Chapter 11
Weight Extraction of Fast QRD-RLS Algorithms

Stefan Werner and Mohammed Mobien

Abstract The main limitation of fast QR-decomposition recursive least-squares (FQRD-RLS) algorithms is that they lack an explicit weight vector term. Furthermore, they do 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 of the FQRD-RLS algorithms are limited to certain output-error-based applications (e.g., noise cancelation), or to applications that can provide a decision-feedback estimate of the training signal (e.g., equalizers operating in decision-directed mode). This chapter presents some observations that allow us to apply the FQRD-RLS algorithms in applications that traditionally have required explicit knowledge of the transversal weights. Section 11.1 reviews the basic concepts of QRD-RLS and the particular FQRD-RLS algorithm that is used in the development of the new applications. Section 11.2 describes how to identify the implicit FQRD-RLS transversal weights. This allows us to use the FQRD-RLS algorithm in a system identification setup. Section 11.3 applies the FQRD-RLS algorithm to burst-trained systems, where the weight vector is updated during a training phase and then kept fixed and used for output filtering. Section 11.4 applies the FQRD-RLS algorithm for single-channel active noise control, where a copy of the adaptive filter is required for filtering a different input sequence than that of the adaptive filter. A discussion on multichannel and lattice extensions is provided in Section 11.5. Finally, conclusions are drawn in Section 11.6.

Stefan Werner
Helsinki University of Technology, Espoo – Finland
e-mail: `stefan.werner@tkk.fi`

Mohammed Mobien
Helsinki University of Technology, Espoo – Finland
e-mail: `mobien@ieee.org`

11.1 FQRD-RLS Preliminaries

To aid the presentation of the new FQRD-RLS applications, this section reviews the basic concepts of the QRD-RLS algorithm and one of the FQRD-RLS algorithms in Chapter 4, namely the FQR_POS_B algorithm.

11.1.1 QR decomposition algorithms

The RLS algorithm minimizes the following cost function:

$$\xi(k) = \sum_{i=0}^{k} \lambda^{k-i}[d(i) - \mathbf{x}^{\mathrm{T}}(i)\mathbf{w}(k)]^2 = \|\mathbf{e}(k)\|^2, \tag{11.1}$$

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

$$\mathbf{e}(k) = \mathbf{d}(k) - \mathbf{X}(k)\mathbf{w}(k), \tag{11.2}$$

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

$$\begin{bmatrix} \mathbf{0} \\ \mathbf{U}(k) \end{bmatrix} = \mathbf{Q}(k)\mathbf{X}(k), \tag{11.3}$$

where $\mathbf{U}(k) \in \mathbb{R}^{(N+1)\times(N+1)}$ is the Cholesky factor of the deterministic autocorrelation matrix $\mathbf{R}(k) = \mathbf{X}^{\mathrm{T}}(k)\mathbf{X}(k)$. Pre-multiplying (11.2) with $\mathbf{Q}(k)$ gives

$$\mathbf{Q}(k)\mathbf{e}(k) = \begin{bmatrix} \mathbf{e}_{q1}(k) \\ \mathbf{e}_{q2}(k) \end{bmatrix} = \begin{bmatrix} \mathbf{d}_{q1}(k) \\ \mathbf{d}_{q2}(k) \end{bmatrix} - \begin{bmatrix} \mathbf{0} \\ \mathbf{U}(k) \end{bmatrix} \mathbf{w}(k). \tag{11.4}$$

The cost function in (11.1) is minimized by choosing $\mathbf{w}(k)$ such that $\mathbf{d}_{q2}(k) - \mathbf{U}(k)\mathbf{w}(k)$ is zero, i.e.,

$$\mathbf{w}(k) = \mathbf{U}^{-1}(k)\mathbf{d}_{q2}(k). \tag{11.5}$$

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

$$\begin{bmatrix} e_{q1}(k) \\ \mathbf{d}_{q2}(k) \end{bmatrix} = \mathbf{Q}_\theta(k) \begin{bmatrix} d(k) \\ \lambda^{1/2}\mathbf{d}_{q2}(k-1) \end{bmatrix} \tag{11.6}$$

and

$$\begin{bmatrix} \mathbf{0} \\ \mathbf{U}(k) \end{bmatrix} = \mathbf{Q}_\theta(k) \begin{bmatrix} \mathbf{x}^{\mathrm{T}}(k) \\ \lambda^{1/2}\mathbf{U}(k-1) \end{bmatrix}, \tag{11.7}$$

where $\mathbf{Q}_\theta(k) \in \mathbb{R}^{(N+2)\times(N+2)}$ is a sequence of Givens rotation matrices which annihilates the input vector $\mathbf{x}(k)$ in (11.7) and is partitioned as [2]

$$\mathbf{Q}_\theta(k) = \begin{bmatrix} \gamma(k) & \mathbf{g}^{\mathrm{T}}(k) \\ \mathbf{f}(k) & \mathbf{E}(k) \end{bmatrix}. \tag{11.8}$$

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 (11.8).

11.1.2 FQR_POS_B algorithm

The idea of the FQRD-RLS algorithm is to replace the matrix update in (11.7) with a vector update. Using (11.5), we can express the *a posteriori* error $\varepsilon(k)$ of the adaptive filter as

$$\varepsilon(k) = d(k) - \underbrace{\mathbf{x}^{\mathrm{T}}(k)\mathbf{U}^{-1}(k)}_{\mathbf{f}^{\mathrm{T}}(k)}\mathbf{d}_{q2}(k), \tag{11.9}$$

where

$$\mathbf{f}(k) = \mathbf{U}^{-\mathrm{T}}(k)\mathbf{x}(k). \tag{11.10}$$

The FQR_POS_B algorithm, introduced in Chapter 4, updates vector $\mathbf{f}(k)$ by using forward and backward prediction equations and applying rotation matrices to triangularize the data matrix. The update is given by

$$\begin{bmatrix} \frac{\varepsilon_b(k)}{\|\mathbf{e}_b(k)\|} \\ \mathbf{f}(k) \end{bmatrix} = \mathbf{Q}'_{\theta f}(k) \begin{bmatrix} \mathbf{f}(k-1) \\ \frac{\varepsilon_f(k)}{\|\mathbf{e}_f(k)\|} \end{bmatrix}, \tag{11.11}$$

where $\varepsilon_f(k)$ and $\varepsilon_b(k)$ are the *a posteriori* forward and backward prediction errors, defined as

$$\begin{aligned} \varepsilon_f(k) &= x(k) - \mathbf{w}_f^{\mathrm{T}}(k)\mathbf{x}(k-1), \text{ and} \\ \varepsilon_b(k) &= x(k-N-1) - \mathbf{w}_b^{\mathrm{T}}(k)\mathbf{x}(k), \end{aligned} \tag{11.12}$$

which are the first elements of the corresponding forward and backward prediction errors vectors, $\mathbf{e}_f(k)$ and $\mathbf{e}_b(k)$, given by (see Chapter 3)

$$\begin{aligned}
\mathbf{e}_f(k) &= \begin{bmatrix} x(k) \\ \lambda^{1/2}x(k-1) \\ \vdots \\ \lambda^{k/2}x(0) \end{bmatrix} - \begin{bmatrix} \mathbf{X}(k-1) \\ \mathbf{0}_{1\times(N+1)} \end{bmatrix} \mathbf{w}_f(k) = \mathbf{d}_f(k) - \begin{bmatrix} \mathbf{X}(k-1) \\ \mathbf{0}_{1\times(N+1)} \end{bmatrix} \mathbf{w}_f(k), \\
\mathbf{e}_b(k) &= \begin{bmatrix} x(k-N-1) \\ \lambda^{1/2}x(k-N-2) \\ \vdots \\ \lambda^{(k-N-1)/2}x(0) \\ \mathbf{0}_{(N+1)\times 1} \end{bmatrix} - \mathbf{X}(k)\mathbf{w}_b(k) = \mathbf{d}_b(k) - \mathbf{X}(k)\mathbf{w}_b(k).
\end{aligned} \tag{11.13}$$

As seen in Chapter 4, vector $\mathbf{w}_f(k)$ is not explicitly used for updating $\mathbf{f}(k)$. Instead, the term $\varepsilon_f(k)/\|\mathbf{e}_f(k)\|$ in (11.11) is recursively computed by the FQRD-RLS algorithm. The methods presented in the following will make explicit use of vector $\mathbf{w}_f(k)$. Therefore, if we consider $\mathbf{e}_f(k)$ in (11.13), then the vector $\mathbf{w}_f(k)$ that minimizes $\|\mathbf{e}_f(k)\|^2$ is given by

$$\mathbf{w}_f(k) = \left[\mathbf{X}^{\mathrm{T}}(k-1)\mathbf{X}(k-1)\right]^{-1} \begin{bmatrix} \mathbf{X}(k-1) \\ \mathbf{0}_{1\times(N+1)} \end{bmatrix}^{\mathrm{T}} \mathbf{d}_f(k). \tag{11.14}$$

Let $\mathbf{Q}_f(k-1)$ denotes the Givens rotation matrix defined as below.

$$\underbrace{\begin{bmatrix} \mathbf{Q}(k-1) & \mathbf{0}_{k\times 1} \\ \mathbf{0}_{1\times k} & 1 \end{bmatrix}^{\mathrm{T}}}_{\mathbf{Q}_f^{\mathrm{T}}(k-1)} \underbrace{\begin{bmatrix} \mathbf{Q}(k-1) & \mathbf{0}_{k\times 1} \\ \mathbf{0}_{1\times k} & 1 \end{bmatrix}}_{\mathbf{Q}_f(k-1)} = \mathbf{I}, \tag{11.15}$$

where $\mathbf{Q}(k-1)$ is the matrix that triangularizes $\mathbf{X}(k-1)$, see (11.4). Applying $\mathbf{Q}(k-1)$ and $\mathbf{Q}_f(k-1)$ to (11.14) yields

$$\begin{aligned}
\mathbf{w}_f(k) &= \left[\mathbf{X}^{\mathrm{T}}(k-1)\mathbf{Q}^{\mathrm{T}}(k-1)\mathbf{Q}(k-1)\mathbf{X}(k-1)\right]^{-1} \\
&\quad \times \begin{bmatrix} \mathbf{X}(k-1) \\ \mathbf{0}_{1\times(N+1)} \end{bmatrix}^{\mathrm{T}} \mathbf{Q}_f^{\mathrm{T}}(k-1)\mathbf{Q}_f(k-1)\mathbf{d}_f(k) \\
&= \left[\mathbf{U}^{\mathrm{T}}(k-1)\mathbf{U}(k-1)\right]^{-1}\mathbf{U}^{\mathrm{T}}(k-1)\mathbf{d}_{fq2}(k) \\
&= \mathbf{U}^{-1}(k-1)\mathbf{d}_{fq2}(k),
\end{aligned} \tag{11.16}$$

where $\mathbf{d}_{f_{q2}}(k)$ corresponds to the last $N+1$ elements (not taking into account the last element which corresponds to $\lambda^{k/2}x(0)$) of the rotated weighted forward error vector defined as $\mathbf{e}_{f_q} = \mathbf{Q}_f(k-1)\mathbf{e}_f(k)$.

Equations (11.5), (11.11), (11.12), and (11.16) are essential for the understanding of the weight extraction (WE) mechanism and output filtering methods explained in the following.

11.2 System Identification with FQRD-RLS

In many applications, it is necessary to identify an unknown system. Examples of such applications are: identification of the acoustic echo path in acoustic echo cancelation, channel identification in communications systems, and active noise control (ANC) [1]. Figure 11.1 shows the basic structure of a system-identification application, where $x(k)$, $y(k)$, $d(k)$, and $e(k)$ are the input, output, desired output, and error signals of the adaptive filter, respectively, at time instant k. The adaptive filter and the unknown system share the same input signal, usually a wideband signal in the case of channel identification or a noisy voice signal in the case of acoustic echo cancelation. The adaptation algorithm compares the desired signal with the output of the adaptive filter in order to minimize the chosen objective function. The desired signal will, in addition to the output from the unknown system, contain some measurement noise $n(k)$ which will affect the variance of the estimate of the unknown system.

Let us now consider two possible approaches for identifying the unknown plant: one using an inverse QRD-RLS (IQRD-RLS) algorithm, with complexity $\mathscr{O}[N^2]$ per iteration, and another one that uses an FQRD-RLS algorithm with complexity $\mathscr{O}[N]$ per iteration. Obviously, the latter approach requires a mechanism in which the transversal weights embedded in the FQRD-RLS variables can be identified at any iteration of interest, e.g., after convergence. If the transversal weights are not required at every iteration, which might be the case in some applications, and the cost of extracting the FQRD-RLS weights is reasonably low, the overall computational complexity would be much lower with the second approach.

The goal of this section is to develop a WE (identification) mechanism that can be used in tandem with the FQRD-RLS algorithm at any particular iteration of interest. This would reduce the overall computational cost (and peak-complexity) of the system identification.

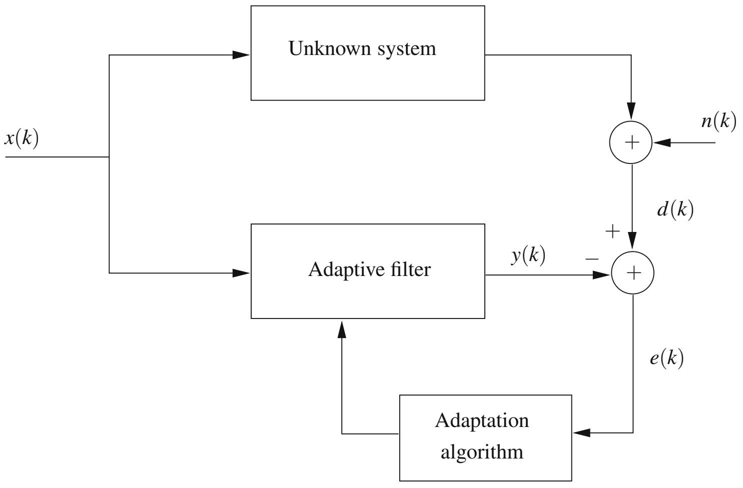


Fig. 11.1 Schematic diagram of a system-identification application.

11.2.1 Weight extraction in the FQRD-RLS algorithm

Note that the reduced computational cost of the FQRD-RLS algorithm is due to the fact that the matrix update Equation (11.7) is replaced with the vector update Equation (11.11). Thus, we are no longer able to separate $\mathbf{U}^{-1}(k)$ from $\mathbf{x}(k)$. As a consequence, if we excite the algorithm with a unit impulse, like in the serial weight flushing technique in [3, 4], (11.9) will not sequence out the correct coefficients.

To approach the solution, we note from (11.5) that the *i*th coefficient of vector $\mathbf{w}(k)$ is given by

$$w_i(k) = \mathbf{d}_{q2}^{\mathrm{T}}(k)\mathbf{u}_i(k), \tag{11.17}$$

where $\mathbf{u}_i(k)$ denotes the *i*th column of matrix $\mathbf{U}^{-\mathrm{T}}(k)$. This means that when $\mathbf{d}_{q2}(k)$ is given, the elements of the weight vector $\mathbf{w}(k)$ can be computed if all the columns of matrix $\mathbf{U}^{-\mathrm{T}}(k)$ are known. In the following, we show how all the column vectors $\mathbf{u}_i(k)$ can be obtained in a serial manner given $\mathbf{u}_0(k)$. The main result is that the column vector $\mathbf{u}_i(k)$ can be obtained from the column vector $\mathbf{u}_{i-1}(k)$ using two relations (denoted by $\rightarrow$):

$$\mathbf{u}_{i-1}(k) \rightarrow \mathbf{u}_{i-1}(k-1) \rightarrow \mathbf{u}_i(k).$$

Let us first look at the relation $\mathbf{u}_{i-1}(k-1) \rightarrow \mathbf{u}_i(k)$. That is, assume that we are given the $(i-1)$th column of $\mathbf{U}^{-\mathrm{T}}(k-1)$ (please note the index value $k-1$). We can then compute the *i*th column of $\mathbf{U}^{-\mathrm{T}}(k)$ using (11.11) as stated in Lemma 1.

Lemma 1. Let $\mathbf{u}_i(k) \in \mathbb{R}^{(N+1)\times 1}$ denote the *i*th column of the upper triangular matrix $\mathbf{U}^{-\mathrm{T}}(k) \in \mathbb{R}^{(N+1)\times(N+1)}$. Given $\mathbf{Q}'_{\theta f}(k) \in \mathbb{R}^{(N+2)\times(N+2)}$, $\mathbf{d}_{fq2}(k) \in \mathbb{R}^{(N+1)\times 1}$, and $\|\mathbf{e}_f(k)\|$ from the FQRD-RLS algorithm, then $\mathbf{u}_i(k)$ can be obtained from $\mathbf{u}_{i-1}(k-1)$ using the following relation:

$$\begin{bmatrix} * \\ \mathbf{u}_i(k) \end{bmatrix} = \mathbf{Q}'_{\theta f}(k) \begin{bmatrix} \mathbf{u}_{i-1}(k-1) \\ \frac{-w_{f,i-1}(k)}{\|\mathbf{e}_f(k)\|} \end{bmatrix}, \; i = 0, \ldots, N, \tag{11.18}$$

where $*$ is a "don't-care" and

$$w_{f,i-1}(k) = \begin{cases} -1 & \text{for } i = 0, \\ \mathbf{u}_{i-1}^{\mathrm{T}}(k-1)\mathbf{d}_{fq2}(k) & \text{otherwise.} \end{cases} \tag{11.19}$$

Equation (11.18) is initialized with $\mathbf{u}_{-1}(k-1) = \mathbf{0}_{(N+1)\times 1}$.

The definitions in (11.10) and (11.12) allow us to rewrite (11.11) as

$$\begin{bmatrix} \frac{-\mathbf{w}_b^{\mathrm{T}}(k)}{\|\mathbf{e}_b(k)\|} & \frac{1}{\|\mathbf{e}_b(k)\|} \\ \mathbf{U}^{-\mathrm{T}}(k) & \mathbf{0} \end{bmatrix} \mathbf{x}^{(N+2)}(k) = \mathbf{Q}'_{\theta f}(k) \begin{bmatrix} \mathbf{0} & \mathbf{U}^{-\mathrm{T}}(k-1) \\ \frac{1}{\|\mathbf{e}_f(k)\|} & \frac{-\mathbf{w}_f^{\mathrm{T}}(k)}{\|\mathbf{e}_f(k)\|} \end{bmatrix} \mathbf{x}^{(N+2)}(k), \quad (11.20)$$

where $\mathbf{x}^{(N+2)}(k) = [x(k)\ x(k-1)\ \cdots\ x(k-N-1)]^{\mathrm{T}}$.

Equation (11.20) is illustrated in Figure 11.2, where we see that (11.18) and (11.19) are just the column description of the matrices multiplying the extended input vector $\mathbf{x}^{(N+2)}(k)$. To account for the first column of (11.20) we initialize with $\mathbf{u}_{-1}(k-1) = \mathbf{0}_{N\times 1}$ and $w_{f,-1}(k) = -1$, which can be clearly seen from Figure 11.2.

Consequently, the first coefficient can be computed directly following the initialization as

$$w_0(k) = \mathbf{d}_{q2}^{\mathrm{T}}(k)\mathbf{u}_0(k). \quad (11.21)$$

To proceed with the next coefficient $w_1(k) = \mathbf{d}_{q2}^{\mathrm{T}}(k)\mathbf{u}_1(k)$, we need vector $\mathbf{u}_0(k-1)$ to compute $\mathbf{u}_1(k)$ using (11.18). In general, having computed $w_i(k)$, i.e., $\mathbf{u}_i(k)$ is known, we need a reverse mechanism, $\mathbf{u}_i(k) \rightarrow \mathbf{u}_i(k-1)$, in tandem with (11.18), (11.19) to allow for the computation of the next weight $w_{i+1}(k)$. How to compute $\mathbf{u}_i(k-1)$ from $\mathbf{u}_i(k)$ is summarized by Lemma 2.

Equations (11.22) and (11.23) are obtained directly from the update equation for $\mathbf{U}^{-\mathrm{T}}(k-1)$, see Chapter 3.

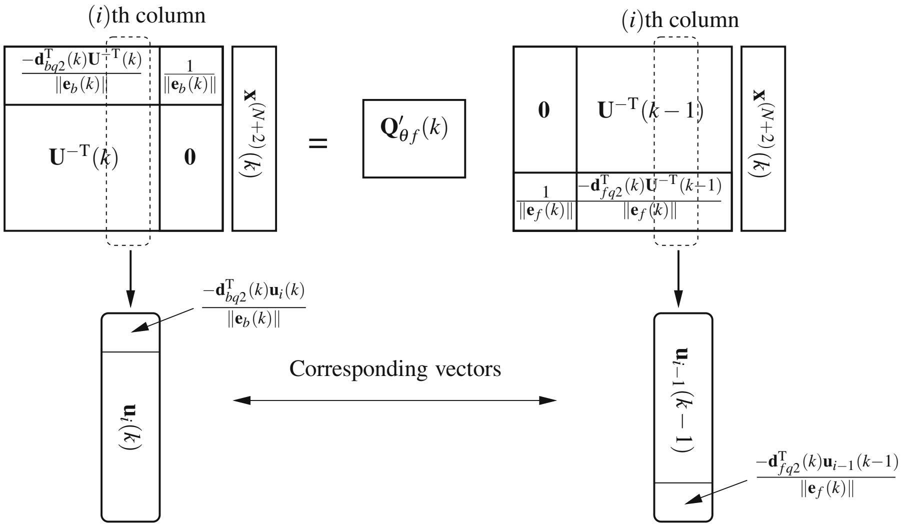

(i)th column
$\frac{-\mathbf{d}_{bq2}^{\mathrm{T}}(k)\mathbf{U}^{-\mathrm{T}}(k)}{\|\mathbf{e}_b(k)\|}$
$\frac{1}{\|\mathbf{e}_b(k)\|}$
$\mathbf{U}^{-\mathrm{T}}(k)$
$\mathbf{0}$
$\mathbf{x}^{(N+2)}(k)$
=
$\mathbf{Q}'_{\theta f}(k)$
(i)th column
$\mathbf{0}$
$\mathbf{U}^{-\mathrm{T}}(k-1)$
$\frac{1}{\|\mathbf{e}_f(k)\|}$
$\frac{-\mathbf{d}_{fq2}^{\mathrm{T}}(k)\mathbf{U}^{-\mathrm{T}}(k-1)}{\|\mathbf{e}_f(k)\|}$
$\mathbf{x}^{(N+2)}(k)$
$\frac{-\mathbf{d}_{bq2}^{\mathrm{T}}(k)\mathbf{u}_i(k)}{\|\mathbf{e}_b(k)\|}$
$\mathbf{u}_i(k)$
Corresponding vectors
$\mathbf{u}_{i-1}(k-1)$
$\frac{-\mathbf{d}_{fq2}^{\mathrm{T}}(k)\mathbf{u}_{i-1}(k-1)}{\|\mathbf{e}_f(k)\|}$

Fig. 11.2 Illustration of (11.18) on how to obtain the ith column of $\mathbf{U}^{-\mathrm{T}}(k)$ (denoted as $\mathbf{u}_i(k)$) from the $(i-1)$th column of $\mathbf{U}^{-\mathrm{T}}(k-1)$ (denoted as $\mathbf{u}_{i-1}(k-1)$).

Lemma 2. Let $\mathbf{u}_i(k) \in \mathbb{R}^{(N+1)\times 1}$ denote the ith column of the upper triangular matrix $\mathbf{U}^{-\mathrm{T}}(k) \in \mathbb{R}^{(N+1)\times(N+1)}$. Given $\mathbf{Q}_\theta(k) \in \mathbb{R}^{(N+2)\times(N+2)}$, $\mathbf{f}(k) \in \mathbb{R}^{(N+1)\times 1}$ and $\gamma(k)$ from the FQRD-RLS algorithm, then $\mathbf{u}_i(k-1)$ can be obtained from $\mathbf{u}_i(k)$ using the relation below

$$\begin{bmatrix} 0 \\ \lambda^{-1/2}\mathbf{u}_i(k-1) \end{bmatrix} = \mathbf{Q}_\theta^{\mathrm{T}}(k) \begin{bmatrix} z_i(k) \\ \mathbf{u}_i(k) \end{bmatrix},\ i = 0,\ldots,N-1, \tag{11.22}$$

where

$$z_i(k) = -\mathbf{f}^{\mathrm{T}}(k)\mathbf{u}_i(k)/\gamma(k). \tag{11.23}$$

The update is given by [5]

$$\begin{bmatrix} \mathbf{z}^{\mathrm{T}}(k) \\ \mathbf{U}^{-\mathrm{T}}(k) \end{bmatrix} = \mathbf{Q}_\theta(k) \begin{bmatrix} \mathbf{0}^{\mathrm{T}} \\ \lambda^{-1/2}\mathbf{U}^{-\mathrm{T}}(k-1) \end{bmatrix}. \tag{11.24}$$

Equation (11.22) is obtained by pre-multiplying both sides of (11.24) with $\mathbf{Q}_\theta^{\mathrm{T}}(k)$ and considering each column separately, $z_i(k)$ being the ith element of vector $\mathbf{z}(k)$. If we now employ the standard partition of $\mathbf{Q}_\theta(k)$ in (11.8), we can directly verify that the first element of (11.22) is equal to

$$0 = \gamma(k)z_i(k) + \mathbf{f}^{\mathrm{T}}(k)\mathbf{u}_i(k), \tag{11.25}$$

which gives the relation in (11.23).

In summary, (11.18) and (11.22) allow for a serial WE of the weights $\mathbf{w}(k)$ embedded in the FQRD-RLS variables. The pseudo-code for the WE mechanism is given in Table 11.1. The number of operations required to completely extract all the coefficients is given in Table 11.2. For comparison, the computational costs of the FQRD-RLS and the IQRD-RLS algorithms are also given.

11.2.2 Example

The FQRD-RLS and IQRD-RLS algorithms, both using $\lambda = 0.95$, were used to identify a system of order $N = 10$. The input signal $x(k)$ was a noise sequence, colored by filtering a zero-mean white Gaussian noise sequence $n_x(k)$ through the fourth-order IIR filter $x(k) = n_x(k) + x(k-1) + 1.2x(k-2) + 0.95x(k-3)$, and the SNR was set to 30 dB. After 1900 iterations, the internal variables that are required for computing $\mathbf{w}(k)$ were saved. The transversal weight vector of the IQRD-RLS

Table 11.1 FQRD-RLS WE algorithm.

Weight extraction

```
Initialization:
$x_i = 0, \forall i \in [1, N]$
$x_0 = -1$
$\mathbf{u}_{-1}(k-1) = \mathbf{0}_{(N+1)\times 1}$
Available from the FQRD-RLS algorithm:
$\mathbf{Q}'_{\theta f}(k)$, $\mathbf{d}_{fq2}(k)$, $\mathbf{f}(k)$, $\|\mathbf{e}_f(k)\|$, $\mathbf{Q}_\theta(k)$, $\gamma(k)$, and $\mathbf{d}_{q2}(k)$

for each $i = 0$ to $N$
{
  Compute $\mathbf{u}_i(k)$ from $\mathbf{u}_{i-1}(k-1)$:
    $w_{f,i-1}(k) = x_i - \mathbf{u}_{i-1}^{\mathrm{T}}(k-1)\mathbf{d}_{fq2}(k)$
    $\begin{bmatrix} * \\ \mathbf{u}_i(k) \end{bmatrix} = \mathbf{Q}'_{\theta f}(k) \begin{bmatrix} \mathbf{u}_{i-1}(k-1) \\ \frac{-w_{f,i-1}(k)}{\|\mathbf{e}_f(k)\|} \end{bmatrix}$

  Compute $\mathbf{u}_i(k-1)$ from $\mathbf{u}_i(k)$:
    $z_i(k) = -\mathbf{f}^{\mathrm{T}}(k)\mathbf{u}_i(k)/\gamma(k)$
    $\begin{bmatrix} 0 \\ \lambda^{-1/2}\mathbf{u}_i(k-1) \end{bmatrix} = \mathbf{Q}_\theta^{\mathrm{T}}(k) \begin{bmatrix} z_j(k) \\ \mathbf{u}_i(k) \end{bmatrix}$

  Compute $w_i(k)$:
    $w_i(k) = \mathbf{d}_{q2}^{\mathrm{T}}(k)\mathbf{u}_i(k)$
}
```

Table 11.2 Computational complexity of WE [8].

Algorithm	Mult.	Div.	SQRT
FQRD-RLS	$19N+23$	$4N+5$	$2N+3$
WE (per weight i, $0 \le i \le N$)	$11N+14-11i$	0	0
WE (total)	$5.5N^2+19.5N+7$	0	0
IQRD-RLS	$3N^2+8N+4$	$2N+2$	$N+1$

algorithm and the weight vector extracted from the FQRD-RLS algorithm are compared by taking the squared difference of each coefficient, i.e.,

$$\Delta w_i^2(k) = |w_{\mathrm{FQRD},i}(k) - w_{\mathrm{IQRD},i}(k)|^2. \tag{11.26}$$

Figure 11.3 shows the learning curves and the weight difference after 1900 iterations. We see that the two algorithms have identical learning curves, and that the transversal weight vector extracted from the FQRD-RLS algorithm is identical to that of the IQRD-RLS algorithm up to the simulation precision.

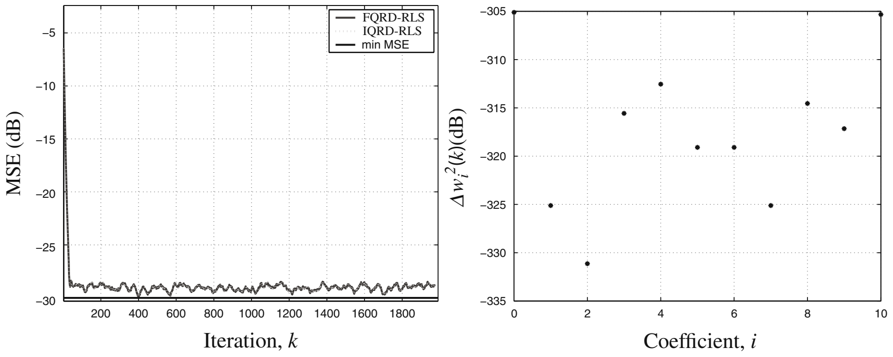


Fig. 11.3 Learning curves of the FQRD-RLS and the IQRD-RLS algorithms (left figure), squared difference between coefficient weights of the IQRD-RLS and the FQRD-RLS algorithms.

11.3 Burst-trained Equalizer with FQRD-RLS

In wireless communications systems, the main factors limiting the system capacity are various kinds of interference such as intersymbol interference (ISI) due to multipath propagation in frequency selective fading channels, co-channel (or multiple access) interference, and adjacent channel interference. ISI is the main impairment in single-user communications and can be corrected through the use of an adaptive equalizer [6]. Figure 11.4 shows the structure of an adaptive equalizer, where $u(k)$ is the user signal of interest and $i(k)$ is co-channel interference. The adaptive filter will try to suppress the channel-induced ISI, and in certain applications also the co-channel interference. The desired signal $d(k)$ is now a delayed replica of the transmitted signal, where the value of the delay D is chosen to compensate for the delay introduced by the channel.

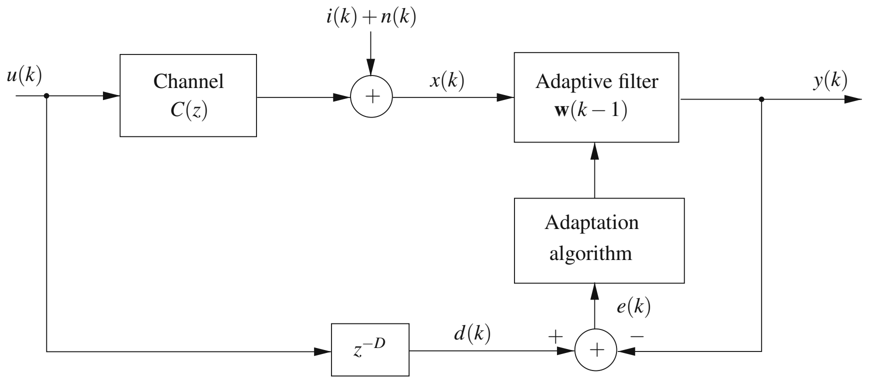


Fig. 11.4 Schematic diagram of an adaptive equalizer.

In this section, we will examine the special case of burst-trained equalizers, where the equalizer coefficients are periodically updated using known training symbols and then used for fixed filtering of a useful data sequence.

11.3.1 Problem description

The problem under consideration is illustrated in Figure 11.5. During time instants $k \leq k_f$, the equalizer operates in *training mode* and its coefficients are updated using the input and desired signal pair $\{\mathbf{x}(k), d(k)\}$. At time instant $k = k_f$, the adaptive process is stopped and the equalizer switches to *data mode* where the coefficient vector obtained during the training mode is kept fixed. That is, the output of the filter is given by

$$y(k) = \begin{cases} \mathbf{w}^{\mathrm{T}}(k-1)\mathbf{x}(k) & k \leq k_f \\ \mathbf{w}^{\mathrm{T}}(k_f)\mathbf{x}(k) & k > k_f \end{cases} \tag{11.27}$$

where $\mathbf{w}(k_f)$ is the coefficient vector of the adaptive filter "frozen" at time instant $k = k_f$.

If the FQRD-RLS algorithm is employed during training, one alternative for carrying out the filtering during the data mode, $k > k_f$, is to first extract the filter coefficients according to Section 11.2 and, thereafter, perform the filtering of $\mathbf{x}(k)$ with a simple transversal structure. To avoid the increased peak complexity $\mathscr{O}[N^2]$ of this solution (at time k_f), we seek here alternative methods with reduced peak complexity $\mathscr{O}[N]$ that can reproduce the output signal in (11.27) from the variables of the FQRD-RLS algorithm available at instant $k = k_f$ [7].

11.3.2 Equivalent-output filtering

After the training of the FQRD-RLS algorithm is stopped ($k = k_f$), filtering of the useful signal should be carried out according to (compare with (11.27))

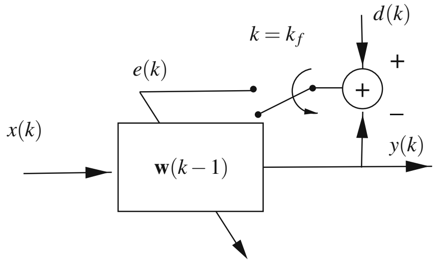


Fig. 11.5 Operation of a burst-trained equalizer. The adaptive filter coefficients are updated during *training mode* ($k \leq k_f$), and kept fixed and used for output filtering in *data mode* ($k > k_f$). Note that there is no weight update after $k > k_f$.

$$y(k) = \underbrace{\mathbf{d}_{q2}^{\mathrm{T}}(k_f)\mathbf{U}^{-\mathrm{T}}(k_f)}_{\mathbf{w}^{\mathrm{T}}(k_f)}\mathbf{x}(k),\ k > k_f, \tag{11.28}$$

where $\mathbf{d}_{q2}(k_f)$ and $\mathbf{U}^{-\mathrm{T}}(k_f)$ are parameters of the FQRD-RLS algorithm at time instant $k = k_f$, respectively. Vector $\mathbf{d}_{q2}(k_f)$ in (11.28) is explicitly available in the FQRD-RLS algorithm. However, knowledge of $\mathbf{U}^{-\mathrm{T}}(k_f)$ is only provided through the variable $\mathbf{f}(k_f) = \mathbf{U}^{-\mathrm{T}}(k_f)\mathbf{x}(k_f)$. Thus, starting with $\mathbf{f}(k_f)$ as an initial value, we need to find a way to obtain vector $\mathbf{U}^{-\mathrm{T}}(k_f)\mathbf{x}(k)$ from vector $\mathbf{U}^{-\mathrm{T}}(k_f)\mathbf{x}(k-1)$, i.e., we need to incorporate the new sample $x(k)$ without affecting $\mathbf{U}^{-\mathrm{T}}(k_f)$ in the process.

This problem is somewhat similar to the WE problem that was treated in the previous section. The solution exploits the following two steps:

$$\mathbf{U}^{-\mathrm{T}}(k_f)\mathbf{x}(k-1) \rightarrow \mathbf{U}^{-\mathrm{T}}(k_f-1)\mathbf{x}(k-1) \rightarrow \mathbf{U}^{-\mathrm{T}}(k_f)\mathbf{x}(k).$$

The first step, summarized by Lemma 3, is obtained by pre-multiplying (11.24) with $\mathbf{Q}_\theta^{\mathrm{T}}(k_f)$ and post-multiplying with the vector $[x(k)\ \mathbf{x}^{\mathrm{T}}(k-1)]^{\mathrm{T}}$. The variable $z(k)$ in (11.31) is obtained in a similar manner as (11.24). That is, by employing the partition of $\mathbf{Q}_\theta(k_f)$ in (11.8), the equation describing the first element of (11.30) is given by

$$0 = \gamma(k_f)z(k) + \mathbf{f}^{\mathrm{T}}(k_f)\mathbf{U}^{-\mathrm{T}}(k_f)\mathbf{x}(k-1). \tag{11.29}$$

Lemma 3. Let $\mathbf{U}^{-\mathrm{T}}(k_f) \in \mathbb{R}^{(N+1)\times(N+1)}$ denote the upper triangular inverse transpose Cholesky matrix corresponding to time instant k_f, and $\mathbf{x}(k-1) \in \mathbb{R}^{(N+1)\times 1}$ be the input vector at any instant $k > k_f$. Given $\mathbf{Q}_\theta(k_f) \in \mathbb{R}^{(N+2)\times(N+2)}$, $\mathbf{f}(k_f) \in \mathbb{R}^{(N+1)\times 1}$, and $\gamma(k_f)$ from the FQRD-RLS algorithm, then $\mathbf{U}^{-\mathrm{T}}(k_f-1)\mathbf{x}(k-1)$ is obtained from $\mathbf{U}^{-\mathrm{T}}(k_f)\mathbf{x}(k-1)$ using the relation below

$$\begin{bmatrix} 0 \\ \lambda^{-1/2}\mathbf{U}^{-\mathrm{T}}(k_f-1)\mathbf{x}(k-1) \end{bmatrix} = \mathbf{Q}_\theta^{\mathrm{T}}(k) \begin{bmatrix} z(k) \\ \mathbf{U}^{-\mathrm{T}}(k_f)\mathbf{x}(k-1) \end{bmatrix}, \tag{11.30}$$

where

$$z(k) = -\mathbf{f}^{\mathrm{T}}(k_f)\mathbf{U}^{-\mathrm{T}}(k_f)\mathbf{x}(k-1)/\gamma(k_f). \tag{11.31}$$

The second step, summarized by Lemma 4, is obtained from (11.20) and freezing the FQRD-RLS variables at $k = k_f$, i.e.,

$$\begin{bmatrix} \frac{-\mathbf{w}_b^{\mathrm{T}}(k_f)}{\|\mathbf{e}_b(k_f)\|} & \frac{1}{\|\mathbf{e}_b(k_f)\|} \\ \mathbf{U}^{-\mathrm{T}}(k_f) & \mathbf{0} \end{bmatrix} \mathbf{x}^{(N+2)}(k) = \mathbf{Q}'_{\theta f}(k_f) \begin{bmatrix} \mathbf{0} & \mathbf{U}^{-\mathrm{T}}(k_f-1) \\ \frac{1}{\|\mathbf{e}_f(k_f)\|} & \frac{-\mathbf{w}_f^{\mathrm{T}}(k_f)}{\|\mathbf{e}_f(k_f)\|} \end{bmatrix} \mathbf{x}^{(N+2)}(k). \tag{11.32}$$

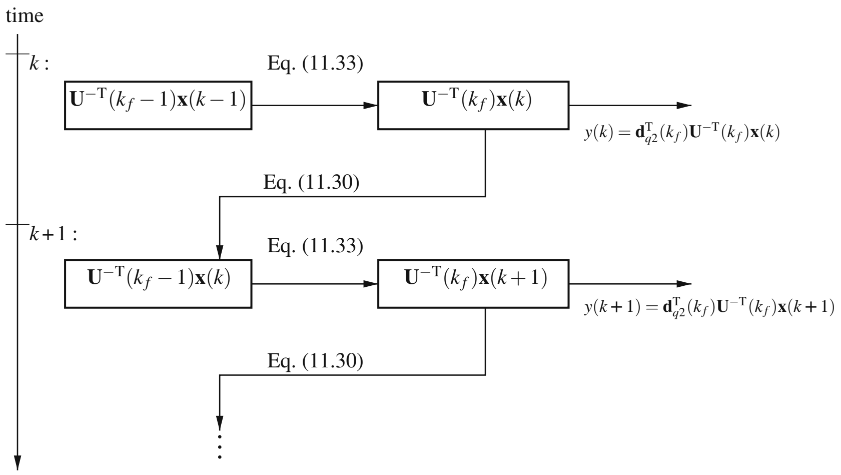
We see that the extended input vector $\mathbf{x}^{(N+2)}(k)$ multiplies both sides of (11.32). Therefore, the time instant for $\mathbf{x}^{(N+2)}(k)$ can be chosen arbitrarily.

Lemma 4. Let $\mathbf{U}^{-\mathrm{T}}(k_f-1) \in \mathbb{R}^{(N+1)\times(N+1)}$ denote the upper triangular inverse transpose Cholesky matrix corresponding to time instant k_f-1, and $\mathbf{x}(k-1) \in \mathbb{R}^{(N+1)\times 1}$ be the input vector at any instant $k > k_f$. Given $\mathbf{Q}'_{\theta f}(k_f) \in \mathbb{R}^{(N+2)\times(N+2)}$, $\mathbf{d}_{fq2}(k_f) \in \mathbb{R}^{(N+1)\times 1}$, and $\|\mathbf{e}_f(k_f)\|$ from the FQRD-RLS algorithm and input sample $x(k)$, then $\mathbf{U}^{-\mathrm{T}}(k_f)\mathbf{x}(k)$ is obtained from $\mathbf{U}^{-\mathrm{T}}(k_f-1)\mathbf{x}(k-1)$ as

$$\begin{bmatrix} * \\ \mathbf{U}^{-\mathrm{T}}(k_f)\mathbf{x}(k) \end{bmatrix} = \mathbf{Q}'_{\theta f}(k_f) \begin{bmatrix} \mathbf{U}^{-\mathrm{T}}(k_f-1)\mathbf{x}(k-1) \\ \frac{x(k)-\mathbf{d}^{\mathrm{T}}_{fq2}(k_f)\mathbf{U}^{-\mathrm{T}}(k_f-1)\mathbf{x}(k-1)}{\|\mathbf{e}_f(k_f)\|} \end{bmatrix}, \tag{11.33}$$

where $*$ is a "don't-care" variable.

In summary, (11.30) and (11.33) allow us to reproduce the equivalent-output signal corresponding to (11.28) without explicit knowledge of the weights embedded in the FQRD-RLS algorithm. The procedure is illustrated in Figure 11.6. The pseudo-code of the equivalent-output filtering algorithm is provided in Table 11.3.



time
k :
Eq. (11.33)
$\mathbf{U}^{-\mathrm{T}}(k_f-1)\mathbf{x}(k-1)$
$\mathbf{U}^{-\mathrm{T}}(k_f)\mathbf{x}(k)$
$y(k) = \mathbf{d}^{\mathrm{T}}_{q2}(k_f)\mathbf{U}^{-\mathrm{T}}(k_f)\mathbf{x}(k)$
Eq. (11.30)
$k+1$:
Eq. (11.33)
$\mathbf{U}^{-\mathrm{T}}(k_f-1)\mathbf{x}(k)$
$\mathbf{U}^{-\mathrm{T}}(k_f)\mathbf{x}(k+1)$
$y(k+1) = \mathbf{d}^{\mathrm{T}}_{q2}(k_f)\mathbf{U}^{-\mathrm{T}}(k_f)\mathbf{x}(k+1)$
Eq. (11.30)

Fig. 11.6 Fixed filtering without explicit knowledge of weight vector embedded in the FQRD-RLS algorithm.

11.3.3 Equivalent-output filtering with explicit weight extraction

The approach presented here is based on the observation that for $k > k_f$ we can divide the input vector $\mathbf{x}(k)$ into two non-overlapping vectors $\mathbf{c}(k) \in \mathbb{C}^{(N+1)\times 1}$ and $\mathbf{v}(k) \in \mathbb{C}^{(N+1)\times 1}$

Table 11.3 Reproducing output $y(k) = \mathbf{w}^{\mathrm{T}}(k_f)\mathbf{x}(k)$ $(k > k_f)$ from the FQRD-RLS variables.

Equivalent-output filtering

```
Initialization:
$\mathbf{r}(k) = \mathbf{f}(k_f) \equiv \mathbf{U}^{-\mathrm{T}}(k_f)\mathbf{x}(k_f)$
Available from the FQRD-RLS algorithm:
$\mathbf{Q}'_{\theta f}(k_f)$, $\mathbf{d}_{fq2}(k_f)$, $\mathbf{f}(k_f)$, $\|\mathbf{e}_f(k_f)\|$, $\mathbf{Q}_\theta(k_f)$, $\gamma(k_f)$, and $\mathbf{d}_{q2}(k_f)$

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

  Compute $\mathbf{U}^{-\mathrm{T}}(k_f)\mathbf{x}(k)$ from $\mathbf{U}^{-\mathrm{T}}(k_f-1)\mathbf{x}(k-1)$:
  $\begin{bmatrix} * \\ \mathbf{r}(k) \end{bmatrix} = \mathbf{Q}'_{\theta f}(k_f) \begin{bmatrix} \tilde{\mathbf{r}}(k) \\ \frac{x(k)-\mathbf{d}_{fq2}^{\mathrm{T}}(k_f)\tilde{\mathbf{r}}(k)}{\|\mathbf{e}_f(k_f)\|} \end{bmatrix}$

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

$$\mathbf{x}(k) = \mathbf{c}(k) + \mathbf{v}(k),\ k > k_f, \tag{11.34}$$

with initial values

$$\begin{aligned} \mathbf{c}(k_f) &= \mathbf{0} \\ \mathbf{v}(k_f) &= \mathbf{x}(k_f), \end{aligned} \tag{11.35}$$

where $\mathbf{c}(k)$ contains the input-samples for $k > k_f$ and $\mathbf{v}(k)$ holds those remaining, i.e., for $k \le k_f$. In other words, for each time instant k the new input-sample $x(k)$ is shifted into $\mathbf{c}(k)$ and a zero is shifted into $\mathbf{v}(k)$. The output $y(k)$ for $k > k_f$ can now be written as

$$y(k) = y_c(k) + y_v(k) = \mathbf{w}^{\mathrm{T}}(k_f)\mathbf{c}(k) + \mathbf{w}^{\mathrm{T}}(k_f)\mathbf{v}(k). \tag{11.36}$$

Note that with the initialization in (11.35), $\mathbf{v}(k) = \mathbf{0}$ and $y(k) = y_c(k)$ for $k > k_f + N$.

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 distributed weight extraction that is used with $\mathbf{c}(k)$ to produce $y_c(k)$, and; one equivalent-output part reproducing $y_v(k)$.

Reproducing $y_v(k)$ is straightforward. We need only to apply the results in previous subsection using vector $\mathbf{v}(k)$ in place of $\mathbf{x}(k)$. Obtaining $y_c(k)$ and $\mathbf{w}(k_f)$ is based on the observation that $y_c(k)$ during the first $N+1$ iterations, following the training phase ($k > k_f$), is given by

$$y_c(k) = \sum_{i=0}^{k-(k_f+1)} w_i(k_f)c(k-i), \;\; k_f < k \leq k_f + N + 1, \tag{11.37}$$

where $c(k) = x(k) \; \forall k > k_f$ and $c(k) = 0 \; \forall k \leq k_f$. We see that (11.37) allows us 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 $N+1$ iterations, and still produces the correct output $y_c(k)$ before all the weights are acquired (according to (11.37)). Invoking Lemmas 1 and 2 using a unit pulse in parallel with (11.37) will sequence out the weights $w_i(k_f)$ at the time instant they show up in (11.37).

After the initial $N+1$ iterations, the output $y(k)$ is simply given by $\mathbf{w}^{\mathrm{T}}(k_f)\mathbf{x}(k) = \mathbf{w}^{\mathrm{T}}(k_f)\mathbf{c}(k)$. In other words, it is not necessary to make all the weights available before starting filtering in data mode (peak complexity $\mathscr{O}[N^2]$ [8]). This distributed weight flushing procedure ensures a peak complexity of $\mathscr{O}[N]$.

11.3.4 Example

The channel equalization example is taken from [9], where the channel is given by

$$C(z) = 0.5 + 1.2z^{-1} + 1.5z^{-2} - z^{-3}.$$

The SNR is set to 30 dB and the order of the equalizer is $N = 49$. During the first 150 iterations (i.e., $k_f = 150$), the equalizer coefficients are updated by the FQRD-RLS algorithm. The training symbols $d(k)$ randomly generated BPSK symbols. Following the initial training sequence, an unknown symbol sequence consisting of 750 4-PAM symbols was transmitted over the channel, and the equalizer output was reproduced using the approach in Section 11.3.3.

For comparison purposes, an IQRD-RLS algorithm was also implemented. Note that the IQRD-RLS has a complexity of $\mathscr{O}[N^2]$ during coefficient adaptation.

Figure 11.7 shows the mean squared error (MSE) curves for the FQRD-RLS and the IQRD-RLS approaches. The results were obtained by averaging and smoothing 100 realizations of the experiment. It can be seen that both algorithms converge to the same solution.

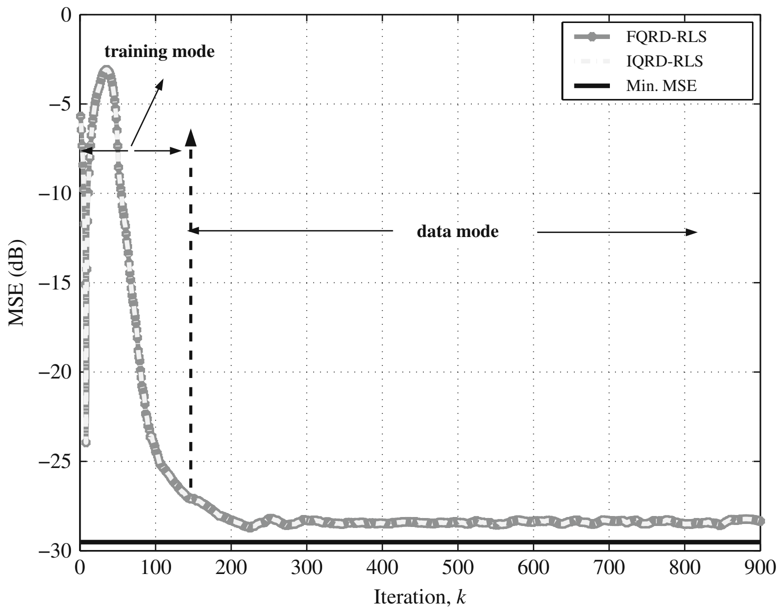


Fig. 11.7 Learning curves of the FQRD-RLS and the IQRD-RLS algorithms.

11.4 Active Noise Control and FQRD-RLS

In ANC, a disturbing (primary) noise is canceled by generating an “anti-noise” signal with identical amplitude and opposite phase [10]. Figure 11.8 shows a single-channel system consisting of one reference sensor (microphone) measuring the noise signal $x(k)$, one actuator (loudspeaker) signal $y(k)$, and one error sensor (microphone) measuring the residual signal $e(k)$. In the figure, $P(z)$ is the

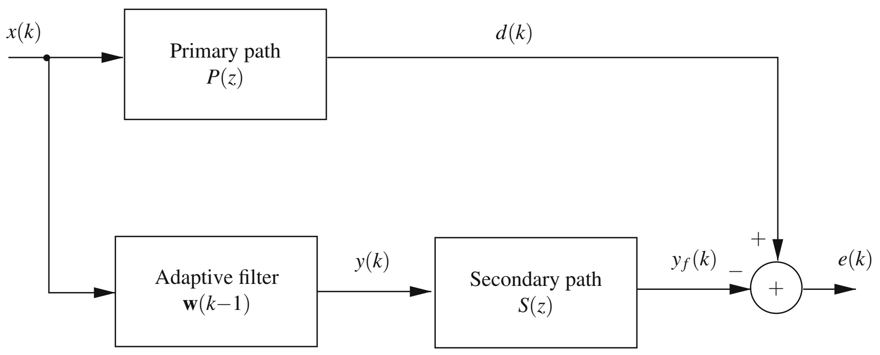


Fig. 11.8 Schematic diagram of an ANC system.

primary-path response, i.e., the (acoustic) response from the noise sensor to the error sensor, and $S(z)$ is the secondary-path response that models the acoustic path between the actuator and error microphone as well as other imperfections like D/A and A/D converters, reconstruction filters, and power amplifier effects [10]. Assuming that $S(z)$ is known, the task of the adaptive filter is to identify the unknown primary path $P(z)$.

11.4.1 Filtered-x RLS

The error signal $e(k)$ observed by the error microphone is given by

$$e(k) = d(k) - y_f(k) = d(k) - s(k) * \underbrace{[\mathbf{x}^{\mathrm{T}}(k)\mathbf{w}(k-1)]}_{y(k)}, \tag{11.38}$$

where $*$ denotes convolution, $s(k)$ is the impulse response of the secondary path $S(z)$, and $y(k) = \sum_{i=0}^{N} x(k-i)w_i(k-1) = x(k) * w$, $w_i(k-1)$ being the ith element of $\mathbf{w}(k-1)$ and w representing the impulse response of the adaptive filter at $k-1$. Knowing that $s(k) * [x(k) * w] = [s(k) * x(k)] * w$, we define the filtered input signal vector

$$\mathbf{x}_f(k) = [x_f(k)\; x_f(k-1)\; \cdots\; x_f(k-N)]^{\mathrm{T}}, \tag{11.39}$$

whose elements are given as delayed versions of $x_f(k) = s(k) * x(k)$. We can now formulate the WLS objective function as

$$J_{\mathbf{w}} = \sum_{i=0}^{k} \lambda^{k-i}[d(i) - \mathbf{x}_f{}^{\mathrm{T}}(i)\mathbf{w}(k)]^2. \tag{11.40}$$

Differentiating the objective function $J_{\mathbf{w}}$ with respect to $\mathbf{w}(k)$ and solving for the minimum results in $\mathbf{w}(k) = \mathbf{R}_f^{-1}(k)\mathbf{p}_f(k)$, where $\mathbf{R}_f(k)$ and $\mathbf{p}_f(k)$ are defined by

$$\begin{aligned}\mathbf{R}_f(k) &= \sum_{i=0}^{k} \lambda^{k-i}\mathbf{x}_f(i)\mathbf{x}_f^{\mathrm{T}}(i), \text{ and}\\ \mathbf{p}_f(i) &= \sum_{i=0}^{k} \lambda^{k-i}\mathbf{x}_f(i)d(i).\end{aligned} \tag{11.41}$$

The *filtered-x* RLS (FX-RLS) algorithm is then obtained in a similar manner as the conventional RLS algorithm in Chapter 2 by substituting $\mathbf{R}_f(k)$ and $\mathbf{p}_f(k)$ in $\mathbf{R}_f^{-1}(k)\mathbf{p}_f(k)$ with their recursive formulations

$$\mathbf{R}_f(k) = \lambda\mathbf{R}_f(k) + \mathbf{x}_f(k)\mathbf{x}_f{}^{\mathrm{T}}(k), \text{ and} \tag{11.42}$$

$$\mathbf{p}_f(k) = \lambda\mathbf{p}_f(k) + d(k)\mathbf{x}_f(k), \tag{11.43}$$

giving the following updating expression

$$\mathbf{w}(k) = \mathbf{w}(k-1) + e(k)\mathbf{R}_f{}^{-1}(k)\mathbf{x}_f(k). \tag{11.44}$$

The inverse $\mathbf{R}_f^{-1}(k)$ can be obtained recursively in terms of $\mathbf{R}_f^{-1}(k-1)$ using the *matrix inversion lemma*[1] [1], thus avoiding direct inversion of $\mathbf{R}_f(k)$ at each time instant k.

Note that $\mathbf{x}_f(k)$ depends on the impulse response $s(k)$ of the secondary path $S(z)$. For most ANC systems, an estimate $\hat{S}(z)$ of $S(z)$ can be obtained offline during an initial training phase [10]. Then the filtered input signal vector $\mathbf{x}_f(k)$ used with the FX-RLS algorithm is given by

$$\mathbf{x}_f(k) = \hat{s}(k) * \mathbf{x}(k), \tag{11.45}$$

where $\hat{s}(k)$ is the impulse response of $\hat{S}(z)$.

11.4.2 Modified filtered-x FQRD-RLS

The main problems with the FX-RLS algorithm are potential divergence behavior in finite-precision environment and high-computational complexity, which is of order N^2. As an alternative, we could think of a more robust solution with reduced complexity that employs the FQRD-RLS algorithm. However, the FQRD-RLS algorithm (and also standard QRD-RLS algorithms) requires explicit knowledge of $d(k)$ to minimize the objective function in (11.40). This should be compared with the FX-RLS implementation in (11.44) that directly employs the error signal $e(k)$ measured by the error microphone. On the other hand, we see from Figure 11.8 that if we pass the actuator signal $y(k)$ through the estimated secondary-path filter $\hat{S}(z)$, i.e., we obtain $\hat{y}_f(k) = \hat{s}(k) * y(k)$, an estimate $\hat{d}(k)$ can be obtained as

$$\hat{d}(k) = e(k) + \hat{y}_f(k) = e(k) + \hat{s}(k) * y(k). \tag{11.46}$$

This leads to the realization in Figure 11.9. This structure is referred to as the *modified filtered-x structure* and has been used with conventional RLS and LMS algorithms as well as with the IQRD-RLS algorithm [11] to improve convergence speed and robustness.

We see from Figure 11.9 that the coefficient vector embedded in the FQRD-RLS variables is needed for reproducing the output signal $y(k) = \mathbf{w}^{\mathrm{T}}(k-1)\mathbf{x}(k)$. We know from Section 11.2 that $\mathbf{w}(k-1)$ can be made explicitly available at every iteration. However, as can be seen from Table 11.2, such an approach would lead to a solution of $\mathscr{O}[N^2]$ complexity per iteration. In other words, there is no obvious gain of using an 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

[1] $[\mathbf{A}+\mathbf{BCD}]^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}[\mathbf{DA}^{-1}\mathbf{B}+\mathbf{C}^{-1}]^{-1}\mathbf{DA}^{-1}$.

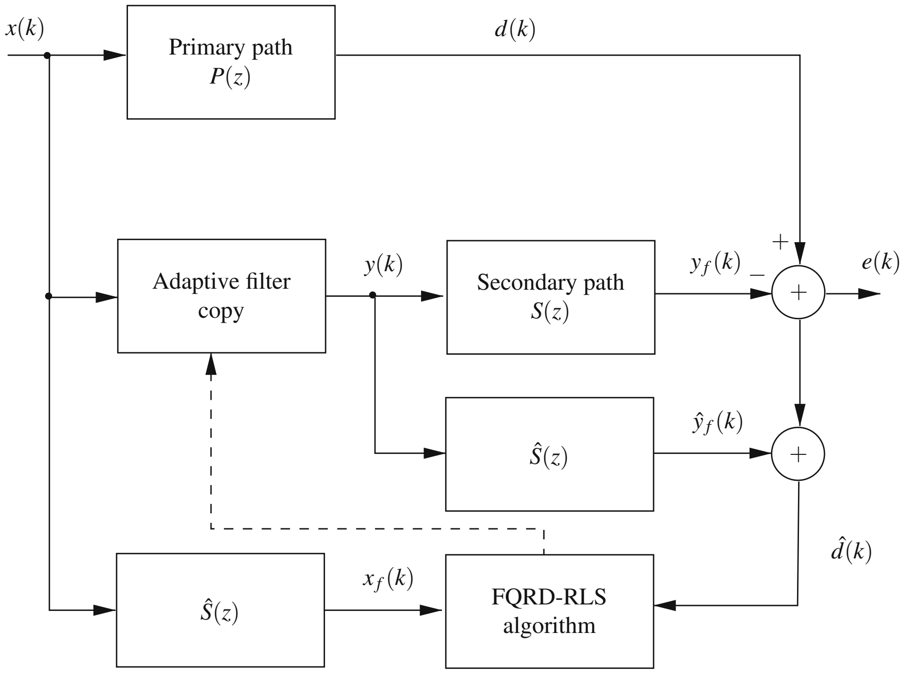


Fig. 11.9 FQRD-RLS in an ANC system.

rate, say once every K samples. This type of intermediate solution was considered in [11], where a QRD least-squares lattice (QRD-LSL) algorithm was employed in the lower branch of Figure 11.9. 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 $y(k)$ associated with the weight vector $\mathbf{w}(k-1)$ embedded in the FQRD-RLS algorithm. The total computational complexity for calculating $y(k)$ and updating $\mathbf{w}(k-1)$ will be $\mathscr{O}[N]$ per iteration. The resulting structure will yield exactly the same result as the modified filtered-x IQRD-RLS in algorithm [11], while reducing the computational complexity by an order of magnitude.

The solution is very similar to the problem dealt with in Section 11.3, where the weight vector in the 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 FQRD-RLS algorithm.

The output $y(k)$ can be expressed as

$$y(k) = \mathbf{w}^{\mathrm{T}}(k-1)\mathbf{x}(k) = \mathbf{d}_{q2}^{\mathrm{T}}(k-1)\mathbf{U}^{-\mathrm{T}}(k-1)\mathbf{x}(k), \tag{11.47}$$

where, $\mathbf{d}_{q2}(k-1)$ and $\mathbf{U}^{-\mathrm{T}}(k-1)$ are parameters of the FQRD-RLS algorithm running in the lower branch of Figure 11.9. The rotated desired signal vector $\mathbf{d}_{q2}(k)$ is directly available in the FQRD-RLS algorithm. However, the Cholesky factor matrix $\mathbf{U}^{-1}(k-1)$ is hidden in vector $\mathbf{f}(k-1) = \mathbf{U}^{\mathrm{T}}(k-1)\mathbf{x}_f(k-1)$. On the other hand, we know from (11.20) that matrix $\mathbf{Q}'_{\theta f}(k-1)$ provides the relation $\mathbf{U}^{\mathrm{T}}(k-2) \rightarrow \mathbf{U}^{\mathrm{T}}(k-1)$. Since vectors $\mathbf{x}_f(k)$ and $\mathbf{x}(k)$ are both initially set to zero, we can use (11.20) for calculating $y(k)$ for $k > 0$. The required computations are summarized by Lemma 5.

Lemma 5. Let $\mathbf{U}^{-\mathrm{T}}(k-1) \in \mathbb{R}^{(N+1)\times(N+1)}$ denote the upper triangular inverse transpose Cholesky matrix corresponding to the filtered autocorrelation matrix $\mathbf{R}_f(k-1)$ defined in (11.41), and $\mathbf{x}(k-1) \in \mathbb{R}^{(N+1)\times 1}$ be the input vector at any instant $k > 0$. Given $\mathbf{Q}'_{\theta f}(k-1) \in \mathbb{R}^{(N+2)\times(N+2)}$, $\mathbf{d}_{fq2}(k-1) \in \mathbb{R}^{(N+1)\times 1}$, and $\|\mathbf{e}_f(k-1)\|$ from the FQRD-RLS algorithm operating in the lower branch of Figure 11.9, then $\mathbf{U}^{-\mathrm{T}}(k-1)\mathbf{x}(k)$ in (11.47) obtained from $\mathbf{U}^{-\mathrm{T}}(k-2)\mathbf{x}$ $(k-1)$ is written as

$$\begin{bmatrix} * \\ \mathbf{U}^{-\mathrm{T}}(k-1)\mathbf{x}(k) \end{bmatrix} = \mathbf{Q}'_{\theta f}(k-1) \begin{bmatrix} \mathbf{U}^{-\mathrm{T}}(k-2)\mathbf{x}(k-1) \\ \frac{x(k)-\mathbf{d}^{\mathrm{T}}_{fq2}(k-1)\mathbf{U}^{-\mathrm{T}}(k-2)\mathbf{x}(k-1)}{\|\mathbf{e}_f(k)\|} \end{bmatrix}, \quad (11.48)$$

where $*$ is a "don't-care" variable.

The algorithm for reproducing the output $y(k)$ in the upper branch of Figure 11.9 is summarized in Table 11.4.

Table 11.4 Modified filtered-x FQRD-RLS algorithm.

```
Output filtering in the upper branch of Figure 11.9

Initialization:
$\mathbf{r}(0) = \mathbf{U}^{\mathrm{T}}(-1)\mathbf{x}(0) = \mathbf{0}_{(N+1)\times 1}$
Available from FQRD-RLS algorithm:
$\mathbf{Q}'_{\theta f}(k-1)$, $\mathbf{d}_{fq2}(k-1)$, and $\|\mathbf{e}_f(k-1)\|$
for each $k$
{
  Compute $\mathbf{U}^{-\mathrm{T}}(k-1)\mathbf{x}(k)$ from $\mathbf{U}^{-\mathrm{T}}(k-2)\mathbf{x}(k-1)$:
    $\begin{bmatrix} * \\ \mathbf{r}(k) \end{bmatrix} = \mathbf{Q}'_{\theta f}(k-1) \begin{bmatrix} \mathbf{r}(k-1) \\ \frac{x(k)-\mathbf{d}^{\mathrm{T}}_{fq2}(k)\mathbf{r}(k-1)}{\|\mathbf{e}_f(k-1)\|} \end{bmatrix}$

  Compute $y(k) = \mathbf{w}^{\mathrm{T}}(k-1)\mathbf{x}(k)$:
    $y(k) = \mathbf{d}^{\mathrm{T}}_{q2}(k)\mathbf{r}(k)$
}
```

11.4.3 Example

To illustrate the modified filtered-x FQRD-RLS algorithm, we consider an ANC setup where the primary-path and secondary-path filters are given as follows:

$$\begin{aligned} P(z) &= 0.4828z^{-3} - 0.8690z^{-4} + 0.0966z^{-5} + 0.0483z^{-6} \\ S(z) &= 0.8909z^{-2} + 0.4454z^{-3} + 0.0891z^{-4}. \end{aligned} \tag{11.49}$$

The order of the adaptive filter is $N = 19$, and the input signal $x(k)$ was a colored noise sequence, colored by filtering a zero-mean white Gaussian noise sequence $n_x(k)$ through the third-order IIR filter $x(k) = -1.2x(k-1) - 0.95x(k-2) + n_x(k)$. The primary signal $d(k)$ was further disturbed by an additive noise sequence $n(k)$, whose variance was set such that the SNR at the error microphone was 28 dB.

For comparison purposes, we implemented the IQRD-RLS algorithm [11] and an FQRD-RLS solution where WE is performed in the lower branch at a reduced rate, once every $K = 20$ or $K = 50$ iterations, and the extracted coefficients are then copied to the upper branch and kept fixed until the next WE takes place. Figure 11.10 shows the MSE curves for the FQRD-RLS and the IQRD-RLS implementations. The results were obtained by averaging and smoothing 50 realizations of the experiment. The FQRD-RLS and the IQRD-RLS algorithms have identical converge behavior (up to machine precision). As expected, the convergence behavior changes when considering a solution that extracts the weights at a reduced rate. In case of a non-stationary primary path (which may be common in practical applications), this difference would be more relevant.

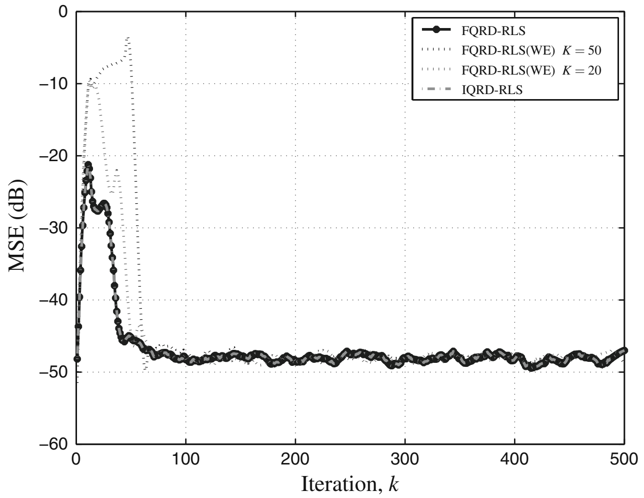


Fig. 11.10 Learning curves of the FQRD-RLS and IQRD-RLS algorithms implemented in the modified filtered-x structure. For comparison purposes, FQRD-RLS WE solutions are shown where the coefficients are extracted at a reduced rate, for every $K = 20$ (intermediary curve) or $K = 50$ (upper curve) iterations, and copied to the upper branch.

11.5 Multichannel and Lattice Implementations

The results presented in this chapter were based on single-channel adaptive FIR structures. All the results can be extended to multichannel FQRD-RLS algorithms, e.g., [9, 12–15]. This extension allows for multichannel applications such as broadband beamforming [16], Volterra system identification [9], Volterra predistorters [17], burst-trained decision-feedback (DFE) and Volterra equalizers [1], multichannel ANC [18], to name but a few.

The result for WE can also be extended to FQRD algorithms belonging to the family of least-squares lattice-based algorithms, e.g., the QRD-LSL algorithms proposed in [19, 20]. The problem of parameter identification in fast RLS algorithms has been previously addressed using the duality between the FQRD-RLS algorithms and the normalized lattice structure [19]. In other words, the WE for the fast QR-decomposition can be solved by finding the reflection coefficients and the backward prediction weight counterparts using duality [19]. It was shown in [4] and indicated in [19] that, by using the least-squares lattice version of the Levinson-Durbin recursion, the transversal weights can be obtained from the variables of the QRD-LSL algorithm at a cost of $\mathscr{O}[N^3]$ operations. In practice, the high cost of the weight identification using an exact Levinson-Durbin algorithm can be avoided by assuming algorithm convergence and infinite memory support, i.e., $\lambda = 1$ [21].The computational complexity is then reduced at the cost of accuracy of the solution. A modification of Lemma 1 in Section 11.2 was introduced in [22] to allow for the transversal weight identification at any chosen iteration with a computational complexity of $\mathscr{O}[N^2]$ without compromising accuracy. The main observation, was that the order-update required by the exact Levinson-Durbin algorithm can be done in one step by exploiting the known QRD-LSL variables, reducing the number of required operations by an order of magnitude.

11.6 Conclusion

This chapter showed how to reuse the internal variables of the fast QRD-RLS (FQRD-RLS) algorithm to enable new applications which are different to the standard output-error type applications. We first considered the problem of system identification and showed how to extract the weights in a serial manner. Thereafter, the WE results were extended to the case of burst-trained equalizers, where the equalizer is periodically re-trained using pilots and then used for fixed filtering of useful data. Finally, we considered the problem of ANC, where a modified filtered-x FQRD-RLS structure was introduced. Simulation results were compared with those using a design based on the IQRD-RLS algorithm. It was verified that, with the help of the WE techniques detailed in this chapter, identical results are obtained using the FQRD-RLS methods at a much lower computational cost.

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

References

1. P. S. R. Diniz, Adaptive Filtering: Algorithms and Practical Implementation. 3rd edition Springer, New York, NY, USA (2008)
2. J. A. Apolinário Jr. and P. S. R. Diniz, A new fast QR algorithm based on *a priori* errors. IEEE Signal Processing Letters, vol. 4, no. 11, pp. 307–309 (November 1997)
3. C. R. Ward, A. J. Robson, P. J. Hargrave, and J. G. McWhirter, Application of a systolic array to adaptive beamforming. IEE Proceedings, Part F – Communications, Radar and Signal Processing, vol. 131, no. 6, pp. 638–645 (October 1984)
4. S. Haykin, Adaptive Filter Theory. 3rd edition Prentice-Hall, Englewood Cliffs, NJ, USA (1996)
5. S. T. Alexander and A. L. Ghirnikar, A method for recursive least squares filtering based upon an inverse QR decomposition. IEEE Transactions on Signal Processing, vol. 41, no. 1, pp. 20–30 (January 1993)
6. S. U. Qureshi, Adaptive equalization. Proceedings of the IEEE, vol. 73, no. 9, pp. 1349–1387 (September 1985)
7. 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. IEEE International Symposium on Circuits and Systems, ISCAS'2006, Kos, Greece (May 2006)
8. M. Shoaib, S. Werner, J. A. Apolinário Jr., and T. I. Laakso, Solution to the weight extraction problem in FQRD-RLS algorithms. IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP'2006, Toulouse, France, pp. 572–575 (May 2006)
9. M. A. Syed and V. J. Mathews, QR-decomposition based algorithms for adaptive Volterra filtering. IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, vol. 40, no. 6, pp. 372–382 (June 1993)
10. S. M. Kuo and D. R. Morgan, Active noise control: a tutorial review. Proceedings of the IEEE, vol. 87, no. 6, pp. 943–973 (June 1999)
11. M. Bouchard, Numerically stable fast convergence least-squares algorithms for multichannel active sound cancellation systems and sound deconvolution systems. Signal Processing, vol. 82, no. 5, pp. 721–736 (May 2002)
12. A. A. Rontogiannis and S. Theodoridis, Multichannel fast QRD-LS adaptive filtering: new technique and algorithms. IEEE Transactions on Signal Processing, vol. 46, no. 11, pp. 2862–2876 (November 1998)
13. A. L. L. Ramos, J. Apolinário Jr, and M. G. Siqueira, A new order recursive multiple order multichannel fast QRD algorithm. Thirty-Eighth Annual Asilomar Conference on Signals, Systems, and Computers, Pacific Grove, USA, pp. 965–969 (November 2004)
14. C. A. Medina, J. Apolinário Jr., and M. G. Siqueira, A unified framework for multichannel fast QRD-LS adaptive filters based on backward prediction errors. IEEE International Midwest Symposium on Circuits and Systems, MWSCAS'02, Tulsa, USA, pp. 668–671 (August 2002)
15. A. L. L. Ramos, J. A. Apolinário Jr., and S. Werner, Multichannel fast QRD-LS adaptive filtering: block-channel and sequential-channel algorithms based on updating backward prediction errors. Signal Processing (Elsevier), vol. 87, pp. 1781–1798 (July 2007)
16. M. Shoaib, S. Werner, J. A. Apolinário Jr., and T. I. Laakso, Multichannel fast QR-decomposition RLS algorithms with explicit weight extraction. European Signal Processing Conference, EUSIPCO'2006, Florence, Italy (September 2006)
17. C. Eun and E. J. Powers, A new Volterra predistorter based on the indirect learning architecture. IEEE Transactions on Signal Processing, vol. 45, no. 1, pp. 20–30 (January 1997)

18. M. Bouchard, Multichannel recursive-least-squares algorithms and fast-transversal-filter algorithms for active noise control and sound reproduction systems. IEEE Transactions on Speech and Audio Processing, vol. 8, no. 5, pp. 606–618 (September 2000)
19. P. A. Regalia and M. G. Bellanger, On the duality between fast QR methods and lattice methods in least squares adaptive filtering. IEEE Transactions on Signal Processing, vol. 39, no. 4, pp. 876–891 (April 1991)
20. M. D. Miranda and M. Gerken, A hybrid least squares QR-lattice algorithm using a priori errors. IEEE Transactions on Signal Processing, vol. 45, no. 12, pp. 2900–2911 (December 1997)
21. A. H. Sayed, Fundamentals of Adaptive Filtering. John Wiley & Sons, Inc., Hoboken, NJ, USA (2003)
22. M. Shoaib, S. Werner, and J. A. Apolinário Jr., Reduced complexity solution for weight extraction in QRD-LSL algorithm. IEEE Signal Processing Letters, vol. 15, pp. 277–280 (February 2008)