}(k_f)$
    $\begin{bmatrix} \frac{\varepsilon_b^{(i)}(k)}{\|\mathbf{e}_b^{(i)}(k_f)\|} \\ \mathbf{r}^{(i)}(k-1) \end{bmatrix} = \mathbf{\Pi}_i \mathbf{Q}_{\theta f}^{(i)}(k_f) \begin{bmatrix} \mathbf{r}^{(i-1)}(k-1) \\ \frac{\varepsilon_f^{(i)}(k)}{\|\mathbf{e}_f^{(i)}(k_f)\|} \end{bmatrix}$
  }
  $\tilde{\mathbf{r}}(k) = \mathbf{r}^{(M)}(k-1)$

  Compute $y^*(k) = \mathbf{x}^{\mathrm{H}}(k)\mathbf{w}(k_f)$:
  $y^*(k) = \mathbf{d}_{q2}^{\mathrm{H}}(k_f)\tilde{\mathbf{r}}(k)$
}

where $\mathbf{w}(k_f)$ is the coefficient vector of the adaptive filter "frozen" at time instant $k = k_f$. This approach of output filtering is not realizable with the MC-FQRD-RLS algorithm.

If the MC-FQRD-RLS algorithm is employed during training, one alternative for carrying out the filtering during data mode is to first extract the filter coefficients according to Section III and, thereafter, perform the filtering of $\mathbf{x}(k)$ with a simple transversal structure. To avoid the increased peak complexity of this operation, we can instead reuse the variables from the MC-FQRD-RLS update at time instant $k = k_f$ to reproduce the equivalent output signal without the need of explicitly extracting the weights $\mathbf{w}(k_f)$, at $k = k_f$. For the single channel FQRD-RLS algorithm, the equivalent output-filtering idea was presented in [17].

B. Approach I: Equivalent-Output Filtering Without Explicit Weight Extraction

This section describes an output filtering method that does not require explicit knowledge of the MC-FQRD-RLS weight vector.

The variables from the MC-FQRD-RLS update at $k = k_f$ can be reused to reproduce the equivalent-output signal associated with $\mathbf{w}(k_f)$ and $\mathbf{x}(k)$. Using the definition of the weight vector in (7), the filter output in *data mode* given by (26) can be rewritten as

$$y(k) = \mathbf{d}_{q2}^{\mathbf{H}}(k_f) \underbrace{\mathbf{U}^{-\mathbf{H}}(k_f)\mathbf{x}(k)}_{\tilde{\mathbf{r}}(k)} \tag{27}$$

TABLE IV
COMPUTATIONAL COMPLEXITY OF WEIGHT EXTRACTION (WE), EQUIVALENT OUTPUT-FILTERING (EF) AND THE INDIRECT LEARNING MECHANISMS (IL)

Weight Extraction	MULT	DIV	SQRT
MC-FQRD-RLS	$14PM+13M+5P-9\sum_{i=1}^{M}p_i$	$3PM+4M+3\sum_{i=1}^{M}p_i$	$2PM+3M-2\sum_{i=0}^{M}p_i$
WE (per coeff j)	$5(P-j)M+7(P-j)-\sum_{i=1}^{M}p_i$	0	0
WE (total)	$\sum_{l=1}^{M}\sum_{j=0}^{N_l}(5(P-j)M+7(P-j)-\sum_{i=1}^{M}p_i)$	0	0
Output Filtering			
EF + WE (per coeff, for $k<P$)	$3\times 5(P-j)M+7(P-j)-\sum_{i=1}^{M}p_i$	0	0
EF (for $k>P$)	P	0	0
Indirect learning			
IL (each iteration)	$5(P-j)M-\sum_{i=1}^{M}p_i$	0	0

where $\mathbf{d}_{q2}(k_f)$ is explicitly available in the FQRD-RLS algorithm at time instant $k=k_f$. However, knowledge of $\mathbf{U}^{-1}(k_f)$ is only provided through the variable $\tilde{\mathbf{r}}(k)=\mathbf{U}^{-\mathrm{H}}(k_f)\mathbf{x}(k_f)$. Thus, starting with $\bar{\mathbf{r}}(k)$ as an initial value, we need to find a way to obtain vector $\mathbf{U}^{-\mathrm{H}}(k_f)\mathbf{x}(k)$ from vector $\mathbf{U}^{-\mathrm{H}}(k_f)\mathbf{x}(k-1)$, i.e., we need to incorporate the new sample $x(k)$ without affecting $\mathbf{U}^{-\mathrm{H}}(k_f)$ in the process. The solution exploits the following two relations:

$$\mathbf{U}^{-\mathbf{H}}(k_f)\mathbf{x}(k-1)\rightarrow\mathbf{U}^{-\mathbf{H}}(k_f-1)\mathbf{x}(k-1)\rightarrow\mathbf{U}^{-\mathbf{H}}(k_f)\mathbf{x}(k). \tag{28}$$

Lemmas 3 and 4 summarize the steps required to compute (27) without explicitly computing $\mathbf{U}^{-\mathrm{H}}(k_f)$.

Lemma 3: Let $\mathbf{U}^{-\mathrm{H}}(k)\in\mathbb{C}^{P\times P}$ denote the upper triangular inverse Hermitian Cholesky matrix, and $\mathbf{x}(k)\in\mathbb{C}^{P\times 1}$ denote the input data vector. Given $\mathbf{Q}_\theta(k)\in\mathbb{C}^{(P+1)\times(P+1)}$ from Table I, then $\bar{\mathbf{a}}(k+l)=\mathbf{U}^{-\mathrm{H}}(k-1)\mathbf{x}(k+l)$ is obtained from $\hat{\mathbf{a}}(k+l)=\mathbf{U}^{-\mathrm{H}}(k)\mathbf{x}(k+l)$ using the relation

$$\begin{bmatrix}0\\ \lambda^{-1/2}\bar{\mathbf{a}}(k+l)\end{bmatrix}=\mathbf{Q}_\theta^{\mathbf{H}}(k)\begin{bmatrix}\tilde{z}(k+l)\\ \hat{\mathbf{a}}(k+l)\end{bmatrix} \tag{29}$$

where $\tilde{z}(k+l)=-\mathbf{f}^{\mathrm{H}}(k)\bar{\mathbf{a}}(k+l)/\gamma^{(M)}(k)$, and $l\geq 1$. The proof for Lemma 3 is available in Appendix II.

Lemma 4: Given $\mathbf{Q}_{\theta f}^{(i)}(k)\in\mathbb{C}^{(P+1)\times(P+1)}$ and $\mathbf{\Pi}_i\in\mathbb{C}^{(P+1)\times(P+1)}$ in (21), then $\hat{\mathbf{a}}(k+l)=\mathbf{U}^{-\mathrm{H}}(k)\mathbf{x}(k+l)$ is obtained from $\bar{\mathbf{a}}(k+l-1)=\mathbf{U}^{-\mathrm{H}}(k-1)\mathbf{x}(k+l-1)$ in M sequential steps

$$\bar{\mathbf{a}}(k+l-1)\rightarrow\bar{\mathbf{a}}^{(1)}(k+l-1)\rightarrow\cdots\rightarrow\bar{\mathbf{a}}^{(M)}(k+l-1)=\hat{\mathbf{a}}(k+l)$$

where $l\geq 0$, and step $\bar{\mathbf{a}}^{(i-1)}(k+l-1)\rightarrow\bar{\mathbf{a}}^{(i)}(k+l-1)$ is given by

$$\begin{bmatrix}\frac{\varepsilon_b^{(i)}(k+l)}{\left\|\mathbf{e}_b^{(i)}(k)\right\|}\\ \bar{\mathbf{a}}^{(i)}(k+l-1)\end{bmatrix}=\mathbf{\Pi}_i\mathbf{Q}_{\theta f}^{(i)}(k)\begin{bmatrix}\bar{\mathbf{a}}^{(i-1)}(k+l-1)\\ \frac{\varepsilon_f^{(i)}(k+l)}{\left\|\mathbf{e}_f^{(i)}(k)\right\|}\end{bmatrix}\quad i=1,\ldots,M \tag{30}$$

and

$$\varepsilon_f^{(i)}(k+l)=x_i(k+l)-\bar{\mathbf{a}}^{\mathbf{H}(i-1)}(k+l-1)\mathbf{d}_{fq2}^{(i)}(k). \tag{31}$$

The proof for Lemma 4 is available in Appendix II. In order to apply the above results for the filtering problem in Fig. 3, we substitute $\mathbf{U}^{-\mathrm{H}}(k-1)$ with $\mathbf{U}^{-\mathrm{H}}(k_f-1)$ in Lemma 3 and 4 and set $l=0$ such that $\mathbf{r}(k)=\mathbf{U}^{-\mathrm{H}}(k_f-1)\mathbf{x}(k)$ and $\tilde{\mathbf{r}}(k)=\mathbf{U}^{-\mathrm{H}}(k_f)\mathbf{x}(k)$. Therefore starting from $\mathbf{r}(k-1)$ we first obtain $\tilde{\mathbf{r}}(k)$ in M sequential steps using Lemma 4

$$\begin{bmatrix}\frac{\varepsilon_b^{(i)}(k)}{\left\|\mathbf{e}_b^{(i)}(k)\right\|}\\ \mathbf{r}^{(i)}(k-1)\end{bmatrix}=\mathbf{\Pi}_i\mathbf{Q}_{\theta f}^{(i)}(k)\begin{bmatrix}\mathbf{r}^{(i-1)}(k-1)\\ \frac{\varepsilon_f^{(i)}(k)}{\left\|\mathbf{e}_f^{(i)}(k)\right\|}\end{bmatrix}\quad i=1,\ldots,M. \tag{32}$$

The updated vector $\tilde{\mathbf{r}}(k)$ is finally obtained from $\tilde{\mathbf{r}}(k)=\mathbf{r}^{(M)}(k-1)$ in (32). The vector $\tilde{\mathbf{r}}(k)$ is then used in (27) to produce $y(k)$. In order to get $\tilde{\mathbf{r}}(k+1)$ using Lemma 4 we first need to compute $\mathbf{r}(k)$ by applying Lemma 3, i.e.

$$\begin{bmatrix}0\\ \lambda^{-1/2}\mathbf{r}(k)\end{bmatrix}=\mathbf{Q}_\theta^{\mathrm{T}}(k)\begin{bmatrix}\tilde{z}(k)\\ \tilde{\mathbf{r}}(k)\end{bmatrix} \tag{33}$$

where $\tilde{z}(k)=\mathbf{f}^{\mathrm{H}}(k)\tilde{\mathbf{r}}(k)/\gamma^{(M)}(k)$. By induction, the output $y(k)$ for all values of k can therefore be obtained with coefficients of the MC-FQRD-RLS frozen at time instant k_f. Note that when $\tilde{\mathbf{r}}(k)$ is obtained from $\tilde{\mathbf{r}}(k-1)$, only the input vector is updated and matrix $\mathbf{U}^{-\mathrm{H}}(k_f)$ remains the same in the process. The steps of the equivalent-output filtering algorithm and its computational complexity are provided in Tables III and IV, respectively.

C. Approach II: Equivalent-Output Filtering With Distributed Weight Extraction

We can divide the input vector $\mathbf{x}(k)$ into two nonoverlapping vectors $\mathbf{c}(k)\in\mathbb{C}^{P\times 1}$ and $\mathbf{v}(k)\in\mathbb{C}^{P\times 1}$

$$\mathbf{x}(k)=\mathbf{c}(k)+\mathbf{v}(k),\quad k>k_f \tag{34}$$

with

$$\begin{aligned}\mathbf{c}(k_f)&=\mathbf{0}\\ \mathbf{v}(k_f)&=\mathbf{x}(k_f)\end{aligned} \tag{35}$$

where $\mathbf{c}(k)$ contains the input-samples from $k>k_f$ and $\mathbf{v}(k)$ holds those remaining from $k\leq k_f$. In other words, for each time instant k, the M new input-samples are shifted into $\mathbf{c}(k)$ and M zeros are shifted into $\mathbf{v}(k)$. This update process, which is similar to (14), (15), is given by

$$\begin{aligned}\mathbf{\Pi}_l\begin{bmatrix}x_l(k)\\ \mathbf{c}^{(l-1)}(k-1)\end{bmatrix}&=\begin{bmatrix}\mathbf{c}^{(l)}(k-1)\\ x_l(k-N_l)\end{bmatrix}\\ \mathbf{\Pi}_l\begin{bmatrix}0\\ \mathbf{v}^{(l-1)}(k-1)\end{bmatrix}&=\begin{bmatrix}\mathbf{v}^{(l)}(k-1)\\ v_l(k-N_l)\end{bmatrix},\ l=1,\ldots,M\end{aligned} \tag{36}$$

with $\mathbf{c}(k)=\mathbf{c}^{(M)}(k-1)$ and $\mathbf{v}(k)=\mathbf{v}^{(M)}(k-1)$.

The output $y(k)$ for $k > k_f$ can now be written as

$$y(k) = y_c(k) + y_v(k) = \mathbf{w}^{\mathbf{H}}(k_f)\mathbf{c}(k) + \mathbf{w}^{\mathbf{H}}(k_f)\mathbf{v}(k). \quad (37)$$

Note that with the initialization in (35), $\mathbf{v}(k) = \mathbf{0}$ and $y(k) = y_c(k)$ for $k > k_f + N_1$.

The above formulation allows us to make use of our previous results and divide the problem into two parts that can be carried out in parallel: one equivalent-output part to produce $y_c(k)$ from the convolution of $\mathbf{c}(k)$ and the coefficients that are obtained using weight extraction, and one equivalent-output part reproducing $y_v(k)$.

Reproducing $y_v(k)$ is straightforward. We only need to apply the results in previous section using vector $\mathbf{v}(k)$ in place of $\mathbf{x}(k)$.

In order to compute the output $y_c(k)$ for $k > k_f$ the coefficients of the channel are extracted using the parallel weight extraction approach given as a remark in Section III. The weight extraction gives one coefficient per channel per iteration. The output $y_c(k)$ is the sum of each channel output $y_{c,l}(k)$

$$\begin{aligned} y_c(k) &= \sum_{l=1}^{M} y_{c,l}(k) \\ y_{c,l}(k) &= \sum_{j=0}^{k-k_f+1} w_{l,j}(k_f)c_l(k-j),\ k_f < k \leq k_f + N_1. \end{aligned} \quad (38)$$

We see from (38) that $y_c(k)$ requires knowledge of M new weights (one per channel) per time instant k, since $c_l(k) = x_l(k) \forall k > k_f$ and $c_l(k) = 0$ for $k \leq k_f$. We can thus apply in parallel with the output filtering $y_c(k)$, an "on-the-fly" weight extraction. The advantage of using parallel weight extraction approach is that $k_f + N_1$ iterations are required to extract all the weights, and then the output $y(k)$ is simply given by $\mathbf{w}^{\mathrm{H}}(k_f)\mathbf{c}(k)$, where at $k > k_f + N_1$, $\mathbf{x}(k) = \mathbf{c}(k)$.

The computational complexity of this approach is given in Table IV. For the initial operations the complexity is reduced linearly from that of Approach 1. After all weights have been acquired, the complexity is only P multiplications per iteration for the rest of the burst.

V. Predistortion of HPA With Memory

This section shows how to apply the MC-FQRD-RLS algorithm in an indirect-learning setup to linearize a nonlinear power amplifier.

A. Problem Description

The objective of a predistorter is to linearize the nonlinear transfer function (including memory) of a high-power amplifier (HPA). As a result, more power efficient transmission is allowed without causing signal waveform distortion and spectral regrowth. Herein we consider an adaptive Volterra-based predistorter that is designed using the indirect learning structure depicted in Fig. 4 [25]. This type of learning structure has also been employed in several recently proposed predistorter designs, see, e.g., [19] and [20]. In the indirect learning structure, the predistorter coefficient vector $\mathbf{w}(k-1)$ is directly calculated

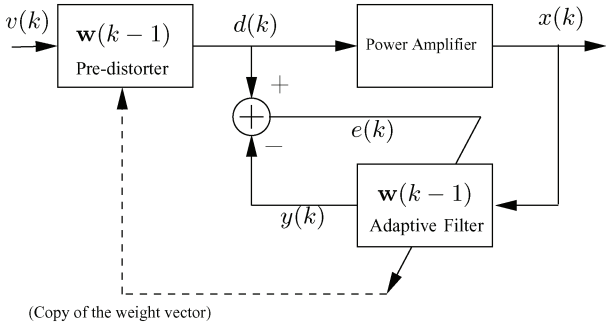

Fig. 4. Indirect learning based predistorter [25].

without the need for a PA parameter identification step. We see in Fig. 4 that the predistorter coefficient vector $\mathbf{w}(k-1)$ is obtained by minimizing the error signal $e(k)$ obtained from the training branch output signal $y(k)$ and the predistorter output signal $d(k)$. The parameters obtained in the training branch are directly copied to the predistorter, avoiding the inverse calculation process.

Let us now consider an implementation where the MC-FQRD-RLS algorithm is used for the predistorter identification. We see from Fig. 4 that the coefficient vector embedded in the MC-FQRD-RLS variables is needed for reproducing the output signal $d(k) = \mathbf{w}^{\mathrm{H}}(k-1)\mathbf{v}(k), \mathbf{v}(k) \in \mathbb{C}^{P\times 1}$ being the predistorter input-signal vector. We know from Section III that $\mathbf{w}(k-1)$ can be made explicitly available at every iteration. However, as can seen from Table V such an approach will lead to a solution of $\mathcal{O}(P^2)$ complexity per iteration. In other words, there is no obvious gain of using an MC-FQRD-RLS algorithm in place of an IQRD-RLS algorithm. One solution to this complexity problem is to extract and copy the weights at a reduced rate, say once every K samples. Obviously, such an approach will no longer yield identical results to an IQRD-RLS algorithm, .and the convergence behavior will certainly be different

Our goal here is to reproduce, at each iteration, the exact output $d(k)$ associated with the weight vector $\mathbf{w}(k-1)$ embedded in the MC-FQRD-RLS algorithm. The total computational complexity for calculating $d(k)$ and updating $\mathbf{w}(k-1)$ will be $\mathcal{O}(P)$ per iteration. The resulting structure will yield exactly the same result as using an IQRD-RLS algorithm, while reducing the computational complexity by an order of magnitude.

B. Multi-Channel Indirect Learning Predistorter

The solution is very similar to that of the problem dealt with in Section IV where the weight vector in the MC-FQRD-RLS algorithm was used for fixed filtering. The main difference here is that the weight vector in the lower branch is continuously updated by the MC-FQRD-RLS algorithm.

The output of the predistorter block is given by

$$\begin{aligned} d(k) &= \sum_{l=1}^{M}\sum_{j=0}^{N_l} w_{l,j}^{*}(k-1)v_l(k-j) \\ &= \mathbf{w}^{\mathbf{H}}(k)\mathbf{v}(k) \\ &= \mathbf{d}_{q2}^{\mathbf{H}}(k-1)\underbrace{\mathbf{U}^{-\mathbf{H}}(k-1)\mathbf{v}(k)}_{\tilde{\mathbf{f}}(k-1)} \end{aligned} \quad (39)$$

where, $\mathbf{d}_{q2}(k-1)$ and $\mathbf{U}^{-\mathrm{H}}(k-1)$ are parameters of the MC-FQRD-RLS algorithm running in the lower branch of Fig. 4. The rotated desired signal vector $\mathbf{d}_{q2}(k)$ is directly available in the MC-FQRD-RLS algorithm. However, the Cholesky factor matrix $\mathbf{U}^{-\mathrm{H}}(k-1)$ is hidden in vector $\mathbf{f}(k-1) = \mathbf{U}^{\mathrm{T}}(k-1)\mathbf{x}(k-1)$. On the other hand, we know from Lemma 4 that, given the new channel input signals $\{x_i\}_{i=1}^{M}$, matrix $\mathbf{Q}_{\theta f}^{(l)}(k-1)$ provides the relation $\mathbf{U}^{\mathrm{H}}(k-2)\mathbf{x}(k-2) \rightarrow \mathbf{U}^{\mathrm{H}}(k-1)\mathbf{x}(k-2)$. Since vectors $\mathbf{x}(k)$ and $\mathbf{v}(k)$ are both initially set to zero, we can modify (30) and (31) for calculating $d(k)$ for $k > 0$. The required computations are summarized by Lemma 5.

Lemma 5: Let $\tilde{\mathbf{f}}(k) \in \mathbb{C}^{P\times 1}$ denote the product $\mathbf{U}^{-\mathrm{H}}(k)\mathbf{v}(k)$, where $\mathbf{U}^{-\mathrm{H}}(k) \in \mathbb{C}^{P\times P}$ is given in (5) and $\mathbf{v}(k) \in \mathbb{C}^{P\times 1}$ is a multichannel input vector. Then $\tilde{\mathbf{f}}(k)$ can be obtained from $\tilde{\mathbf{f}}(k-1)$ in M sequential steps

$$\tilde{\mathbf{f}}(k-1) \rightarrow \tilde{\mathbf{f}}^{(1)}(k-1) \rightarrow \cdots \rightarrow \tilde{\mathbf{f}}^{(M)}(k-1) = \tilde{\mathbf{f}}(k)$$

where each step $\tilde{\mathbf{f}}^{(l-1)}(k-1) \rightarrow \tilde{\mathbf{f}}^{(l)}(k-1)$ is given as

$$\begin{bmatrix} \frac{\varepsilon_b^{(i)}(k)}{\left\|\mathbf{e}_b^{(i)}(k)\right\|} \\ \tilde{\mathbf{f}}^{(i)}(k-1) \end{bmatrix} = \mathbf{\Pi}_i \mathbf{Q}_{\theta f}^{(i)}(k) \begin{bmatrix} \tilde{\mathbf{f}}^{(i-1)}(k-1) \\ \frac{\varepsilon_f^{(i)}(k)}{\left\|\mathbf{e}_f^{(i)}(k)\right\|} \end{bmatrix} \tag{40}$$

where

$$\varepsilon_f^{(i)}(k) = v_l(k) - \left[\mathbf{d}_{fq2}^{(i)}(k)\right]^{\mathbf{H}} \tilde{\mathbf{f}}^{(i-1)}(k-1) \tag{41}$$

with $\mathbf{Q}_{\theta f}^{(i)}(k) \in \mathbb{C}^{(P+1)\times(P+1)}$ and $\mathbf{\Pi}_i \in \mathbb{C}^{(P+1)\times(P+1)}$ as in (21). The index $i = 1, \ldots, M$ and, at $i = 1$ $\tilde{\mathbf{f}}^{(0)}(k-1) = \tilde{\mathbf{f}}(k-1)$. Finally, $\tilde{\mathbf{f}}(k) = \tilde{\mathbf{f}}^{(M)}(k-1)$

The proof of Lemma 5 is given in Appendix.

Equation (40) requires approximately $4P$ multiplications at each of the M steps. Thus, the total number of multiplications is $\mathcal{O}(PM)$. A better approximation of the computational complexity of the proposed method is given in Table IV. The pseudocode for the reproducing the predistorter output $d(k)$ is given in Table V.

Remark 4: In the adaptive filtering block, vector $\mathbf{f}(k)$ is initialized with zeros, and the rotation matrices are taken as identity matrices. Similarly, in the predistorter block $\tilde{\mathbf{f}}(k)$ is initialized with zeros. It is evident from Fig. 4 that the initial weight vector in the predistorter block cannot be zero, since this will not force any update to take place. This is circumvented by initializing the last value of the all zero vector $\mathbf{d}_{q2}(k)$ with a unity.

Remark 5: The novel mechanisms described in this paper allow us to apply the MC-FQRD-RLS algorithm to applications that traditionally exploit adaptation algorithms with explicit update equations for the coefficient weight vector, e.g., DFE and indirect learning predistorter. The reduction in computational complexity is achieved by avoiding the weight extraction mechanism at a particular time instant, k, which would require $\mathcal{O}(P^2)$ operations per iteration. The methods presented here instead exploit the update equations in the MC-FQRD-RLS algorithm, for these equations require only $\mathcal{O}(P)$ operations per iteration index k. As a result, the computational complexity of implementing MC-FQRD-RLS algorithm in a DFE or an indirect learning predistorter is reduced by an order of magnitude, which otherwise would have been $\mathcal{O}(P^2)$ owing to the complexity of the weight extraction mechanism. Note that in burst-trained adaptive filters, weight extraction may also be carried out in tandem with the output-filtering to further lower the overall computational complexity.

TABLE V
INDIRECT LEARNING ALGORITHM

Indirect learning starting from $k = 0$

Available from the MC-FQRD-RLS algorithm:
$\mathbf{d}_{fq2}^{(i)}(k-2)$, $\|\mathbf{e}_f^{(i)}(k-2)\|$, $\mathbf{Q}_{\theta f}^{(i)}(k-1)$ and $\mathbf{d}_{q2}(k-1)$, $i = 1, \ldots, M$

Initialization:
$\tilde{\mathbf{f}}^{(0)}(-1) = \mathbf{0}_{P\times 1}$

for each k
{
  Compute $\tilde{\mathbf{f}}(k)$ from $\tilde{\mathbf{f}}^{(i-1)}(k-1)$:
  $\tilde{\mathbf{f}}^{(0)}(k-1) = \tilde{\mathbf{f}}(k-1)$
  for each $i = 1 : M$
  {
    $\varepsilon_f^{(i)}(k-2) = v_i(k) - [\mathbf{d}_{fq2}^{(i)}(k-2)]^{\mathrm{H}} \tilde{\mathbf{f}}^{(i-1)}(k-1)$
    $\begin{bmatrix} \frac{\varepsilon_b^{(i)}(k-1)}{\|\mathbf{e}_b^{(i)}(k-2)\|} \\ \tilde{\mathbf{f}}^{(i)}(k-1) \end{bmatrix} = \mathbf{Q}_{\theta f}^{(i)}(k-1) \begin{bmatrix} \tilde{\mathbf{f}}^{(i-1)}(k-1) \\ \frac{\varepsilon_f^{(i)}(k-2)}{\|\mathbf{e}_f^{(i)}(k-2)\|} \end{bmatrix}$
  }
  $\tilde{\mathbf{f}}(k) = \tilde{\mathbf{f}}^{(M)}(k-1)$
  Compute $d(k) = \mathbf{w}^{\mathrm{H}}(k-1)\mathbf{v}(k)$:
  $d^*(k) = \mathbf{d}_{q2}^{\mathrm{H}}(k-1)\tilde{\mathbf{f}}(k)$
}

VI. SIMULATIONS

A. Volterra System Identification

The weight extraction algorithm is evaluated using a Volterra system identification setup given in [9]. The system to be identified comprises of 10 linear $\{b_{0,j}\}_{j=0}^{N_0}$ and 55 quadratic $\{b_{l,j}\}_{l=1,j=0}^{10,N_l}$ coefficients given in Table VI. A perfect knowledge of the number of coefficients of the unknown system is assumed. The input signal is obtained by filtering an additive white Gaussian noise with zero mean and 0.0248 variance by an FIR filter with impulse response $0.9045\delta(n) + 1\delta(n-1) + 0.9045\delta(n-2)$, where $\delta(n)$ is the Kronecker delta function. As a result, output of the unknown system has unit variance. The SNR is set to 30 dB. The results of the MC-FQRD-RLS algorithm and IQRD-RLS algorithm are compared. The forgetting factor is set to 0.99. Results presented are ensemble average of 10 runs over 2000 samples. The squared error of the weight vector coefficients obtained from IQRD-RLS and FQRD-RLS algorithms is computed as

$$w_{\mathrm{diff},i}(k) = 20\log_{10}\left|w_{\mathrm{FQR},i}(k) - w_{\mathrm{IQR},i}(k)\right| \tag{42}$$

and the plot is shown in Fig. 5, where $w_{\mathrm{FQR},i}(k)$ is the ith weight of the FQRD-RLS algorithm, and $w_{\mathrm{IQR},i}$ is the ith weight of the inverse QRD-RLS algorithm. The squared difference of output error signal of the IQRD-RLS and the FQRD-RLS algorithms is shown in Fig. 6. It is important to mention that for a valid comparison of weight vectors, the learning curve of both the algorithms should match exactly.

TABLE VI
LINEAR AND QUADRATIC COEFFICIENTS USED IN THE SYSTEM IDENTIFICATION SETUP

$l \times j$	0	1	2	3	4	5	6	7	8	9
Linear coefficients										
0	-0.052	0.723	0.435	-0.196	-0.143	0.812	0.354	0.077	-1.379	2.25
Quadratic coefficients										
1	1.020	-1.812	-1.138	-0.592	-0.144	-0.966	-1.454	1.820	-4.022	5.562
2	1.389	-2.608	-1.486	-1.382	-2.308	4.256	0.626	-0.264	2.890	0
3	-0.635	-0.468	-1.508	0.812	1.284	1.580	-1.800	0.748	0	0
4	-1.044	0.536	-2.092	-0.774	-3.314	-0.348	0.272	0	0	0
5	0.011	2.918	0.698	0.752	-3.496	0.460	0	0	0	0
6	-0.987	3.940	2.926	-0.508	1.648	0	0	0	0	0
7	0.198	-0.362	-2.402	1.646	0	0	0	0	0	0
8	-1.732	-1.334	-3.070	0	0	0	0	0	0	0
9	0.860	0.648	0	0	0	0	0	0	0	0
10	0.305	0	0	0	0	0	0	0	0	0

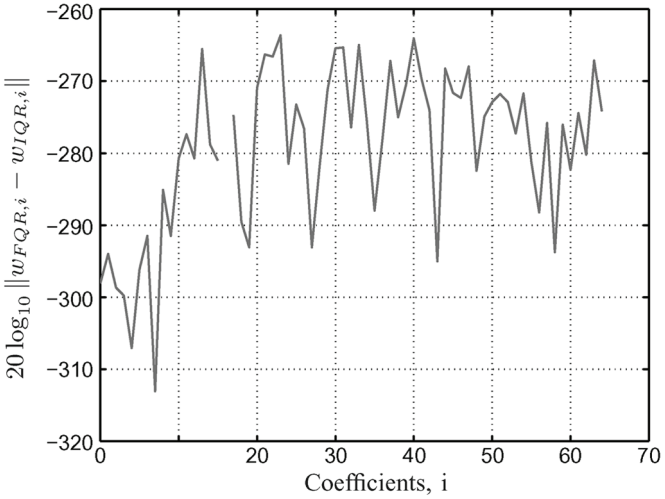

Fig. 5. Coefficient weight difference of the MC-FQRD-RLS algorithm and the IQRD-RLS algorithm.

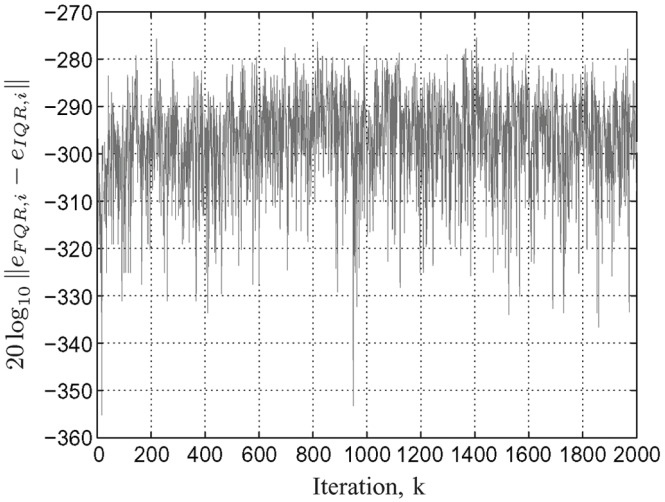

Fig. 6. Difference in MSE of the MC-FQRD-RLS and the IQRD-RLS algorithms.

This means, both the algorithms should have equivalent initial conditions. (See Remark 2 in Section III.)

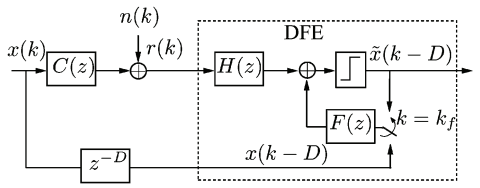

Fig. 7. A decision feedback equalizer setup.

B. Burst-Trained DFE

In this section we consider a burst-trained decision feedback equalizer (DFE), see Fig. 7. Viewing the DFE as a multichannel filter we will have $M = 2$ channels. In our simulations, the lengths of feedforward $H(z)$ and feedback $F(z)$ filters are $N_1 = 10$ and $N_2 = 3$, respectively. A minimum phase channel is considered, whose impulse response is defined by $\mathbf{c}(z) = 1 + 0.9z^{-1} + 0.8z^{-2} + 0.7z^{-3} \ldots + 0.1z^{-9}$. During the first 300 iterations (i.e., $k_f = 300$), the equalizer coefficients are updated by the MC-FQRD-RLS algorithm. The training symbols $d(k)$ were randomly generated QPSK symbols. Following the initial training sequence, an unknown symbol sequence consisting of 700 16-QAM symbols was transmitted over the channel and the equalizer output was reproduced using the approach in Section IV. For comparison purposes, an IQRD-RLS algorithm was also implemented. Note that the IQRD-RLS has a complexity of $\mathcal{O}(P^2)$ during coefficient adaptation. The SNR was set to 30 dB, and the forgetting factor was chosen to $\lambda = 0.95$. The training signal was delayed by $D = 3$ samples.

The learning curves, averaged over 50 ensembles, and the recovered constellation are shown in Figs. 8 and 9. The results show that the MC-FQRD-RLS algorithm present exactly the same behavior as IQRD-RLS algorithm.

C. Adaptive Volterra-Based Predistorter

The predistorter simulation setup for this example is taken from [25], where a simplified baseband nonlinear satellite channel is considered. The PA is modeled as a Wiener-Hammerstein system which is formed by a cascade of a linear filter $\mathbf{h}$, a nonlinear function $f\{.\}$, and another linear filter $\mathbf{g}$. The

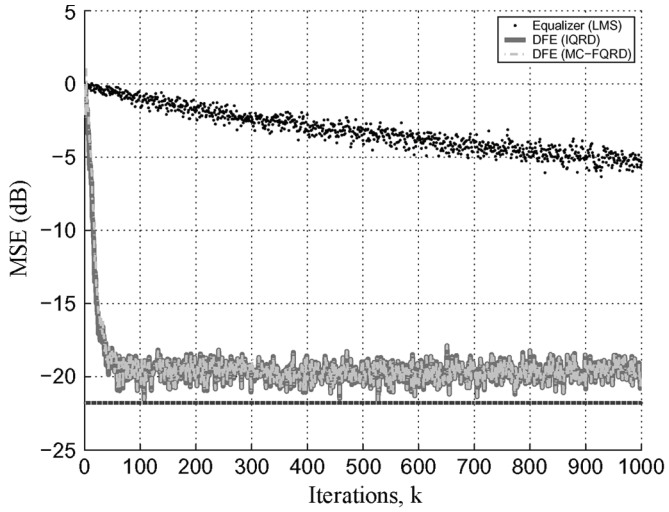

Fig. 8. Learning curves for the MC-FQRD-RLS algorithm using output filtering, and IQRD-RLS in DFE setup.

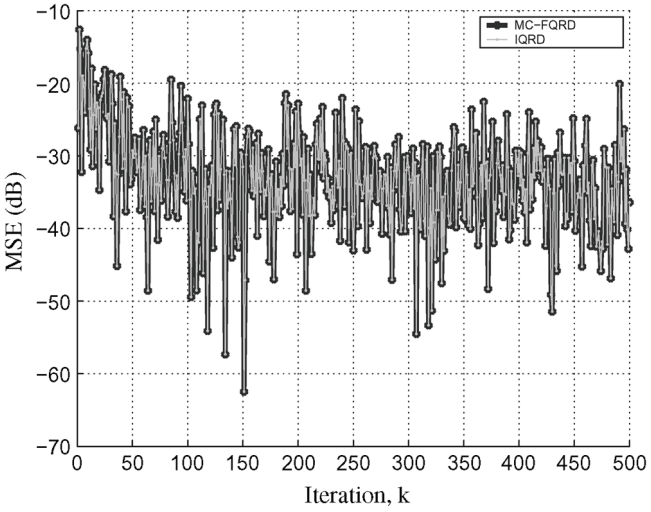

Fig. 10. Learning curve for the MC-FQRD-RLS algorithm using weight extraction Lemmas, and the IQRD-RLS algorithm.

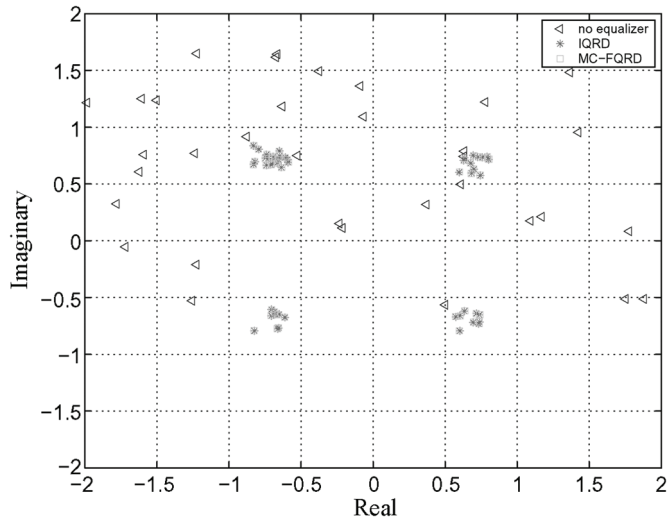

Fig. 9. 4-QAM constellation distorted by a linear channel and the equalized constellations using DFE implemented using the MC-FQRD-RLS (output filtering) and the IQRD-RLS algorithms.

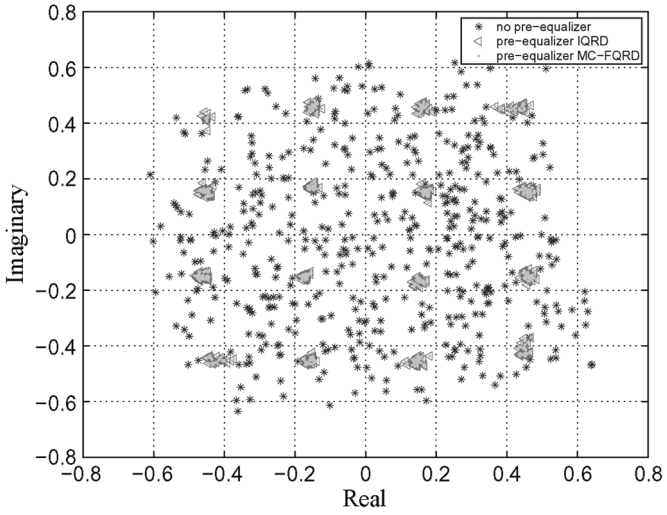

Fig. 11. 16-QAM constellation distorted by a nonlinear amplifier and the compensated constellations using the indirect learning architecture.

non-linearity is due to the traveling wave tube amplifier (TWT) with the AM/AM and AM/PM conversion given as

$$\begin{aligned} f\left(d(k)\right) &= A\left(d(k)\right) e^{j\Phi[d(k)]}, \text{ with} \\ A\left[r(k)\right] &= \frac{\alpha_a r(k)}{1+\beta_a r^2(k)} \\ \Phi\left[r(k)\right] &= \frac{\alpha_p r^2(k)}{1+\beta_p r^2(k)} \end{aligned} \tag{43}$$

where α_a, β_a, α_p, and β_p are the constants given by 2.0, 1.0, $\pi/3$, and 1.0, respectively. The linear prefilter and postfilter responsible for the memory effects are taken as FIR with coefficient vectors

$$\begin{aligned} \mathbf{h} &= [0.8 \quad 0.1]^{\mathrm{T}} \\ \mathbf{g} &= [0.9 \quad 0.2 \quad 0.1]^{\mathrm{T}}. \end{aligned} \tag{44}$$

A third-order Volterra model with a memory length of 7, i.e., 119 taps is employed for the indirect learning architecture in order to compensate for the nonlinear system. The input signal comprises of 500 symbols from a 16-QAM constellation, and there are 50 simulation runs. The maximum amplitude value of the constellation is restricted to 0.64 so that the output of the predistorter is in the input range of the nonlinear TWT model. MC-FQRD-RLS algorithm with weight extraction is compared with in-direct learning mechanism introduced in Section V-B. Fig. 10 shows the same learning curve for both algorithms. The IQRD-RLS algorithm is also considered for comparison. It can be seen that all the learning curves converge to -40 dB. The constellation diagram with and without a PD is shown in Fig. 11. We see that both the IQRD-RLS and the MC-FQRD-RLS algorithms are able to compensate for the nonlinear distortion. However, the computational complexity of the MC-FQRD-RLS algorithm is significantly lower than that of the IQRD-RLS algorithm.

VII. Conclusions

This article showed how to reuse the internal variables of the multichannel fast QR decomposition RLS (MC-FQRD-RLS) algorithm to enable new applications which are different from the standard output-error type applications. We first considered the problem of multichannel (Volterra) system identification and showed how to extract the weights in a serial manner. Thereafter, the weight extraction results were extended to the case of burst-trained decision-feedback equalizers, where the equalizer is periodically retrained using pilots and then used for fixed filtering of useful data. Finally, we considered the problem adaptive predistortion of high-power amplifiers using the indirect learning control structure. Simulation results in these example applications were compared with those using a designs based on the inverse QRD-RLS algorithm. It was verified that identical results are obtained using the FQRD-RLS methods at a much lower computational cost.

Appendix I

A. Update of Vector $\mathbf{f}(k)$

The forward and backward prediction and error vectors, $\mathbf{e}_f^{(i)}(k)$ and $\mathbf{e}_b^{(i)}(k)$ for the ith channel are defined as

$$\begin{aligned}\mathbf{e}_f^{(i)}(k) &= \mathbf{d}_f^{(i)}(k) - \begin{bmatrix}\mathbf{X}^{(i-1)}(k-1)\\ \mathbf{0}_{1\times P}\end{bmatrix}\mathbf{w}_f(k)\\ &= \underbrace{\begin{bmatrix}\mathbf{d}_f^{(i)}(k) & \mathbf{X}^{(i-1)}(k-1)\\ & \mathbf{0}_{1\times P}\end{bmatrix}}_{\mathbf{X}_f(k)}\begin{bmatrix}1\\ -\mathbf{w}_f(k)\end{bmatrix} \quad (45)\\ \mathbf{e}_b^{(i)}(k) &= \mathbf{d}_b^{(i)}(k) - \mathbf{X}^{(i)}(k-1)\mathbf{w}_b(k)\\ &= \underbrace{\begin{bmatrix}\mathbf{X}^{(i)}(k-1) & \mathbf{d}_b^{(i)}(k)\end{bmatrix}}_{\mathbf{X}_b(k)}\begin{bmatrix}-\mathbf{w}_b(k)\\ 1\end{bmatrix} \quad (46)\end{aligned}$$

where $\mathbf{w}_f(k)$ and $\mathbf{w}_b(k)$ are the respective forward and backward prediction coefficient vectors, and $\mathbf{d}_f^{(i)}(k)$ and $\mathbf{d}_b^{(i)}(k)$ are the forward and backward desired signal vectors given by

$$\mathbf{d}_f^{(i)}(k) = \begin{bmatrix}x_i(k)\ \lambda^{1/2}x_i(k-1)\ \ldots\ \lambda^{k/2}x_i(0)\end{bmatrix}^{\mathrm{T}} \quad (47)$$

$$\mathbf{d}_b^{(i)}(k) = \begin{bmatrix}x_i(k-N_i)\ \ldots\ \lambda^{(k-N_i)/2}x_i(0)\ \mathbf{0}_{1\times(N_i+1)}\end{bmatrix}^{\mathrm{T}}. \quad (48)$$

Let $\mathbf{Q}_b^{(i)}(k) \in \mathbb{C}^{(k+2)\times(k+2)}$ and $\mathbf{Q}_f^{(i)}(k) \in \mathbb{C}^{(k+2)\times(k+2)}$ denote the respective Givens rotations matrices responsible for the Cholesky factorization of $\mathbf{X}^{(i)}(k-1)$ and $\mathbf{X}^{(i-1)}(k-1)$, i.e.

$$\mathbf{Q}_b^{(i)}(k)\mathbf{X}_b(k) = \begin{bmatrix}\mathbf{0}_{(k+2-P)\times(P+1)}\\ \mathbf{0}_{1\times P} & \left\|\mathbf{e}_b^{(i)}(k)\right\|\\ \mathbf{U}^{(i)}(k-1) & \mathbf{d}_{bq2}^{(i)}(k)\end{bmatrix} \quad (49)$$

and

$$\mathbf{Q}_f^{(i)}(k)\mathbf{X}_f(k) = \begin{bmatrix}\mathbf{0}_{(k+2-P)\times(P+1)}\\ \mathbf{d}_{fq2}^{(i)}(k) & \mathbf{U}^{(i-1)}(k-1)\\ \left\|\mathbf{e}_f^{(i)}(k)\right\| & \mathbf{0}_{1\times P}\end{bmatrix}. \quad (50)$$

The forward and backward prediction coefficient vectors that minimize the norm of forward and backward prediction error in (45) and (46), obtained in a similar manner as (7), are given by

$$\mathbf{w}_f(k) = \mathbf{U}^{-(i-1)}(k-1)\mathbf{d}_{fq2}^{(i)}(k) \quad (51)$$

$$\mathbf{w}_b(k) = \left.\mathbf{U}^{-(i)}(k-1)\right]\mathbf{d}_{bq2}^{(i)}(k) \quad (52)$$

Equations (45) and (46) relate to each other via the permutation matrix $\mathbf{\Pi}_i$ in (14) as [22]

$$\mathbf{X}_b(k) = \mathbf{X}_f(k)\mathbf{\Pi}_i, \quad i = 1,\ldots,M. \quad (53)$$

From (53), we obtain

$$\begin{aligned}&\mathbf{X}_b^{\mathbf{H}}(k)\mathbf{Q}_b^{\mathbf{H}(i)}(k)\mathbf{Q}_b^{(i)}(k)\mathbf{X}_b(k)\\ &\qquad = \mathbf{\Pi}_i^{\mathbf{H}}\mathbf{X}_f^{\mathbf{H}}(k)\mathbf{Q}_f^{\mathbf{H}(i)}(k)\mathbf{Q}_f^{(i)}(k)\mathbf{X}_f(k)\mathbf{\Pi}_i. \quad (54)\end{aligned}$$

Inserting (49) and (50) in (54) yields

$$\begin{aligned}&\begin{bmatrix}\mathbf{0}_{1\times P} & \left\|\mathbf{e}_b^{(i)}(k)\right\|\\ \mathbf{U}^{(i)}(k-1) & \mathbf{d}_{bq2}^{(i)}(k)\end{bmatrix}^{\mathbf{H}}\begin{bmatrix}\mathbf{0}_{P\times 1} & \left\|\mathbf{e}_b^{(i)}(k)\right\|\\ \mathbf{U}^{(i)}(k-1) & \mathbf{d}_{bq2}^{(i)}(k)\end{bmatrix}\\ &\quad = \mathbf{\Pi}_i\begin{bmatrix}\mathbf{d}_{fq2}^{(i)}(k) & \mathbf{U}^{(i-1)}(k-1)\\ \left\|\mathbf{e}_f^{(i)}(k)\right\| & \mathbf{0}_{P\times 1}\end{bmatrix}^{\mathbf{H}}\\ &\qquad \times\begin{bmatrix}\mathbf{d}_{fq2}^{(i)}(k) & \mathbf{U}^{(i-1)}(k-1)\\ \left\|\mathbf{e}_f^{(i)}(k)\right\| & \mathbf{0}_{1\times P}\end{bmatrix}\mathbf{\Pi}_i. \quad (55)\end{aligned}$$

In order to relate the two square-root factors in (55) we need to introduce the unitary transformation $\mathbf{\Pi}_i\mathbf{Q}_{\theta f}^{(i)}(k)$. Therefore, we can write

$$\begin{aligned}&\begin{bmatrix}\mathbf{0}_{1\times P} & \left\|\mathbf{e}_b^{(i)}(k)\right\|\\ \mathbf{U}^{(i)}(k-1) & \mathbf{d}_{bq2}^{(i)}(k)\end{bmatrix}\\ &\qquad = \mathbf{\Pi}_i\mathbf{Q}_{\theta f}^{(i)}(k)\begin{bmatrix}\mathbf{d}_{fq2}^{\mathbf{H}(i)}(k) & \mathbf{U}^{(i-1)}(k-1)\\ \left\|\mathbf{e}_f^{(i)}(k)\right\| & \mathbf{0}_{1\times P}\end{bmatrix}\mathbf{\Pi}_i \quad (56)\end{aligned}$$

where rotation matrix $\mathbf{Q}_{\theta f}^{(i)}(k) \in \mathbb{C}^{(P+1)\times(P+1)}$ annihilates the nonzero values and the permutation matrix $\mathbf{\Pi}_i$ operates from left and right to shift rows and the columns to complete the triangularization, see [22] for details. The norms $\|\mathbf{e}_b^{(i)}(k)\|$ and $\|\mathbf{e}_f^{(i)}(k)\|$ are defined as

$$\left\|\mathbf{e}_b^{(i)}(k)\right\| = \sqrt{\left\|\mathbf{e}_b^{(i)}(k-1)\right\|^2 + \left\|e_{bq1}^{(i)}(k)\right\|^2} \quad (57)$$

$$\left\|\mathbf{e}_f^{(i)}(k)\right\| = \sqrt{\left\|\mathbf{e}_f^{(i)}(k-1)\right\|^2 + \left\|e_{fq1}^{(i)}(k)\right\|^2}. \quad (58)$$

where $e_{bq1}^{(i)}(k)$ and $e_{fq1}^{(i)}(k)$ are the first elements of $\mathbf{Q}(k)\mathbf{e}_f(k)$ and $\mathbf{Q}(k)\mathbf{e}_b(k)$, respectively. They also related to vectors $\mathbf{d}_{bq2}^{(i)}(k)$ and $\mathbf{d}_{fq2}^{(i)}(k)$ in similar way as $e_{q1}(k)$ relates to vector $\mathbf{d}_{q2}(k)$ in (9).

Inverting (56) and taking the Hermitian transpose results in

$$\begin{aligned}&\begin{bmatrix}-\frac{\mathbf{d}_{bq2}^{\mathbf{H}(i)}(k)\mathbf{U}^{-\mathbf{H}(i)}(k-1)}{\left\|\mathbf{e}_b^{(i)}(k)\right\|} & \frac{1}{\left\|\mathbf{e}_b^{(i)}(k)\right\|}\\ \mathbf{U}^{-\mathbf{H}(i)}(k-1) & \mathbf{0}_{P\times 1}\end{bmatrix}\\ &= \mathbf{\Pi}_i\mathbf{Q}_{\theta f}^{(i)}(k)\begin{bmatrix}\mathbf{0}_{P\times 1} & \mathbf{U}^{-\mathbf{H}(i-1)}(k-1)\\ \frac{1}{\left\|\mathbf{e}_f^{(i)}(k)\right\|} & -\frac{\mathbf{d}_{fq2}^{\mathbf{H}(i)}(k)\mathbf{U}^{-\mathbf{H}(i-1)}(k-1)}{\left\|\mathbf{e}_f^{(i)}(k)\right\|}\end{bmatrix}\mathbf{\Pi}_i. \quad (59)\end{aligned}$$

Note that the iteration index i leads to the update from $\mathbf{U}^{-\mathrm{H}}(k-1)$ to $\mathbf{U}^{-\mathrm{H}}(k)$ in M-steps, where the current index value k of ith channel input $x_i(k)$ is consumed at each step. Finally, after postmultiplying both sides of (59) with $[x_i(k)\ [\mathbf{x}^{(i-1)}(k-1)]^{\mathrm{T}}]^{\mathrm{T}}$ we obtain

$$\begin{bmatrix} \frac{x_i(k-N_i)-\mathbf{w}_b^{\mathbf{H}(i)}(k)\mathbf{x}^{(i)}(k)}{\left\|\mathbf{e}_b^{(i)}(k)\right\|} \\ \mathbf{U}^{-\mathbf{H}(i)}(k-1)\mathbf{x}^{(i)}(k) \end{bmatrix} = \mathbf{\Pi}_i \mathbf{Q}_{\theta f}^{(i)}(k) \begin{bmatrix} \mathbf{U}^{-\mathbf{H}(i-1)}(k-1)\mathbf{x}^{(i)}(k-1) \\ \frac{x_i(k)-\mathbf{w}_f^{\mathbf{H}(i)}(k)\mathbf{x}^{(i)}(k-1)}{\left\|\mathbf{e}_f^{(i)}(k)\right\|} \end{bmatrix}. \quad (60)$$

APPENDIX II

A. Proof of Lemma 1

The update equation for $\mathbf{U}^{-\mathrm{H}}(k-1)$ in the inverse QRD-RLS algorithm is given by [16]

$$\begin{bmatrix} \mathbf{z}^{\mathbf{H}}(k) \\ \mathbf{U}^{-\mathbf{H}}(k) \end{bmatrix} = \mathbf{Q}_\theta(k) \begin{bmatrix} \mathbf{0}_{1\times P}^{\mathrm{T}} \\ \lambda^{-1/2}\mathbf{U}^{-\mathbf{H}}(k-1) \end{bmatrix} \quad (61)$$

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

$$\begin{bmatrix} 0 \\ \lambda^{-1/2}\mathbf{u}_{l,j}(k-1) \end{bmatrix} = \mathbf{Q}_\theta^{\mathbf{H}}(k) \begin{bmatrix} z_j(k) \\ \mathbf{u}_{l,j}(k) \end{bmatrix} \quad (62)$$

where $z_j(k)$ is the jth element of vector $\mathbf{z}(k)$

$$z_j(k) = -\mathbf{f}^{\mathbf{H}}(k)\mathbf{u}_{l,j}(k)/\gamma^{(M)}(k) \quad (63)$$

and $\mathbf{f}(k-1)$ and $\gamma^{(M)}(k-1)$ are provided by the MC-FQRD-RLS algorithm.

B. Proof of Lemma 2

The proof follows directly from the column partitions of (59), i.e.

$$\begin{bmatrix} \frac{-w_{b,j-1}^{(i)}(k-1)}{\left\|\mathbf{e}_b^{(i)}(k-1)\right\|} \\ \mathbf{u}_{l,j}^{(i)}(k-1) \end{bmatrix} = \mathbf{\Pi}_i \mathbf{Q}_{\theta f}^{(i)}(k) \begin{bmatrix} \mathbf{u}_{l,j}^{(i-1)}(k-1) \\ \frac{-w_{f,j-1}^{(i)}(k-1)}{\left\|\mathbf{e}_f^{(i)}(k-1)\right\|} \end{bmatrix} \quad (64)$$

where $i = 1, \ldots, M$, $j = 0, \ldots, N_i - 1$, and

$$w_{f,j-1}^{(i)}(k-1) = \begin{cases} 0 & \text{for } j < l-1 \\ -1 & \text{for } j = l-1 \\ -\left[\mathbf{u}_{l,j}^{(i-1)}(k-1)\right]^{\mathrm{T}} \mathbf{d}_{fq2}^{(i)}(k) & \text{otherwise.} \end{cases} \quad (65)$$

We see from (59) that in order to obtain the first column, initialization has to be done with zeros. Therefore $\mathbf{u}_{l,j}^{(0)}(k-1) = \mathbf{u}_{l,j-1}^{(M)}(k-2)$ and $\mathbf{u}_{l,0}^{(0)}(k-1) = \mathbf{0}_{P+i\times 1}$.

C. Proof of Lemma 3

Premultiply both sides of (61) with $\mathbf{Q}_\theta^{\mathrm{H}}(k-1)$ and post multiplying with the input signal vector $\mathbf{x}(k+l)$ leads to

$$\begin{bmatrix} 0 \\ \lambda^{-1/2}\mathbf{U}^{-\mathbf{H}}(k-1)\mathbf{x}(k+l) \end{bmatrix} = \mathbf{Q}_\theta^{\mathbf{H}}(k) \begin{bmatrix} \tilde{z}(k+l) \\ \mathbf{U}^{-\mathbf{H}}(k)\mathbf{x}(k+l) \end{bmatrix} \quad (66)$$

where $\tilde{z}(k+l) = -[\mathbf{f}(k)]^{\mathrm{T}}\mathbf{U}^{-\mathrm{H}}(k)\mathbf{x}(k+l)/\gamma(k)$.

D. Proof of Lemma 4

We postmultiply (59) with a data vector $[x_i(k+l)\ \mathbf{x}^{T(i-1)}(k+l-1)]^{\mathrm{T}}$. This is allowed as (59) can be postmultiplied by any input vector on both sides, yielding Lemma 4.

E. Proof of Lemma 5

We postmultiply (59) with a data vector $[v_i(k)\ \mathbf{v}^{T(i-1)}(k-1)]^{\mathrm{T}}$. This is allowed as (59) can be postmultiplied by any input vector on both sides, yielding Lemma 5. Please note that the initially the input vector is set to zero. ■

REFERENCES

[1] K. H. Kim and E. J. Powers, "Analysis of initialization and numerical instability of fast RLS algorithms," in *Proc. Int. Conf. Acoust., Speech, Signal Process. (ICASSP'91)*, Toronto, Canada, Apr. 1991.

[2] 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 Process.*, vol. 39, no. 4, pp. 876–891, Apr. 1991.

[3] D. T. M. Slock, "Reconciling fast RLS lattice and QR algorithms," in *Proc. Int. Conf. Acoust., Speech, Signal Process. (ICASSP'90)*, Albuquerque, NM, Apr. 1990, vol. 3, pp. 1591–1594.

[4] A. A. Rontogiannis and S. Theodoridis, "Multichannel fast QRD-LS adaptive filtering: New technique and algorithms," *IEEE Trans. Signal Process.*, vol. 46, no. 11, pp. 2862–2876, Nov. 1998.

[5] A. Ramos, J. Apolinário, Jr., and M. G. Siqueira, "A new order recursive multiple order multichannel fast QRD algorithm," in *Proc. 38th Ann. Asilomar Conf. Signals, Syst., Comput.*, Pacific Grove, CA, Nov. 2004.

[6] C. A. Medina, J. Apolinário, Jr., and M. Siqueira, "A unified framework for multichannel fast QRD-LS adaptive filters based on backward prediction errors," in *Proc. IEEE Int. Midwest Symp. Circuits Syst. MWSCAS'O2*, Tulsa, OK, Aug. 2002.

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

[8] J. A. Apolinário, Jr., M. G. Siqueira, and P. S. R. Diniz, "Fast QR algorithms based on backward prediction errors: A new implementation and its finite precision performance," *Circuits Syst. Signal Process.*, vol. 22, pp. 335–349, Jul./Aug. 2003.

[9] M. A. Syed and V. J. Mathews, "QR-decomposition based algorithms for adaptive Volterra filtering," *IEEE Trans. Circuits Syst. I: Fund. Theory Appl.*, vol. 40, no. 6, pp. 372–382, Jun. 1993.

[10] M. Shoaib, S. Werner, J. A. Apolinário, Jr., and T. I. Laakso, "Solution to the weight extraction problem in FQRD-RLS algorithms," in *Proc. Int. Conf. Acoust., Speech, Signal Process. (ICASSP'06)*, Toulouse, France, May 2006.

[11] M. Shoaib, S. Werner, J. A. Apolinário, Jr., and T. I. Laakso, "Multichannel fast QR-decomposition RLS algorithms with explicit weight extraction," in *Proc. EUSIPCO'2006*, Florence, Italy, Sep. 2006.

[12] J. A. Apolinário Jr., *QRD-RLS Adaptive Filtering*. New York: Springer, 2009.

[13] S. Haykin, *Adaptive Filter Theory*, 3rd ed. Englewood Cliffs, NJ: Prentice-Hall, 1996.

[14] G. Carayannis, D. Manolakis, and N. Kalouptsidis, "A unified view of parametric processing algorithms for prewindowed signals," *Signal Process.*, vol. 10, no. 4, pp. 335–368, Jun. 1986.

[15] M. Shoaib, S. Werner, and J. A. Apolinário Jr., "Reduced complexity solution for weight extraction in QRD-LSL algorithm," *IEEE Signal Process. Lett.*, vol. 15, pp. 277–280, 2008.

[16] S. T. Alexander and A. L. Ghirnikar, "A method for recursive least squares filtering based upon an inverse QR decomposition," *IEEE Trans. Signal Process.*, vol. 41, no. 1, pp. 20–30, Jan. 1993.

[17] M. Shoaib, S. Werner, J. A. Apolinário, Jr., and T. I. Laakso, "Equivalent output-filtering using fast QRD-RLS algorithm for burst-type training applications," in *Proc. ISCAS'2006*, Kos, Greece, May 2006.

[18] D. Psaltis, A. Sideris, and A. A. Yamamura, "A multilayered neural network controller," *IEEE Control Syst. Mag.*, vol. 8, pp. 17–21, Apr. 1988.

[19] D. R. Morgan, Z. Ma, J. Kim, M. G. Zierdt, and J. Pastalan, "A generalized memory polynomial model for digital predistortion of RF power amplifiers," *IEEE Trans. Signal Process.*, vol. 54, no. 10, pp. 3852–3860, Oct. 2006.
[20] L. Ding, G. T. Zhou, D. R. Morgan, Z. Ma, J. S. Kenney, J. Kim, and C. R. Giardina, "A robust digital baseband predistorter constructed using memory polynomials," *IEEE Trans. Commun.*, vol. 52, no. 1, pp. 159–165, Jan. 2004.
[21] M. Bouchard, "Numerically stable fast convergence least-squares algorithms for multichannel active sound cancellation systems and sound deconvolution systems," *Signal Process.*, vol. 82, no. 5, pp. 721–736, May 2002.
[22] A. Ramos, J. A. Apolinário, Jr., and S. Werner, "Multichannel fast QRD-RLS adaptive filtering: Block-channel and sequential algorithm based on updating backward prediction errors," *Signal Process.*, vol. 87, pp. 1781–1798, Jul. 2007.
[23] A. Ramos and J. A. Apolinário, Jr., "A new multiple order multichannel fast QRD algorithm and its application to non-linear system identification," in *Proc. XXI Simp. Brasileiro de Telecomun. SBT'04*, Belém, PA, Brazil, Sep. 2004.
[24] P. S. R. Diniz, *Adaptive Filtering: Algorithms and Practical Implementation*, 3rd ed. New York: Springer, 2008.
[25] C. Eun and E. J. Powers, "A new Volterra predistorter based on the indirect learning architecture," *IEEE Trans. Signal Process.*, vol. 45, no. 1, pp. 20–30, Jan. 1997.

Mobien Shoaib (M'03) received undergraduate education in computer engineering from the National University of Sciences and Technology (NUST) in 2001. He received the M.S. degree in telecommunication engineering, from the Department of Electrical Engineering, Helsinki University of Technology (HUT), Espoo, Finland, in 2005.

He is working as a Researcher with the Department of Signal Processing and Acoustics, TKK. He is also currently associated with Prince Sultan Advanced Technology Research Institute (PSATRI), King Saud University.

Stefan Werner (SM'04) received the M.Sc. degree in electrical engineering from the Royal Institute of Technology (KTH), Stockholm, Sweden, in 1998, and the D.Sc. (EE) degree (with honors) from the Signal Processing Laboratory, Smart and Novel Radios (SMARAD) Center of Excellence, Helsinki University of Technology (TKK), Espoo, Finland, in 2002.

He is currently an Academy Research Fellow with the Department of Signal Processing and Acoustics, TKK, where he is also appointed as a Docent. His research interests include adaptive signal processing, signal processing for communications, and statistical signal processing.

Dr. Werner is a member of the editorial board for the *Signal Processing* journal.

José Antonio Apolinário, Jr. (SM'04) graduated from the Military Academy of Agulhas Negras (AMAN), Resende, Brazil, in 1981 and received the B.Sc. degree from the Military Institute of Engineering (IME), Rio de Janeiro, Brazil, in 1988, the M.Sc. degree from the University of Brasília (UnB), Brasília, Brazil, in 1993, and the D.Sc. degree from the Federal University of Rio de Janeiro (COPPE/UFRJ), Rio de Janeiro, in 1998, all in electrical engineering.

He is currently an Adjunct Professor with the Department of Electrical Engineering, IME, where he has already served as the Head of Department and as the Vice-Rector for Study and Research. He was a Visiting Professor with the Escuela Politécnica del Ejército (ESPE), Quito, Ecuador, from 1999 to 2000 and a Visiting Researcher and twice a Visiting Professor with Helsinki University of Technology (HUT), Finland, in 1997, 2004, and 2006, respectively. His research interests comprise many aspects of linear and nonlinear digital signal processing, including adaptive filtering, speech, and array processing. He has recently edited the book *QRD-RLS Adaptive Filtering* (New York: Springer, 2009).

Dr. Apolinário has organized and been the first Chair of the Rio de Janeiro Chapter of the IEEE Communications Society.