Equivalent Output-Filtering using Fast QRD-RLS Algorithm for Burst-Type Training Applications

M. Shoaib,[1)] S. Werner,[1)] J. A. Apolinário Jr.,[2)] T. I. Laakso [1)]

[1)]Helsinki University of Technology
Signal Processing Laboratory
Espoo, Finland
{mobien.shoaib, stefan.werner, timo.laakso}@hut.fi

[2)]Instituto Militar de Engenharia
Department of Electrical Engineering
Rio de Janeiro, Brazil
apolin@ieee.org

***Abstract*— Fast QR decomposition RLS (FQRD-RLS) algorithms are well known for their good numerical properties and low computational complexity. The FQRD-RLS algorithms do not provide access to the filter weights, and their use have so far been limited to problems seeking an estimate of the output error signal. In this paper we present techniques which allow us to reproduce the equivalent output signal corresponding to any input-signal applied to the weight vector of the FQRD-RLS algorithm. As a consequence, we can extend the range of applications of the FQRD-RLS to include 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. The proposed output-filtering techniques are tested in an equalizer setup. 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.**

I. Introduction

The recursive least-squares (RLS) is one of the fastest converging adaptive filtering algorithms. The convergence of the RLS algorithm serves as a benchmark for adaptive filtering algorithms. However, there are numerical stability issues associated with it, mainly when implemented in finite precision [1]. The conventional QRD-RLS algorithm exhibits RLS like convergence and numerical robustness at the same computational complexity as the RLS, i.e., $\mathcal{O}(N^2)$, N being the number of filter coefficients [2]. A number of low-complexity derivatives of QRD-RLS algorithm have been successfully discovered [3]–[8]. In this paper we focus on this efficient subset, commonly called FQRD-RLS algorithms.

The main idea in FQRD-RLS algorithms is to exploit the underlying time-shift structure of the input data vector in order to replace matrix update equations with vector update equations [8]. The vector update equations are derived from forward and backward predictions. This paper considers algorithms based on the updating of backward prediction errors which are known to be numerically robust [7].

A major limitation of the FQRD-RLS algorithms is the unavailability of an explicit weight vector term. Therefore, the applications are limited to output error based (e.g., noise or echo cancellation), or those that require a decision-feedback estimate of the training signal (e.g., continuously updated equalizer). In a burst-training application the adaptation is performed in a training block. The weight vector obtained at the end of the training block is used for processing the useful data. The absence of the weights in the FQRD-RLS makes the switch from the training mode to the data processing mode nontrivial.

In the following we show how the available internal variables related to the FQRD-RLS implicit weight vector can be frozen at any time instant and used for output filtering of a different data sequence. The results of the filtering will yield the same result as if the output is due to filtering with the weights obtained using any RLS algorithm such as the stable $\mathcal{O}(N^2)$ inverse QRD-RLS [9].

II. FQRD-RLS Algorithm

In this section we present the basic concepts of the QRD-RLS algorithm and one version of the FQRD-RLS algorithms, based on backward prediction error update [7], [8] to aid the explanation of the proposed fixed filtering idea.

A. Basic concepts of QR decomposition algorithms

The RLS algorithm minimizes the following cost function

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

where λ is the forgetting factor and $\mathbf{e}(k) \in \mathbb{R}^{(k+1)\times 1}$ is the *a posteriori* error vector given as

$$\mathbf{e}(k) = \mathbf{d}(k) - \mathbf{X}(k)\mathbf{w}(k) \tag{2}$$

where $\mathbf{d}(k) \in \mathbb{R}^{(k+1)\times 1}$ is the desired signal vector, $\mathbf{X}(k) \in \mathbb{R}^{(k+1)\times N}$ is the input data matrix, and $\mathbf{w}(k) \in \mathbb{R}^{N\times 1}$. The QRD-RLS algorithm uses an orthogonal rotation matrix $\mathbf{Q}(k)$ to triangularize matrix $\mathbf{X}(k)$ as [2]

$$\begin{bmatrix} \mathbf{0}_{(k+1-N)\times N} \\ \mathbf{U}(k) \end{bmatrix} = \mathbf{Q}(k)\mathbf{X}(k) \tag{3}$$

where $\mathbf{U}(k) \in \mathbb{R}^{N\times N}$ is the Cholesky factor of the deterministic autocorrelation matrix $\mathbf{R}(k) = \mathbf{X}^T(k)\mathbf{X}(k)$. Pre-multiplying (2) with $\mathbf{Q}(k)$ gives

$$\mathbf{Q}(k)\mathbf{e}(k) = \begin{bmatrix} \mathbf{e}_{q1}(k) \\ \mathbf{e}_{q2}(k) \end{bmatrix} = \begin{bmatrix} \mathbf{d}_{q1}(k) \\ \mathbf{d}_{q2}(k) \end{bmatrix} - \begin{bmatrix} \mathbf{0}_{(k+1-N)\times N} \\ \mathbf{U}(k) \end{bmatrix} \mathbf{w}(k) \tag{4}$$

The cost function in (1) is minimized by choosing $\mathbf{w}(k)$ such that $\mathbf{d}_{q2}(k) - \mathbf{U}(k)\mathbf{w}(k)$ is zero. The QRD-RLS algorithm updates vector $\mathbf{d}_{q2}(k)$ and matrix $\mathbf{U}(k)$ as

$$\begin{bmatrix} e_{q1}(k) \\ \mathbf{d}_{q2}(k) \end{bmatrix} = \mathbf{Q}_\theta(k) \begin{bmatrix} d(k) \\ \lambda^{1/2}\mathbf{d}_{q2}(k-1) \end{bmatrix} \tag{5}$$

and

$$\begin{bmatrix} \mathbf{0}_{1\times N} \\ \mathbf{U}(k) \end{bmatrix} = \mathbf{Q}_\theta(k) \begin{bmatrix} \mathbf{x}^{\mathrm{T}}(k) \\ \lambda^{1/2}\mathbf{U}(k-1) \end{bmatrix} \tag{6}$$

where $\mathbf{Q}_\theta(k) \in \mathbb{R}^{(N+1)\times(N+1)}$ is a sequence of Givens rotation matrices which annihilates the input vector $\mathbf{x}(k)$ in (6) and is partitioned as [4]

$$\mathbf{Q}_\theta(k) = \begin{bmatrix} \gamma(k) & \mathbf{g}^{\mathrm{T}}(k) \\ \mathbf{f}(k) & \mathbf{E}(k) \end{bmatrix} \tag{7}$$

The QRD-RLS algorithm is complete with the definition of the *a priori* error value $e(k) = e_{q1}(k)/\gamma(k)$ where $\gamma(k)$ is a scalar found in matrix $\mathbf{Q}_\theta(k)$, see (7).

B. The FQR_PRI_B algorithm

The idea of the FQRD-RLS algorithm is to replace the matrix update equation (6) with a vector update equation. The *a priori* error is given as

$$\begin{aligned} e(k) &= d(k) - \mathbf{w}^{\mathrm{T}}(k-1)\mathbf{x}(k) \\ &= d(k) - \mathbf{d}_{q2}^{\mathrm{T}}(k-1)\underbrace{\mathbf{U}^{-\mathrm{T}}(k-1)\mathbf{x}(k)}_{\lambda^{1/2}\mathbf{a}(k)} \end{aligned} \quad (8)$$

where $\mathbf{g}(k) = -\gamma(k)\mathbf{a}(k)$.

In the FQR_PRI_B algorithm [7], [8], the update equation for vector $\mathbf{a}(k)$ is used. The update equation, obtained by using forward and backward prediction equations and applying rotation matrices to triangularize the data matrix, is given by

$$\begin{bmatrix} \frac{e_b(k)}{\lambda^{1/2}\|\mathbf{e}_b(k-1)\|} \\ \mathbf{a}(k) \end{bmatrix} = \mathbf{Q}_{\theta f}(k-1)\begin{bmatrix} \mathbf{a}(k-1) \\ \frac{e_f(k)}{\lambda^{1/2}\|\mathbf{e}_f(k-1)\|} \end{bmatrix} \quad (9)$$

The FQR_PRI_B algorithm is given at the beginning of Table II. See [5] for implementation details.

III. Output filtering for burst-type training

This section defines the problem, states the limitations of the FQRD-RLS and provides the derivation of the proposed approaches.

A. System description

The output filtering problem under consideration is illustrated in Figure 1. As can be seen from the figure, the adaptive filter for time instants $k < k_f$ is updated using its input and desired signal pair $\{x(k), d(k)\}$; we call it *training mode*. At time instant $k = k_f$, the adaptive process is stopped and from there onwards the coefficient vector at hand $\mathbf{w}(k_f)$ is frozen and used for filtering, with a possibly different input sequence, i.e., $\tilde{x}(k)$; 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

$$y(k) = \begin{cases} \mathbf{w}^{\mathrm{T}}(k-1)\mathbf{x}(k) & k < k_f \\ \mathbf{w}_f^{\mathrm{T}}\tilde{\mathbf{x}}(k) & k \geq k_f \end{cases} \quad (10)$$

where $\mathbf{w}_f = \mathbf{w}(k_f - 1)$ is the coefficient vector of the adaptive filter "frozen" at a time instant immediately before $k = k_f$ and $\tilde{x}(k)$ is the input signal for the time instant $k \geq k_f$. This approach of output filtering is unrealizable using FQRD-RLS algorithm considering its current form.

In the proposed method, the FQRD-RLS algorithm is used during the training mode. In the next subsection, we describe how output filtering is carried out in the data mode using the implicit weights of the FQRD-RLS algorithm obtained at $k = k_f - 1$.

B. Lemmas for equivalent output-filtering

The variables from the FQRD-RLS update at time instant $k = k_f - 1$ can be reused to reproduce the equivalent output signal. For this purpose, the output after *weight freezing* is written as

$$\begin{aligned} y(k) &= \underbrace{\mathbf{d}_{q2}^{\mathrm{T}}(k_f-1)\mathbf{U}^{-\mathrm{T}}(k_f-1)}_{\mathbf{w}^{\mathrm{T}}(k_f-1)}\tilde{\mathbf{x}}(k) \\ &= \mathbf{d}_{q2}^{\mathrm{T}}(k_f-1)\mathbf{r}(k), \ \ k \geq k_f \end{aligned} \quad (11)$$

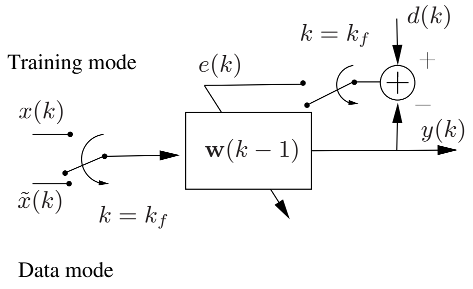

Fig. 1. The output filtering with fixed weights using adaptive filter setup.

where, $\mathbf{d}_{q2}(k_f-1)$ and $\mathbf{U}^{-\mathrm{T}}(k_f-1)$ are parameters of the FQRD-RLS at time instant $k = k_f - 1$, respectively, and $\mathbf{r}(k)$ from Eq. (11) is $\mathbf{U}^{-\mathrm{T}}(k_f-1)\tilde{\mathbf{x}}(k)$. The following lemmas are required in order to compute Eq. (11) without explicitly computing matrix $\mathbf{U}^{-\mathrm{T}}(k_f-1)$ (the proofs are given in the Appendix).

Lemma 1: *Let $\mathbf{x}(k) \in \mathbb{R}^{N\times 1}$ be the input data vector and $\mathbf{u}_{r,i}(k) \in \mathbb{R}^{N\times 1}$ denote the i^{th} column of the upper triangular matrix $\mathbf{U}^{-1}(k) \in \mathbb{R}^{N\times N}$. Given $\mathbf{Q}_\theta(k-1) \in \mathbb{R}^{(N+1)\times(N+1)}$ from Table II, then $\mathbf{U}^{-\mathrm{T}}(k-2)\mathbf{x}(k)$ can be obtained from $\mathbf{U}^{-\mathrm{T}}(k-1)\mathbf{x}(k)$ using the relation below*

$$\begin{bmatrix} 0 \\ \lambda^{-1/2}\mathbf{U}^{-\mathrm{T}}(k-2)\mathbf{x}(k) \end{bmatrix} = \mathbf{Q}_\theta^{\mathrm{T}}(k-1)\begin{bmatrix} z_k \\ \mathbf{U}^{-\mathrm{T}}(k-1)\mathbf{x}(k) \end{bmatrix} \quad (12)$$

where $z_k = -\mathbf{f}^{\mathrm{T}}(k-1)\mathbf{U}^{-\mathrm{T}}(k-1)\mathbf{x}(k)/\gamma(k-1)$ and is unknown a priori.

Lemma 2: *Let $\mathbf{x}(k) \in \mathbb{R}^{N\times 1}$ be the input data vector and $\mathbf{u}_{r,i}(k) \in \mathbb{R}^{N\times 1}$ denote the i^{th} column of the upper triangular matrix $\mathbf{U}^{-1}(k) \in \mathbb{R}^{N\times N}$. Given $\mathbf{Q}_{\theta f}(k-1) \in \mathbb{R}^{(N+1)\times(N+1)}$ from Table II, then $\mathbf{U}^{-\mathrm{T}}(k-1)\mathbf{x}(k)$ can be obtained from $\mathbf{U}^{-\mathrm{T}}(k-2)\mathbf{x}(k-1)$ using the following relation*

$$\begin{bmatrix} * \\ \mathbf{U}^{-\mathrm{T}}(k-1)\mathbf{x}(k) \end{bmatrix} = \mathbf{Q}_{\theta f}(k-1)\begin{bmatrix} \mathbf{U}^{-\mathrm{T}}(k-2)\mathbf{x}(k-1) \\ * \end{bmatrix} \quad (13)$$

where $*$ denote variables unknown a priori that can be solved for recursively, and $\mathbf{u}_{r,N-1}^{\mathrm{T}}(k)\mathbf{x}(k) = \frac{x(k)}{\|e_f^{(0)}(k-1)\|}$.

In the next two subsection we describe two approaches that make use of the above lemmas to filter any input sequence with the FQRD-RLS weight vector.

C. Approach 1: Output filtering without weight extraction

This section explains an output filtering approach that does not require the explicit knowledge of the FQRD-RLS weight vector. If we substitute $\mathbf{U}^{-\mathrm{T}}(k-1)$ with $\mathbf{U}^{-\mathrm{T}}(k_f-1)$ and $\mathbf{x}(k)$ with $\tilde{\mathbf{x}}(k)$ in the Lemmas, we get

$$\begin{bmatrix} 0 \\ \lambda^{-1/2}\tilde{\mathbf{r}}(k-1) \end{bmatrix} = \mathbf{Q}_\theta^{\mathrm{T}}(k_f-1)\begin{bmatrix} * \\ \mathbf{r}(k-1) \end{bmatrix} \quad (14)$$

and

$$\begin{bmatrix} * \\ \mathbf{r}(k) \end{bmatrix} = \mathbf{Q}_{\theta f}(k_f-1)\begin{bmatrix} \tilde{\mathbf{r}}(k-1) \\ * \end{bmatrix} \quad (15)$$

where $\tilde{\mathbf{r}}(k-1) = \mathbf{U}^{-\mathrm{T}}(k_f-2)\tilde{\mathbf{x}}(k-1)$ is an intermediate value required to get $\mathbf{r}(k)$. At a time instant after $k = k_f$ we have

$$\mathbf{r}(k-1) = \mathbf{U}^{-\mathrm{T}}(k_f-1)\tilde{\mathbf{x}}(k-1) \quad (16)$$

and applying Eq. (14) and (15) in succession we get

$$\mathbf{r}(k) = \mathbf{U}^{-\mathrm{T}}(k_f-1)\tilde{\mathbf{x}}(k) \quad (17)$$

The output $y(k)$ can then be obtained by using the updated $\mathbf{r}(k)$ in Eq. (11). Note that when $\mathbf{r}(k)$ is obtained from $\mathbf{r}(k-1)$, only the

TABLE I

COMPUTATIONAL COMPLEXITY IN TERMS OF NUMBER OF OPERATIONS.

ALGORITHMS	MULT	DIV	SQRT
training mode			
FQR_PRI_B ($0 \leq k < k_f$)	$19N+4$	$4N+1$	$2N+1$
Inverse QRD-RLS	$3N^2+2N+1$	$2N$	N
data mode			
Approach 1	$8N-7$	$2N$	0
Approach 2 ($k \leq k_f+N$)	$16N-6-13i$	1	0
Approach 2 ($k > k_f+N$)	N	0	0
Inverse QRD-RLS	N	0	0

TABLE II

EQUIVALENT OUTPUT-FILTERING METHOD FOR BURST TYPE TRAINING

Conventional FQRD-RLS algorithm with input signal $x(k)$	
for each $0 \leq k \leq k_f$ Obtaining $\mathbf{d}_{fq2}(k)$: $\begin{bmatrix} e_{fq1}(k) \\ \mathbf{d}_{fq2}(k) \end{bmatrix} = \mathbf{Q}_\theta(k-1) \begin{bmatrix} x(k) \\ \lambda^{1/2}\mathbf{d}_{fq2}(k-1) \end{bmatrix}$ Obtaining $\mathbf{a}(k)$: $\begin{bmatrix} \frac{e_b(k)}{\lambda^{1/2}\lVert\mathbf{e}_b(k-1)\rVert} \\ \mathbf{a}(k) \end{bmatrix} = \mathbf{Q}_{\theta f}(k-1) \begin{bmatrix} \mathbf{a}(k-1) \\ \frac{e_f(k)}{\lambda^{1/2}\lVert\mathbf{e}_f(k-1)\rVert} \end{bmatrix}$ Obtaining $\lVert\mathbf{e}_f(k)\rVert$: $\lVert\mathbf{e}_f(k)\rVert = \sqrt{e^2_{fq1}(k) + \lambda\lVert\mathbf{e}_f(k-1)\rVert^2}$ Obtaining $\mathbf{Q}_{\theta f}(k)$: $\begin{bmatrix} \mathbf{0} \\ \lVert\mathbf{e}_f^{(0)}(k)\rVert \end{bmatrix} = \mathbf{Q}_{\theta f}(k) \begin{bmatrix} \mathbf{d}_{fq2}(k) \\ \lVert\mathbf{e}_f(k)\rVert \end{bmatrix}$ Obtaining $\mathbf{Q}_\theta(k)$: $\begin{bmatrix} 1/\gamma(k) \\ \mathbf{0} \end{bmatrix} = \mathbf{Q}_\theta(k) \begin{bmatrix} 1 \\ -\mathbf{a}(k) \end{bmatrix}$ Joint Process Estimation: $\begin{bmatrix} e_{q1}(k) \\ \mathbf{d}_{q2}(k) \end{bmatrix} = \mathbf{Q}_\theta(k) \begin{bmatrix} d(k) \\ \lambda^{1/2}\mathbf{d}_{q2}(k-1) \end{bmatrix}$ $e(k) = e_{q1}(k)/\gamma(k)$	
Output filtering Approach 1	Output filtering Approach 2
Initialization: $\tilde{\mathbf{r}}(k_f) = \mathbf{0}$ **for** each $k > k_f$ Obtaining $\mathbf{r}(k)$: $\begin{bmatrix} * \\ \mathbf{r}(k) \end{bmatrix} = \mathbf{Q}_{\theta f}(k_f) \begin{bmatrix} \tilde{\mathbf{r}}(k_f) \\ * \end{bmatrix}$ Updating $\tilde{\mathbf{r}}(k)$: $\begin{bmatrix} 0 \\ \lambda^{-1/2}\tilde{\mathbf{r}}(k) \end{bmatrix} = \mathbf{Q}_\theta^{\mathrm{T}}(k_f) \begin{bmatrix} z_k \\ \mathbf{r}(k) \end{bmatrix}$ Obtaining the output: $y(k) = \mathbf{d}^{\mathrm{T}}_{q2}(k_f)\mathbf{r}(k)$	Initialization: $w_{f,-1}(k_f) = -1$ Obtaining $\mathbf{f}$ and $\gamma(k_f)$: $\begin{bmatrix} \gamma \\ \mathbf{f} \end{bmatrix} = \mathbf{Q}_\theta \begin{bmatrix} 1 \\ \mathbf{0}_{N\times 1} \end{bmatrix}$ **for** each $k \leq k_f + N + 1$ Get $\mathbf{u}_k(k_f)$ from $\mathbf{u}_{k-1}(k_f-1)$: $\begin{bmatrix} \frac{-w_{b,k}}{\lVert\mathbf{e}_b\rVert} \\ \mathbf{u}_k \end{bmatrix} = \tilde{\mathbf{Q}}_{\theta f} \begin{bmatrix} \mathbf{u}_{k-1} \\ \frac{-w_{f,k-1}}{\lVert\mathbf{e}_f\rVert} \end{bmatrix}$ Get $\mathbf{u}_k(k_f-1)$ from $\mathbf{u}_k(k_f)$: $z_k = -\mathbf{f}^{\mathrm{T}}\mathbf{u}_k/\gamma$ $\begin{bmatrix} 0 \\ \lambda^{-1/2}\mathbf{u}_k \end{bmatrix} = \mathbf{Q}_\theta^{\mathrm{T}} \begin{bmatrix} z_k \\ \mathbf{u}_k \end{bmatrix}$ Obtaining $w_{f,k}(k_f)$: $w_{f,k} = \mathbf{u}_k^{\mathrm{T}}\mathbf{d}_{fq2}$ Obtaining weights $w_k(k_f)$: $w_k = \mathbf{u}_k^{\mathrm{T}}\mathbf{d}_{q2}$ Obtaining output $y(k)$: $\tilde{y}(k) = \sum_{i=0}^{k-1} w_f(k_f)\tilde{x}(k-i)$ **for** each $k > k_f + N + 1$ Obtaining output $y(k)$: $y(k) = \mathbf{w}^{\mathrm{T}}\tilde{\mathbf{x}}(k)$

input vector $\tilde{\mathbf{x}}(k)$ is updated and the matrix $\mathbf{U}^{-\mathrm{T}}(k_f-1)$ remains the same in the process. The computational complexity of the algorithms and the detailed equations are provided in Tables I and II, respectively.

D. Approach 2: Output filtering with distributed weight flushing

Another approach is based on the observation that the output during the first N iterations in data mode is given by

$$\tilde{y}(k) = \sum_{i=0}^{k-1} w_i(k_f-1)\tilde{x}(k-i);\ k \leq N \tag{18}$$

As can be noted from Eq. (18), it is only necessary to extract the weights in a distributed manner, i.e., one weight per each new incoming sample. Such "on-the-fly" extraction provides us with all the weights after N iterations, and still produces the correct output values before all the weights are acquired (according to Eq. (18)). After the initial N iterations in the data mode, the output is simply given by $\mathbf{w}^{\mathrm{T}}(k_f-1)\tilde{\mathbf{x}}(k)$. Invoking Lemmas 1 and 2 using a unit pulse (as in [10]) in parallel with the output filtering in Eq. (18) will sequence out the weights $w_i(k_f-1)$ at the time instant it shows up in Eq. (18). In other words, it is not necessary to make all the weights available before starting filtering in data mode (peak complexity $\mathcal{O}(N^2)$ [10]). This distributed weight flushing procedure ensures a peak complexity of $\mathcal{O}(N)$.

The algorithm for the filtering operation is given in Table II (Approach 2). As can be seen from the table, variable z_k in this approach is not computed recursively (see [10] for details). The computational complexity of this approach is given in Table I. For the N initial iterations the computational complexity decreases linearly from $16N-19$ to $3N-6$. After that all the weights are obtained, the complexity is only N multiplications per iteration for the rest of the data burst. The peak complexity in terms of number of operations required per iteration for the proposed method and the inverse QRD-RLS algorithm are given in Table I for comparison.

IV. SIMULATIONS

The channel equalization example is taken from [11], where the channel taps are $\begin{bmatrix}0.5 & 1.2 & 1.5 & -1\end{bmatrix}^{\mathrm{T}}$. The SNR is 30 dB and the equalizer has $N = 35$ coefficients. The equalizer runs in two modes: the training mode and the data mode. In the training mode 150 symbols from 4-QAM are used whereas in the data mode 750 symbols from a 16-QAM constellation are processed. For comparison, the inverse QRD-RLS algorithm [9] is implemented along with the output filtering based FQRD-RLS algorithm. The MSE of the algorithms are shown in Figure 2, averaged over 100 runs. It can be seen that both the algorithms converge to the same solution. Also, the constellation diagram in the data mode is given in Figure 3 for both algorithms. The constellation points achieved with the inverse QRD-RLS and the proposed methods cannot be distinguished in the figure (difference lower than -250 dB). The squared weight-difference of the weights of the inverse QRD-RLS algorithm and those extracted using Approach 2 was calculated and averaged over $K = 100$ ensemble using

$$\Delta\bar{w}_i = \frac{1}{K}\sum_{j=0}^{K-1}\left[w^j_{IQRD,i} - w^j_{FQRD,i}\right]^2 \tag{19}$$

where j is the simulation run and i corresponds to the coefficient number of the respective algorithm. Figure 4 verifies that both algorithms acheive the same solution.

V. CONCLUSIONS

This paper showed how to reuse the internal variables of the fast QRD-RLS (FQRD-RLS) algorithm to perform output filtering of a different data sequence than the one related to the FQRD-RLS update recursions. The techniques presented here facilitate new applications of the FQRD-RLS algorithm, which are different from the standard output-error type applications. The new output filtering technique was used in tandem with the FQRD-RLS algorithm to design a burst-trained equalizer for which a data sequence is processed using the implicit weights of the FQRD-RLS algorithm. The results were compared with that of using a design based on the inverse QRD-RLS algorithm. It was verified that identical results are obtained using the proposed design method at a much lower computational cost.

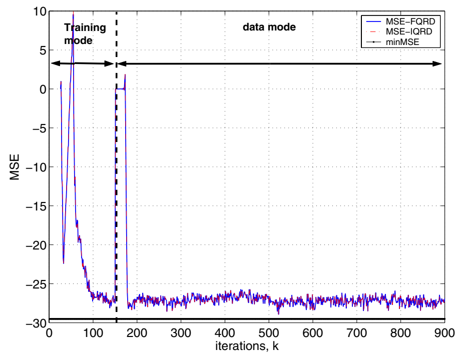

Fig. 2. Learning curves.

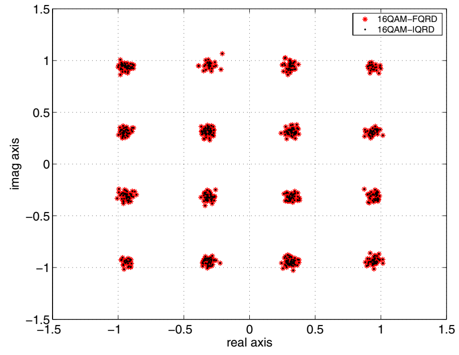

Fig. 3. Equalized signal in data mode.

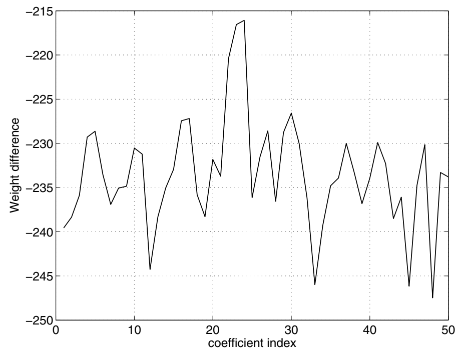

Fig. 4. Weight difference of the algorithms.

Appendix

A. Proof of Lemma 1

It can be shown that the following relation holds for the QRD-RLS algorithms [9]

$$\begin{bmatrix} \mathbf{z}^{\mathrm{T}}(k-1) \\ \mathbf{U}^{-\mathrm{T}}(k-1) \end{bmatrix} = \mathbf{Q}_\theta(k-1) \begin{bmatrix} \mathbf{0}^{\mathrm{T}} \\ \lambda^{-1/2}\mathbf{U}^{-\mathrm{T}}(k-2) \end{bmatrix} \tag{20}$$

Pre-multiplying (20) with $\mathbf{Q}_\theta^{\mathrm{T}}(k-1)$ followed by post-multiplication with $\mathbf{x}(k)$ leads to

$$\begin{bmatrix} 0 \\ \lambda^{-1/2}\mathbf{U}^{-\mathrm{T}}(k-2)\mathbf{x}(k) \end{bmatrix} = \mathbf{Q}_\theta^{\mathrm{T}}(k-1) \begin{bmatrix} \mathbf{z}^{T}(k-1)\mathbf{x}(k) \\ \mathbf{U}^{-\mathrm{T}}(k-1)\mathbf{x}(k) \end{bmatrix} \tag{21}$$

where $\mathbf{z}(k) = -\frac{\mathbf{f}^{\mathrm{T}}(\mathbf{k})\mathbf{U}^{-\mathrm{T}}(k)}{\gamma(k)}$ and $\mathbf{z}^{\mathrm{T}}(k-1)\mathbf{x}(k)$ corresponds to the variable z_k in Equation (12) which is unknown *a priori*.

Using (12) and (7), the value of the unknown z_k can be computed from known values with the following expression

$$z_k = -\mathbf{f}^{\mathrm{T}}(k-1)\mathbf{U}^{-\mathrm{T}}(k-1)\mathbf{x}(k)/\gamma(k-1) \tag{22}$$

where the elements of vector $\mathbf{f}(k-1)$ are given by $f_j(k-1) = \sin\theta_{N-j-1}(k-1)\prod_{i=0}^{N-j-2}\cos\theta_i(k-1)$ with angles $\sin\theta_j(k-1)$ and $\cos\theta_j(k-1)$ associated with the Givens rotation matrices. The vector can then be stored for further use. A more efficient approach used in practice to solve the expression is to consider $\mathbf{Q}_\theta(k)$ as a sequence of Givens rotation matrices. Each rotation matrix is then applied one by one and the expression can be solved for recursively. See [5] for similar implementation.

B. Proof of Lemma 2

Combining the definition of $\mathbf{a}(k)$ with (9), we get

$$\begin{bmatrix} \frac{e_b(k)}{\|\mathbf{e}_b(k-1)\|} \\ \mathbf{U}^{-\mathrm{T}}(k-1)\mathbf{x}(k) \end{bmatrix} = \mathbf{Q}_{\theta f}(k-1) \begin{bmatrix} \mathbf{U}^{-\mathrm{T}}(k-2)\mathbf{x}(k-1) \\ \frac{e_f(k)}{\|\mathbf{e}_f(k-1)\|} \end{bmatrix} \tag{23}$$

We proceed to show how Equation (23) can be evaluated without the *a priori* knowledge of $\frac{e_f(k)}{\|\mathbf{e}_f(k-1)\|}$ and $\frac{e_b(k)}{\|\mathbf{e}_b(k-1)\|}$. Matrix $\mathbf{Q}_{\theta f}(k)$ can be written as a sequence of Givens rotation matrices. With the last element of vector $\mathbf{U}^{-\mathrm{T}}(k-2)\mathbf{x}(k-1)$ known, the Givens rotation matrices are applied one by one and the solution can be obtained in a recursive manner.

Acknowledgment

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

References

[1] J. Cioffi, "Limited-precision effects in adaptive filtering," *IEEE Trans. Circuits and Syst.*, vol. 34, no. 7, pp. 821 – 833, Jul. 1987.
[2] P. S. R. Diniz, *Adaptive Filtering: Algorithms and Practical Implementation*. Norwell, MA, USA: Kluwer Academic Press, 1997.
[3] J. M. Cioffi, "The fast adaptive ROTOR's RLS algorithm," *IEEE Trans. on Acoust., Speech, Signal Processing*, vol. 38, pp. 631–653, April 1990.
[4] J. A. Apolinário Jr., P. S. R. Diniz, "A new fast QR algorithm based on a priori errors," *IEEE Signal Processing Lett.*, vol. 4, no. 11, pp. 307–309, Nov. 1997.
[5] J. A. Apolinário Jr., M. G. Siqueira, P. S. R. Diniz, "On fast QR algorithms based on backward prediction errors: New results and comparisons," in *Proceedings of the First Balkan Conference on Signal Processing, Communications, Circuits, and Systems*, Istanbul, Turkey, June 2000.
[6] M. G. Bellanger, "The FLS-QR algorithm for adaptive filtering," *Signal Processing*, vol. 17, pp. 291–304, March 1984.
[7] M. D. Miranda, M. Gerken, "A hybrid least squares QR-lattice algorithm using a priori errors," *IEEE Trans. Signal Processing*, vol. 45, no. 12, pp. 2900–2911, Dec. 1997.
[8] A. A. Rontogiannis, S. Theodoridis, "New fast inverse QR least squares adaptive algorithms," *ICASSP*, Detroit, MI, USA, 1995, pp. 1412–1415.
[9] S. T. Alexander, A. L. Ghirnikar, "A method for recursive least squares filtering based upon an inverse QR decomposition," *IEEE Trans. Signal Processing*, vol. 41, no. 1, pp. 20–30, Jan. 1993.
[10] M. Shoaib, S. Werner, J. A. Apolinário Jr., T. I. Laakso, "Weight extraction in FQRD-RLS algorithms," accepted in ICASSP'2006, Toulouse, France, 2006.
[11] A. Sayed, *Fundamentals of Adaptive Filtering*. NY: Wiley, 2003.